UNIVERSITAT JAUME·I

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Ensembles of Artificial Neural Networks: Analysis and Development of Design Methods

Ph.D. Thesis

Presented by:

Joaquín Torres Sospedra

Supervised by:

Carlos A. Hernández Espinosa

Mercedes Fernández Redondo

Castellón, September 2011

Ensembles of Artificial Neural Networks: Analysis and Development of Design Methods

Ph.D. Thesis

Joaquín Torres Sospedra

2011

"To me it seems that those sciences are vain and full of error which are not born of experience, mother of all certainty, first hand experience which in its origins, or means, or end has passed through one of the five senses."

Leonardo Da Vinci 1452-1519.
Italian polymath, scientist, mathematician, engineer, inventor, anatomist, painter, sculptor, architect, botanist, musician and writer.

Abstract

This thesis is focused on the analysis and development of *Ensembles of Neural Networks*. An ensemble is a system in which a set of heterogeneous *Artificial Neural Networks* are generated in order to outperform the *Single-network* based classifiers. However, this proposed thesis differs from others related to ensembles of neural networks [1, 2, 3, 4, 5, 6, 7] since it is organized as follows.

In this thesis, firstly, an ensemble methods comparison has been introduced in order to provide a rank-based list of the best ensemble methods existing in the bibliography. This comparison has been split into two researches which represents two chapters of the thesis.

Moreover, there is another important step related to the ensembles of neural networks which is how to combine the information provided by the neural networks in the ensemble. In the bibliography, there are some alternatives to apply in order to get an accurate combination of the information provided by the heterogeneous set of networks. For this reason, a combiner comparison has also been introduced in this thesis.

Furthermore, *Ensembles of Neural Networks* is only a kind of *Multiple Classifier System* based on neural networks. However, there are other alternatives to generate *MCS* based on neural networks which are quite different to *Ensembles*. The most important systems are *Stacked Generalization* and *Mixture of Experts*. These two systems will be also analysed in this thesis and new alternatives are proposed.

One of the results of the comparative research developed is a deep understanding of the field of ensembles. So new ensemble methods and combiners can be designed after analyzing the results provided by the research performed. Concretely, two new ensemble methods, a new ensemble methodology called *Cross-Validated Boosting* and two reordering algorithms are proposed in this thesis. The best overall results are obtained by the ensemble methods proposed.

Finally, all the experiments done have been carried out on a common experimental setup. The experiments have been repeated ten times on nineteen different datasets from the *UCI* repository in order to validate the results. Moreover, the procedure applied to set up specific parameters is quite similar in all the experiments performed.

It is important to conclude by remarking that the main contributions are:

- An experimental setup to prepare the experiments which can be applied for further comparisons.
- A guide to select the most appropriate methods to build and combine ensembles and multiple classifiers systems.
- New methods proposed to build ensembles and other multiple classifier systems.

keywords

Multilayer Perceptron, *Radial Basis Functions*, *Multiple Classifier Systems*, *Ensembles of Neural Networks*, *Boosting methods*, *Cross-Validation Committees*, *Stacked Generalization* and *Mixture of Experts*.

Acknowledgments

This thesis has been developed at the *Neural Networks & Soft Computing* research group of the *Universitat Jaume I*. First of all, I am very grateful to my advisors, *Carlos Hernández-Espinosa* and *Mercedes Fernández-Redondo* for their guidance and support during its development. They provided extremely valuable help with the many difficulties that arose during this period. I have learned a great deal after all these years of working with them.

Secondly, I would like to thank *Pedro Gomez-Vilda* for the opportunity he gave me to collaborate in the project '*Modelado y clasificación automática de patrones de voz patológica para su aplicación clínica sobre Internet*'. This project was my first contact with a serious research and it encouraged me to continue my PhD studies.

Thanks in general to the *Department of Engineering and Computer Science* of the *Universitat Jaume I* for providing me the framework for carrying on my research and teaching work, and to the local and national institutions that have provide the funds for making this research possible. This thesis would not have been developed without the funds provided by the *Universitat Jaume I* and its *FPI doctoral grant (PREDOC/2005/44)* and the projects '*Modelado y clasificación automática de patrones de voz patológica para su aplicación clínica sobre Internet (TIC2002-02273)*' and '*Desarrollo de métodos de diseño de conjuntos de redes neuronales (P1-1B2004-03)*'. These grants not only have funded my research, some material required by the laboratory and research group has also been bought thanks to them.

I would also like to thank the members, current and former, of the laboratory and department who I have been working with for their warm welcome and their help and encouragement from the beginning. These staff have eased my life at work and they made me feel at home.

Finally, I would also like to specially thank *María Angeles López-Malo* for using their valuable time to introduce me to the field of researching. Her advice boosted my confidence so I decided to continue with a *PhD*.

I am specially thankful to *Patricio Nebot-Roglá* and *Gabriel Recatalá-Ballester*, for their friendship and their extremely valuable advices. Thanks also to all my friends who have made my years at the university so enjoyable. Among those already mentioned, thanks to Wirz, Juan Carlos, Xavi, "Van der Khul" football team, and many others, though not mentioned by name, they have not been less important to me.

And last but not least, to my wife *Ana Fortuño* and my family, for their capacity to listen, to advise and for their patience with me, and, above all, for giving me their unconditional support, regardless of what I was doing with this special *bussiness*.

Joaquín Torres Sospedra
Castellón, Mars 2011

Resumen

Esta tesis se centra en el análisis y desarrollo de *Conjuntos de Redes Neuronales*. Un conjunto es un sistema en el que un numero determinado de *Redes Neuronales Artificiales* heterogéneas se entrenan con el fin de superar el rendimiendo de clasificadores basados en una *única red*. Sin embargo, esta tesis difiere de las tesis relacionadas con los conjuntos de redes neuronales de otros investigadores [1, 2, 3, 4, 5, 6, 7] ya que en ella se ha realizado lo que detallamos a continuación.

En esta tesis, en primer lugar, se ha introducido una comparativa de métodos de diseño de conjuntos con el fin de proporcionar una lista ordenada con los mejores métodos de conjuntos. Esta comparativa se ha dividido en dos estudios los cuales se representan con dos capítulos de esta tesis.

Además, hay otro paso importante relacionado con los conjuntos de redes neuronales que es la forma de combinar la información proporcionada por las redes neuronales individuales. Se pueden aplicar diferentes alternativas para conseguir una combinación apropiada de las redes del conjunto. Por esta razón, se ha realizado en esta tesis una comparativa de combinadores.

Por otra parte, hay otras alternativas para generar sistemas basados en múltiples clasificadores que son muy diferentes a los métodos de conjuntos. Los sistemas más importantes son *Stacked Generalización* y *Mixutre of Experts*. Estos dos sistemas serán también analizados en esta tesis.

Fruto de los conocimientos adquiridos es la propuesta de nuevos métodos de conjuntos que aparecen en esta tesis. Concretamente estos son: dos nuevos métodos *Boosting*, una metodología de conjuntos llamada *Cross-Validated Boosting* y dos algoritmos de reordenamiento. Los métodos de diseño de conjuntos que hemos propuesto en esta tesis, en general, obtienen los mejores resultados.

Por último, todos los experimentos realizados se han llevado a cabo en un marco experimental común. Los experimentos se han repetido diez veces en diecinueve bases de datos del *repositorio UCI* con el fin de validar los resultados. Por otra parte, el procedimiento aplicado para establecer los parámetros específicos es bastante similar en todos los experimentos realizados.

Es importante concluir señalando que las principales contribuciones de esta tesis son:

- Un marco experimental para preparar los experimentos que se pueden aplicar para otras futuras comparaciones.
- Una guía para seleccionar los métodos más adecuados para crear y combinar conjuntos de clasificadores y sistemas de múltiples clasificadores.
- Nuevos métodos de diseño de conjuntos de redes neuronales y otros Sistemas de Múltiples Clasificadores.

Palabras clave

Perceptron Multicapa, *Funciones de Base Radial*, *Sistemas de Múltiples Clasificadores*, *Conjuntos de Redes Neuronales*, *Sobre entrenamiento de patrones conflictivos*, *Validación Cruzada*, *Generalización Apilada* y *Mezcla de Expertos*

Agradecimientos

Esta tesis se ha desarrollado en el grupo de investigación *Redes Neuronales y Computación Suave* de la *Universitat Jaume I*. En primer lugar, estoy muy agradecido a mis directores, *Carlos Hernández Espinosa* y *Mercedes Fernández Redondo* para su orientación y apoyo durante el desarrollo de esta tesis. Su ayuda ha sido primordial para superar las numerosas dificultades que surgieron durante este período. He aprendido mucho después de todos estos años de trabajo con ellos.

En segundo lugar, quisiera dar las gracias a *Pedro Gómez Vilda* por la oportunidad que me dió para colaborar en el proyecto "*Modelado y clasificación automática de patrones de voz patológica para su Aplicación Clínica sobre Internet*". Este proyecto fue mi primer contacto con una investigación seria y me animó a continuar con mis estudios de doctorado.

Gracias en general al *Departamento de Ingeniería y Ciencias de los Computadores* por proporcionarme el material y apoyo para realizar mi trabajo de investigación y docente, y a las instituciones locales y nacionales con cuyos fondos ha sido posible realizar esta investigación. Esta tesis no se habría podido desarrollar sin los fondos aportados por la *Universitat Jaume I* y su programa de *Becas FPI (PREDOC/2005/44)* y por los proyectos '*Modelado y clasificación automática de patrones de voz patológica para su Aplicación Clínica sobre Internet (TIC2002-02273)*' y '*Desarrollo de métodos de diseño de los conjuntos de redes neuronales (P1-1B2004-03)*'. Estas ayudas no sólo han financiado mi investigación, también algunos materiales requeridos para el grupo de investigación han podido ser adquiridos gracias a ellas.

También me gustaría agradecer a los miembros, actuales y anteriores, del laboratorio y el departamento con el que he estado trabajando por su cálida acogida y su ayuda desde el principio. Estos compañeros han facilitado mi vida en el trabajo y me hicieron sentir como en casa.

Por último, también quiero agradecer especialmente a *María Ángeles López Malo* por utilizar su valioso tiempo para introducirme en el campo de la investigación. Sus consejos me ayudaron a tomar la decisión de realizar el programa de doctorado.

Estoy especialmente agradecido a *Patricio Nebot Roglá* y *Gabriel Recatalá Ballester*, por su amistad y sus consejos muy valiosos. Gracias también a todos los amigos que han hecho mis años en la universidad tan agradable. Entre los ya mencionados, gracias a Juan Carlos, Xavi, "Van der Khul" equipo de fútbol, Wirz, y otros muchos, que aunque no menciono, no han sido menos importantes para mí.

Y por último, pero no menos importante, a mi mujer *Ana Fortuño* y mi familia, por su capacidad de escuchar, para asesorar y por su paciencia conmigo, y, sobre todo, por haberme dado su apoyo incondicional, independientemente de lo que estaba haciendo con este *negocio* tan especial.

Joaquín Torres Sospedra
Castellón, Marzo de 2011

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

- ***ACB*:** Ensemble method "*Averaged-Conservative Boosting*"
- ***Adaboost*:** Ensemble method "*Adaptive Boosting*"
- ***Aggreboost*:** Ensemble method "*Aggressive Boosting*"
- ***ANN*:** Artificial Neural Network
- ***ARCx4*:** Ensemble method "*Arcing Classifier*"
- ***ATA*:** Ensemble method "*Adaptive Training Algorithm*"
- ***ATA-BE*:** Ensemble method "*Adaptive Training Algorithm, selecting the Best Epoch*"
- ***ATA-LE*:** Ensemble method "*Adaptive Training Algorithm, selecting the Last Epoch*"
- ***Aveboost*:** Ensemble method "*Averaged Boosting*"
- ***Aveboost2*:** Ensemble method "*Averaged Boosting version 2*"
- ***BADD*:** Combiner "*Basic Defuzzyfication Distribution*"
- ***Bagging*:** Ensemble method "*Bootstrap Aggregating*"
- ***Bagnoise*:** Ensemble method "*Bootstrap Aggregating with noise*"
- ***CELS*:** Ensemble method "*Cooperative Ensemble Learning System*"
- ***Choqet ddd*:** Combiner "*Choquet Integral with Data-Dependent Densities*"
- ***CN*:** Combination Network (in *Stacked Generalization* model)
- ***Conserboost*:** Ensemble method "*Conservative Boosting*"
- ***CVC*:** Ensemble method "*Cross-Validation Committee*"
- ***CVCv1*:** Ensemble method "*Cross-Validation Committee version 1*"
- ***CVCv2*:** Ensemble method "*Cross-Validation Committee version 2*"
- ***CVCv2.5*:** Ensemble method "*Cross-Validation Committee version 2.5*"
- ***CVCv2ACB*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 2*" and "*Averaged-Conservative Boosting*"

- ***CVCv2Adaboost*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 2*" and "*Adaptive Boosting*"
- ***CVCv2Aggreboost*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 2*" and "*Aggressive Boosting*"
- ***CVCv2ARCx4*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 2*" and "*Arcing Classifier*"
- ***CVCv2Aveboost*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 2*" and "*Averaged Boosting*"
- ***CVCv2Aveboost2*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 2*" and "*Averaged Boosting version 2*"
- ***CVCv2Conserboost*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 2*" and "*Conservative Boosting*"
- ***CVCv2Inverboost*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 2*" and "*Inverse Boosting*"
- ***CVCv2TCAv1*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 2*" and "*Totally Corrective Adaptive Boosting version 1*"
- ***CVCv2TCAv2*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 2*" and "*Totally Corrective Adaptive Boosting version 2*"
- ***CVCv2WCB*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 2*" and "*Weighted-Conservative Boosting*"
- ***CVCv3*:** Ensemble method "*Cross-Validation Committee version 3*"
- ***CVCv3ACB*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 3*" and "*Averaged-Conservative Boosting*"
- ***CVCv3Adaboost*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 3*" and "*Adaptive Boosting*"
- ***CVCv3Aggreboost*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 3*" and "*Aggressive Boosting*"
- ***CVCv3ARCx4*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 3*" and "*Arcing Classifier*"
- ***CVCv3Aveboost*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 3*" and "*Averaged Boosting*"
- ***CVCv3Aveboost2*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 3*" and "*Averaged Boosting version 2*"
- ***CVCv3Conserboost*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 3*" and "*Conservative Boosting*"

- ***CVCv3Inverboost*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 3*" and "*Inverse Boosting*"
- ***CVCv3TCAv1*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 3*" and "*Totally Corrective Adaptive Boosting version 1*"
- ***CVCv3TCAv2*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 3*" and "*Totally Corrective Adaptive Boosting version 2*"
- ***CVCv3WCB*:** New *Cross-Validated Boosting* ensemble based on "*Cross-Validation Committee version 3*" and "*Weighted-Conservative Boosting*"
- ***DAN*:** Combiner "*Dynamically Averaged Networks*"
- ***DANv1*:** Combiner "*Dynamically Averaged Networks version 1*"
- ***DANv2*:** Combiner "*Dynamically Averaged Networks version 2*"
- ***DECO*:** Ensemble method "*Decorrelated*"
- ***DP*:** Ensemble method "*Disjoint Partitions*"
- ***DPR*:** Ensemble method "*Disjoint Partitions with replications*"
- ***EENCL*:** Ensemble method "*Evolutionary Ensemble with Negative Correlation Learning*"
- ***EENCL-BG*:** Ensemble method "*Evolutionary Ensemble with Negative Correlation Learning, selecting the Best Generation*"
- ***EENCL-LG*:** Ensemble method "*Evolutionary Ensemble with Negative Correlation Learning, selecting the Last Generation*"
- ***EN*:** Expert Network (in *Mixture of Experts* and *Stacked Generalization* models)
- ***EVOL*:** Ensemble method "*Ensembles Voting On-Line*"
- ***FSCL*:**"*Frequency Sensitive Competitive Learning*" (clustering algorithm)
- ***GN*:** Gating Network (in *Mixture of Experts* model)
- ***Inverboost*:** Ensemble method "*Inverse Boosting*"
- ***IoP*:** Increase of Performance (measurement)
- ***KNN*:** K-Nearest Neighbors
- ***MCS*:** Multiple Classifiers System
- ***MF*:** Multilayer Feedforward (network architecture)
- ***MFSE*:** *Simple Ensemble* of Multilayer Feedforward Networks
- ***MSE*:** Mean Squared Error

- ***NCL*:** Ensemble method "*Negative Correlation Learning*"
- ***OLA*:** Ensemble method "*Observational Learning Algorithm*"
- ***OP*:** Ensemble method "*Overlapping Partitions*"
- ***OPR*:** Ensemble method "*Overlapping Partitions with replications*"
- ***PER*:** Percentage of Error Reduction (measurement)
- ***RBF*:** Radial Basis Functions (network architecture)
- ***RBFSE*:** *Simple Ensemble* of Radial Basic Functions
- $\boldsymbol{RBFSE}_{\mathbf{min-max}}$**:** *Simple Ensemble* of Radial Basic Functions with mix-max normalization
- $\boldsymbol{RBFSE}_{\mathbf{sum}}$**:** *Simple Ensemble* of Radial Basic Functions with sum normalization
- $\boldsymbol{RBFSE}_{\mathbf{threshold}}$**:** *Simple Ensemble* of Radial Basic Functions with threshold normalization
- ***SE*:** Ensemble method "*Simple Ensemble*"
- ***SE**:** Special version of the ensemble method "*Simple Ensemble*"
- ***SG*:** Stacked Generalization
- ***SN*:** Single Network
- ***STC*:** "*Stacked Generalization*" combiner
- ***STC+*:** "*Stacked Generalization plus*" combiner
- ***STCP*:** "*Stacked Generalization plus*" combiner
- ***TCA*:** Ensemble method "*Totally Corrective Adaptive Boosting*"
- ***TCAv1*:** Ensemble method "*Totally Corrective Adaptive Boosting version 1*"
- ***TCAv2*:** Ensemble method "*Totally Corrective Adaptive Boosting version 2*"
- ***UCI*:** University of California, Irvine
- ***W. Ave*:** Combiner "*Weighted Average*"
- ***W.Ave ddw*:** Combiner "*Weighted Average with Data-Dependent Weights*"
- ***WCB*:** Ensemble method "*Weighted-Conservative Boosting*"
- ***WTA*:** Combiner "*Winner Takes All*"

Chapter 1

Introduction

1.1 Historical Background

The research on *Artificial Neural Networks* started in 1943 when McCulloch Pitts discovered that an increase in computational resources could be obtained by combining simple processing units [8]. Nowadays, some ideas he suggested are still being used.

After a few years of research done by Donald O. Hebb [9], H. D. Block [10] and Frank Rosenblatt [11, 12], Minsky and Papert demonstrated in [13] that the perceptron has some limitations that do not allow it to solve non-linear separable classification problems. A simple function like the XOR problem could not be implemented by a perceptron.

In the 80's, the research on the field of Neural Networks revived. In 1989 the director of the Defense Advanced Research Projects Agency, henceforth *DARPA*, of United States started funding projects related to Neural Networks. Moreover, John Hopfield, a well-known American scientist, started working on the Neural Networks field and published two important papers [14, 15]. Furthermore, David B. Parker [16] and Yann LeCun [17] proposed simultaneously a new learning algorithm to train neural networks which could solve non-linear functions. This algorithm was called *Backpropagation* and the XOR problem could be solved with it.

Finally, in June 1987 the First International Joint Conference on Neural Networks, also known as *IJCNN*, took place in Washington, USA. Then, this research field was consolidated by other important publications and conferences.

1.2 Motivation and Objectives

Although a neural network can be used for signal processing and data segmentation, among other disciplines, in the current research they are going to be used to solve classification problems. Moreover, there are some areas in which a single neural network can be applied as medicine, trade, statistics, among others, as indicated in references [18, 19]. However, important improvements are being done in the field called *Multiple Classifiers Systems* (*MCS*) based on neural networks. Better results in classification can be obtained if a set of different networks collaborate, in a cooperative or in a competitive way, in order to solve a problem.

Analogously, in the real world, humans and other biological systems tend to collaborate, in some different ways, in order to achieve a main goal or a portion of it as shown in the next figures. Sometimes, collaboration requires strict rules or a centered supervision.

Concretely, figure 1.1 (a) shows some firefighters collaborating in order to extinguish a fire. Although all of them are cooperating and sharing the same goal, some firefighters are close to the fire whereas there are others on a helicopter (although they do not appear on the picture). They are not doing exactly the same, but they share the same objective. This is an example of cooperation in which all the elements, in this case firefighters, are solving the same problem.

(a) (b)

Figure 1.1: Two different collaborations

Figure 1.1 (b) shows a flat or an office which is being rebuilt. This figure is quite representative because there are bricklayers, plumbers, electricians and painters performing specific tasks which are part of a bigger task, the repair of the flat. This is an example of cooperation in which each *element* is solving a specific part of a task.

In both cases, there is at least a supervisor who controls the elements of the system. There is a team or a person who coordinates the firefighters, specially in severe cases. In the case of the office, two people are clearly supervising the jobs done in the office.

Nowadays, some advances in classifiers are given by different kinds of *Multiple Classifier Systems*. There are cooperative and competitive models which can be successfully applied to neural networks in order to generate accurate classifiers. These models are inherited by our nature and our society because cooperating and collaborating is the first option to overcome big problems in daily life. We consider that *Multiple Classifier Systems* based on neural networks are quite interesting so they should be deeply analysed and compared. Moreover, an appropriate mechanism to fuse the information provided by the networks should be also strongly considered for a deep study as an important part of the design of a *MCS*.

However, the proposed methods in the literature, to design or combine the networks of an ensemble, are introduced without performing an exhaustive or representative comparison with other traditional methods. In most of the cases, there are only references to *Bagging* and *Boosting* approaches as important ensemble methods. Moreover, the criteria to setup the experiments and validate the results depends on the author. So comparing the methods that have been proposed in the bibliography is nearly impossible with the results provided by each author, even if the same datasets were used. Two methods can not be directly compared if the experimental setup differs.

Additionally, some recent ensemble methods or combiners are proposed exclusively to solve an own specific problem. These proposals do not include information about the accuracy of the proposed system on other problems or datasets. So comparing these new methods to traditional ones is impossible.

Furthermore, there are a huge number of methods to design and build ensembles of neural networks. Also, the research on ensembles is still important as recent papers published in the most important conferences on neural networks show.

For these exposed reasons, we think that the importance of the proposal of new methods is highly reduced if there is not an objective and robust basis to measure the accuracy or performance of the proposed method and to establish the rules for a further comparison with other methodologies.

We think that it is necessary to perform an exhaustive study of the different ensemble methods and ensemble combiners proposed in the bibliography. On the one hand, we will establish the basis and criteria to measure the accuracy of ensembles. On the other hand, we will propose a rank based list of the best ensemble alternatives according to its performance.

Finally, the study of traditional ensemble methods and other models improves the knowledge on this field. So the proposal of new ensembles is easier and possible. In this thesis, new ensemble alternatives and models have been proposed by modifying a traditional ensemble or by mixing some different approaches into a single procedure.

We can conclude by remarking that the main objectives of this thesis originally were:

- Perform a complete comparison of methods to design/build Multiple Classifiers Systems. A high number of alternatives will be analyzed and their performance will be tested carrying out a common experimental setup.
- Perform a deep comparison on ensembles combiners. Similarly, the comparison among combiners will not be reduced to simple or well known procedures. Some methodologies will be analyzed and tested.
- Analyze more complex Multiple Classifier Systems, such as *Stacked Generalization* and *Mixture of Experts*, and adapt them to the ensemble approach if possible.
- Design new ensemble methods. In this thesis, we propose some new ensemble variants, a new ensemble methodology called *Cross-Validated Boosting* and two reordering algorithms.

1.3 Tools

A high number of alternatives to generate multiple classifiers systems along with ensemble combiners have been implemented to develop the current dissertation. To carry out these experiments, a large capacity of calculus and digital storage have been required.

Nowadays, the *Neural Networks and Soft Computing* research group (*NN*&SC) of *Universitat Jaume I* has several computers in the laboratory *TI-1114-DL* (figure 1.3), which are powered by *Intel Xeon* processor, exclusively for research tasks. Moreover, each computer have enough main memory to run the experiments and enough secondary memory to temporally store the data generated. Finally, there are two RAID 5 systems to keep the generated data before storing it on *DDS4* tapes.

Recently, some new computers based on *Intel Core Quad* and *Intel i7* technology have been added to the laboratory so the computational capacity has been highly increased. Moreover, a new digital *LTO2* tape recorder has been bought to definitively store the generated data. Finally, a new 3GB RAID 5 system has been introduced in a Xeon computer to temporally store the data.

The laboratory equipment has been shared with other members of the *Neural Networks and Soft Computing* group and undergraduated students. Concretely, Maria del Carmen Ortiz Gomez was researching on the optimization of the *RBF* network design for her *PhD* thesis whereas Elena Prades Carceller performed some experiments for her final undergraduated project.

The supervisors of this thesis, Carlos Antonio Hernández Espinosa and María de las Mercedes Fernández Redondo, have also performed some experiments related to different projects in the laboratory.

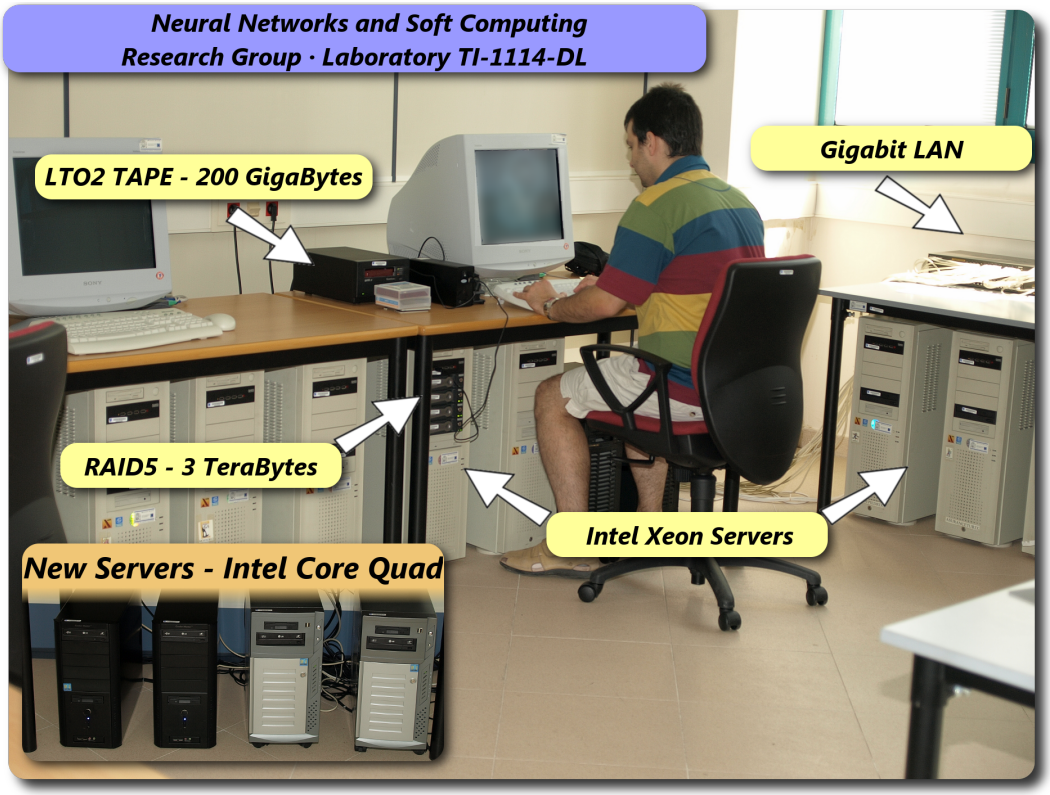


Figure 1.2: *NN&SC* Laboratory

The programming language chosen to implement the simulations is MatLab. MatLab is a development system whose operations on vectors and matrices are highly optimized so it perfectly fits on developing neural networks. Arithmetical operations related to high-sized matrices are often required in the training procedure of neural networks, [20].

Finally, only free software has been utilized in order to prepare this thesis and the published papers. LaTeX has been used to write the thesis whereas the figures, plots, graphs and diagrams have been prepared with *Open Office*. Some books and manuals related to LaTeX [21, 22, 23] and *Open Office* have been reviewed.

1.4 Contributions

All the main objectives previously described have been successfully reached as it is shown in this thesis. The main contributions, in a general way, derived from the research performed are:

- A base to prepare the experiments and simulations for further comparisons is proposed in this thesis. This procedure is better than a criteria based on each author, because when a new method is proposed their results can be easily compared with the rest of the methods employed in this thesis in a common experimental configuration, like the setup we propose and has been utilized.
- A guide to select the most appropriate methods to design ensembles and multiple classifiers systems. In the conclusions chapter, a reduced list of methods or models to build ensembles and *Multiple Classifier Systems* (*MCS*) is proposed as methods or models that should be seriously considered for classification tasks. This recommendation is the final conclusion of the comparisons done.
- A ranking of the best combiners, the procedures to fuse the networks which fit better on a wide number of databases and ensemble alternatives.
- New methods to build ensembles and multiple classifier systems. After reviewing and implementing traditional alternatives, new ones have been proposed. Some of this new variants have provided excellent results on several datasets and they also are on the top of the best performing ensembles shown in the comparison ranking.
- New combiners applied to ensembles of neural networks. The experiments done in this thesis show that some of these new ways used to fuse the networks of an ensembles have improved the performance of the original ensembles.

Furthermore, the global research done in this thesis has been published in the most important conferences as we will show in the last chapter related to the general conclusions.

1.5 Thesis outline

The rest of this thesis is organized as follows:

- Concepts and first experiments related to *Pattern Recognition*, *Neural Networks Architectures* and *Multiple Classifier Systems* will be reviewed in chapters 2 and 3.
- Chapters 4 and 5 will be focused on the comparison of methods to design ensembles of neural networks. The ensemble methods analyzed have been divided into two categories depending on their nature. In the first category, the ensemble variants analyzed are those in which the training algorithm or network structure is modified. The second category is based on other alternatives in which the modifications are applied to the learning set.
- Chapter 6 will deal with the procedures used to combine ensembles of neural networks. In this chapter, the combiners reviewed will be applied to the most representative ensemble methods.
- Some reordering procedures of the training set will be also applied to the traditional ensembles in chapter 7. These reordering procedures are proposed in this thesis and they will randomly alter the sequence of patterns so the training will not be viced with the same sequence. They add diversity to the different networks of the ensemble.
- Some new boosting variants have also been proposed and they will be described and analyzed in chapter 8. Firstly, some basic improvements are proposed and applied to boosting. Finally, we successfully introduce a completely new methodology in which *Boosting* is mixed in another important ensemble approach, *Cross-Validation*.
- Other *Multiple Classifier Systems* will be analyzed in chapters 9 and 10. The *Stacked Generalization* model is described and studied as a *Multiple Classifier System MCS* and as an ensemble combiner in the first chapter whereas the second chapter will be focused on the *Mixture of Experts* model.
- Finally, the last chapter will provide the general conclusions and we will describe some lines of future work.
- The appendixes will show:
  - The description of the datasets used in the experiments performed in this thesis.
  - The specific parameters applied in the experiments for the networks, ensembles and more complex models.
  - The complete raw results obtained by the ensembles and MCS we have built.

Chapter 2

Pattern Recognition with Artificial Neural Networks

2.1 Introduction

The current chapter will be focused on describing the basic characteristics of Artificial Neural Networks, henceforth *ANN*, when they are applied to solve classification problems. Firstly, some preliminary definitions will be shown. Secondly, the two most widely used architectures along with the process to train them will be described: *Multilayer Feedforward* and *Radial Basis Functions*.

2.2 Preliminar definitions

Here, some basic concepts related to neural networks will be defined and introduced.

2.2.1 Pattern recognition and Classification

Defining *Pattern Recognition* is not a simple task, in reference [24] we can see the following interesting definition:

> "Pattern recognition is about assigning labels to objects. Objects are described by a set of measurements called also attributes or features."

A pattern is a set of attributes that represent an object or situation from the real world whereas the label represents the class which contains this object along with similar objects.

Pattern recognition can be seen as the act of processing raw data and taking a decision based on the category of the data. In classification problems, a classifier assigns the most appropriate class label to a pattern.

2.2.2 Artificial Neural Networks

It is a complex task to define exactly what a neural network is. There are some definitions that depend on the viewpoint of the author.

A basic definition of an artificial neural network can be found in [25]:

> "A neural network is a processing device, either an algorithm, or actual hardware, whose design was motivated by the design and functioning of human brains and components thereof."

In [26], an artificial neural network is defined as follows:

> "An ANN is a network of many very simple processors (*units*), each possibly having a local memory. The units are connected by unidirectional communication channels (*connections*), which carry numeric (as opposed to symbolic) data. The units operate only on their local data and on the inputs they receive via the connections."

The first definition described the basic functionality of an artificial neural network. The second one focuses on the elements which process the information and their interconnections. However, there are some gaps in those definitions because the learning process is not specified and there are not any limitations in its structure.

Artificial neural networks will be used to solve classification problems in this thesis. However, they can also be used signal processing or data segmentation.

2.2.3 Network Architecture

The network architecture specifies internal units and how they are organized. Reviewing the bibliography it can be seen that the most widely used architectures of neural networks are the following ones:

- *Multilayer Feedforward.*
- *Radial Basis Functions.*
- *Kohonen self-organizing network.*
- *Hopfield network.*
- *Boltzmann machines.*
- *Neuro-fuzzy networks.*

Although there is an important number of alternatives, *Multilayer Feedforward* networks, henceforth *MF*, and *Radial Basis Functions*, henceforth *RBF*, will be researched in this thesis. Both selected architectures are well known and they are the most extended in the bibliography. Their architecture specifications along with the corresponding learning procedures will be reviewed in this chapter.

2.2.4 Learning process

There are some classification systems, like the artificial neural networks studied in this chapter, that classify data based on *a priori* knowledge. In the case of artificial neural networks, this a priori knowledge is given by the learning process.

During the learning process, known patterns of a particular classification problem are presented to the network in order to improve its performance and its generalization ability. The generalization capability is the ability to correctly respond to patterns which were not used during the training process. Usually, an optimization function method based on gradient descent is applied in order to minimize the error or maximize the accuracy of the network.

The learning process can be *Supervised* or *Not supervised*. A learning process is *Supervised* when the class label of the patterns presented to the network is known and used in the training process. If the class label is unknown or unused, the learning process is *Not supervised*. The most important training algorithms, like *Backpropagation*, are *Supervised*. The not supervised learning processes are often used, for instance, to know how the pattern are distributed on the input space.

This thesis will be centered on neural networks trained with supervised learning algorithms. However, some ensemble combiners and MCS models will use not supervised learning algorithms.

2.3 Artificial Neural Networks

As we mentioned previously, we are going to work with *Artificial Neural Networks* in the case of solving classification problems. In this section the internal operation of ANNs and the two most widely used network architectures along with their training algorithms are going to be described.

2.3.1 Neurons

The biologic neurons are the information process elements of the human brain and they are strongly interconnected. A biological neuron is composed by three main components: The dendrites, the soma and the axon. The dendrites of a neuron are cellular extensions with many branches which send to the soma the electric signals received. The soma is the central part of the neuron. Its main task is processing all the signals received. The axon carries the output nerve signals away from the soma. Many neurons have only one axon, but this axon may undergo extensive branching, enabling communication with many target neurons. The synapses are specialized structures where chemicals neurotransmitter are released in order to communicate with target neurons. In figure 2.1 a graphical description of a neuron and their interconnection is shown.

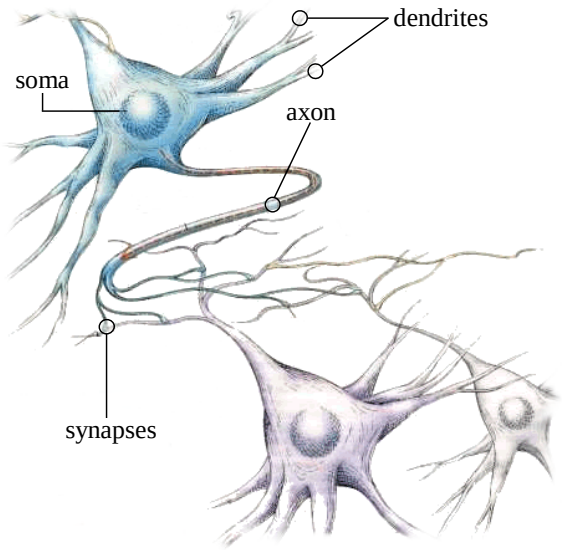


Figure 2.1: Biologic Neurons - Structure and Interconnection

On the other hand, the artificial neurons try to imitate the biological neurons behavior. The artificial neurons receive some values from the input sensors or other artificial neurons and these values are mathematically processed in order to get an output value.

In figure 2.2 a simple artificial neuron model is shown. In this model, a transfer function ϕ is applied to the summatory of the weighted values arriving from the input sensors.

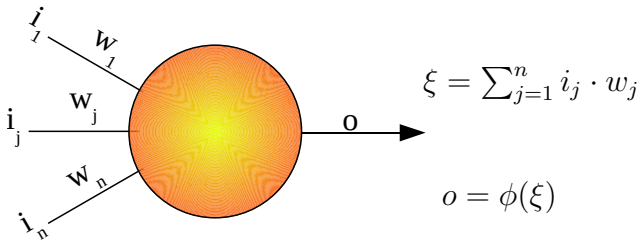


Figure 2.2: Artificial neuron description

The most frequently used transfer functions are the following ones:

- The *Sigmoid* transfer function:

$$\phi_{sig}(\xi) = \frac{1}{1 + \exp(-\xi)} \tag{2.1}$$

- The *Identity* transfer function:

$$\phi_{id}(\xi) = \xi \tag{2.2}$$

- The *Step* transfer function:

$$\phi_{step}(\xi) = \begin{cases} 1, & \text{si } \xi > 0 \\ 0, & \text{otherwise,} \end{cases} \tag{2.3}$$

It has been demonstrated that a *Multilayer Feedforward* network with a single hidden layer and step nodes can approximate *any* function with a specified precision. Unfortunately, the most important training algorithms, like *Backpropagation*, require the use of a derivative transfer function and it is not possible to calculate the derivate of the step function. The sigmoid transfer function has a similar behavior to the step function, as it can be seen in figure 2.3, and it is possible to calculate its derivate. For these reasons, the sigmoid transfer function is the most often used in some neural networks.

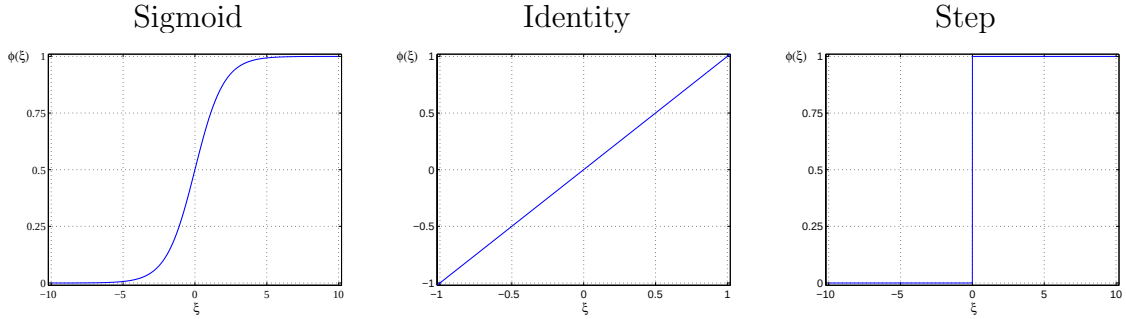


Figure 2.3: Graphical representation of the different transfer functions

2.3.2 The Multilayer Feedforward architecture

The *Multilayer Feedforward* architecture, henceforth *MF*, is one of the most extended network topologies, we can see the full description in [18] and [19]. In a *MF* network, the neurons are organized in independent layers. Each layer is composed by a few independent neurons which only process the information provided by the elements of the previous layer. Once the information is processed, the results are sent to the next layer.

The *MF* network is composed by the following layers:

- *Input layer.*
- One or several *Hidden layers.*
- *Output layer.*

The input layer is the layer which transmits the input information of the incoming pattern to the hidden layers. The number of neurons of this layer corresponds to the number of input parameters or characteristics of the problem. The identity transfer function is applied to the neurons of this layer.

The hidden layers are in charge of processing the information coming from the previous layer. The sigmoid transfer function is applied to the neurons of these layers. The number of units depends on the classification problem.

The output layer is the layer which processes the information provided by the last hidden layer. This layer provides the final output of the classification process by applying the sigmoid transfer unit to each neuron of the layer. The number of units corresponds to the number of output parameters or classes of the problem.

Although a *MF* network can be composed by several hidden layers, a single hidden layer is generally enough. A *MF* with a single hidden layer and with *step* nodes can approximate any function with a defined precision [27, 28]. In this thesis we will use in our experiments a single hidden layered version of the *MF* network with *sigmoid* nodes.

Furthermore, the connections between the input layer and the hidden layer, and the connections between the hidden layer and the output layer are weighted. These weighted connections will be denoted as $wih_{i,j}$ and $who_{j,k}$ respectively.

A learning process, also called training, has to be applied to establish the appropriate values of the connection weights. The first step consists on setting random values to these weights. Then an iterative training algorithm is applied in order to adjust the weights and make them to converge to an 'optimal' status. Usually, these training algorithms are based on gradient descend.

Finally, the diagram of the *Multilayer Feedforward* network architecture is shown in figure 2.4.

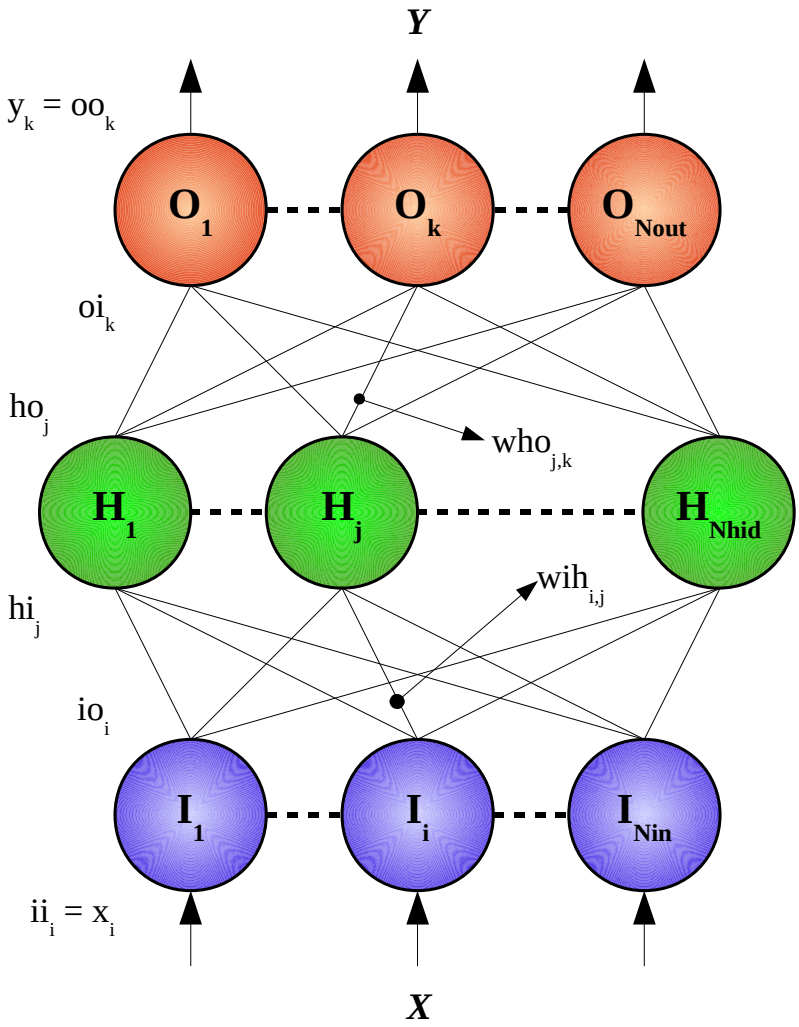


Figure 2.4: Multilayer Feedforward Network

For the input layer, the input values, ii, and the output values, io, correspond to the value of the input parameters because the identity transfer function is applied in the input layer units.

$$ii_i = io_i = x_i \tag{2.4}$$

For the hidden layer, the input values, hi, and the output values, ho, are given by equations 2.5 and 2.6. We can notice in equation 2.6 how the sigmoid transfer function is applied.

$$hi_j = \sum_{i=1}^{Ninputs} wih_{i,j} \cdot x_i + \theta_j \tag{2.5}$$

Where θ_j is a threshold value which is adapted during the training procedure.

$$ho_j = \frac{1}{1 + \exp(-hi_j)} \tag{2.6}$$

For the output layer, the input values, oi, and the output values, oo (also denotated by y), are given by equations 2.7 and 2.8. We can also notice in equation 2.8 how the sigmoid transfer function is applied. Analogously, ξ_k in equation 2.7 is a threshold value which is also adapted during the training procedure.

$$oi_k = \sum_{j=1}^{Nhidden} who_{j,k} \cdot ho_j + \xi_k \tag{2.7}$$

$$y_k = oo_k = \frac{1}{1 + \exp(-oi_k)} \tag{2.8}$$

Where:

- N_{inputs} (N_{in} in the previous figure) is the number of input parameters of the problem.
- N_{hidden} (N_{hid} in the previous figure) is the number of units of the hidden layer.
- $N_{classes}$ (N_{out} in the previous figure) is the number of classes of the problem.
- $wih_{i,j}$ corresponds to the value of the weighted connection between neuron i of the input layer and neuron j of the hidden layer.
- $who_{j,k}$ corresponds to the value of the weighted connection between neuron j of the hidden layer and neuron k of the output layer.

In the previous equations, we can notice that the input value of the neurons of the hidden layer and output layer, hi_j and oi_k, corresponds to the weighted summatory of the output of the previous layer plus a threshold value.

To implement the threshold values in a practical way, a special neuron is added to its input layer and to the hidden layer. This special neuron always have the value 1 at output so the threshold values always correspond to the value of the weighted connections of this neuron. In this way, the equations of the weights of the training algorithm can also be applied to the thresholds.

Once the output of the network is calculated, the class to which the pattern x corresponds is determined by equation 2.9.

$$Class(x) = \underset{c=1,\ldots,Nclasses}{\arg\max} \; (y_c(x)) \tag{2.9}$$

With this equation we calculate the predicted class for an input pattern x and the process of recognition has finished. The function $arg\ max$ represents the argument, c, with maximum value, $y_c(x)$. In this case, the pattern x belongs to the class c whose associated output y_c has the highest value.

2.3.3 Training a Multilayer Feedforward Network

Backpropagation is a supervised learning algorithm commonly applied to train *MF* networks. The learning process with this algorithm is based on the minimization of the *Mean Squared Error* (*MSE*). This error, denoted in equation 2.10, measures the difference between the output provided by the network for pattern x, *y(x)*, and the desired output or target of the pattern x in the training set, *d(x)*.

$$Error(x) = \frac{1}{2} \cdot \sum_{c=1}^{N_c} (d_c(x) - y_c(x))^2 \tag{2.10}$$

Where the desired output d is given by:

$$d_c(x) = \begin{cases} 1 & \text{if } x \in class\ c \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

The algorithmic description of the learning procedure is described in algorithm 2.1. Whereas a graphical representation of an epoch is shown in figure 2.5.

Algorithm 2.1 MF Network Training$\{T, V\}$

```
Set initial weights randomly
for e = 1 to epochs do
  for x = 1 to N_patterns do
    Adjust the value of the weights who and wih
  end for
  Calculate MSE over validation set V
  Save epoch weights and calculated validation MSE
end for
Select epoch with lowest validation MSE
Assign best epoch configuration to the network
```

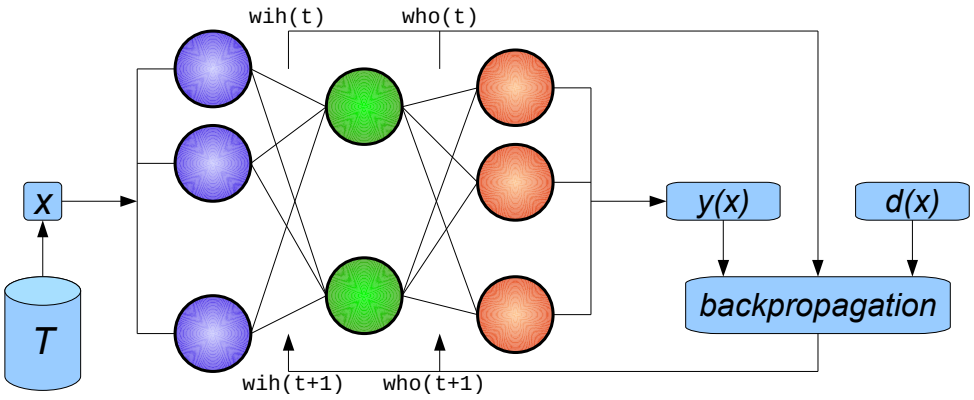


Figure 2.5: Graphical description of an iteration of Backpropagation

In *Backpropagation*, equation 2.12 is applied in order to adjust the weights of the *MF* networks.

$$W(t+1) = W(t) - \eta \frac{\partial Error}{\partial W}(t) + \alpha \cdot (W(t) - W(t-1)) \tag{2.12}$$

Where η is the adaptation step, W is a particular weight and α is the momentum rate. Evaluating the derivate, equation 2.13 is obtained to adjust the weights of the connections between the hidden layer and the output layer.

$$\frac{\partial Error}{\partial W}(t) = \frac{\partial Error}{\partial who_{j,k}}(t) = -\delta_k \cdot ho_j \tag{2.13}$$

Where ho_j is the output of the j-th hidden node (2.6) and:

$$\delta_k = (d_k - y_k) \cdot (1 - y_k) \cdot y_k \tag{2.14}$$

Furthermore, equation 2.15 can be applied to adjust the weights of the connections between the input layer and the hidden layer.

$$\frac{\partial Error}{\partial W}(t) = \frac{\partial Error}{\partial wih_{i,j}}(t) = -\left(ho_j \cdot (1 - ho_j) \cdot \sum_{h=1}^{Noutput} \delta_h \cdot who_{j,h} \right) \cdot x_i \tag{2.15}$$

To perform the experiments, the original datasets of each database have been divided into three different subsets. The first set is the training set T which is used to adapt the weights whereas the second set is the validation set V which is used to select the network configuration with the best estimated generalization capability. Finally, the third set is the test set TST which is used to obtain the accuracy of the network and the final results.

When we refer to the original learning set L, we really refer to the union of the training and validation sets, the only sets which are involved on the learning procedure. A graphical description of the procedure applied to generate the specific sets is shown in figure 2.6.

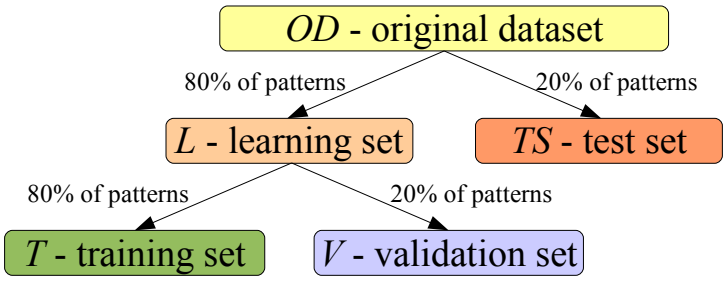


Figure 2.6: Generating the Training, Validation and Test sets

The major problem of this training algorithm is *overfitting*. When overfitting occurs, the network is only able to correctly classify the patterns from the training set and the network accuracy drastically decreases with new patterns which were not used in training.

Figure 2.7 depicts an example of *overfitting*. In the figure, we can notice how the *MSE* on the validation set increases from 1000 to 5000 epochs. For this reason the final network is based on the MSE on the validation set.

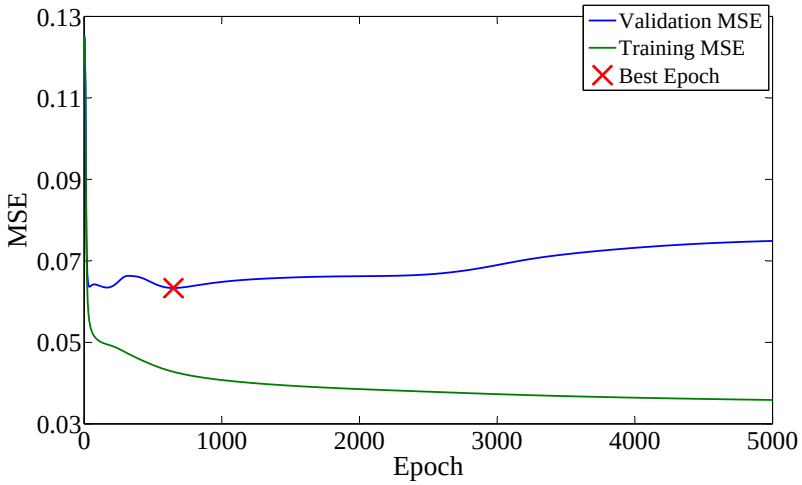


Figure 2.7: Mean Square Error on Training and Validation sets

2.3.4 The Radial Basis Functions architecture

The *Radial Basis Functions*, henceforth *RBF*, is another network architecture commonly applied to classification problems. It has been demonstrated that, in some cases, the results that have been obtained with *RBF* networks are better than the results of *MF* networks. Moreover, the training of an *RBF* network can be faster compared to the *MF* architecture when a not supervised training algorithm is employed for the hidden layer of the *RBF*. For these reasons, the *RBF* network architecture has to be seriously considered to solve classification problems.

In this network topology, the basic elements are grouped into three layers:

- The input layer.
- The hidden layer.
- The output later.

The transfer functions used in *RBF* networks are totally different to the traditional functions we have previously defined. A gaussian transfer function is applied to the neurons of the hidden layer, also called *Clusters* or *Gaussian Units* (GU). In the input and output layers, the identity transfer function is applied to the neurons. Moreover, the interconnections between the input and hidden layer are not weighted.

In both architectures, *MF* and *RBF*, there is an *input layer* which, sometimes, is omitted in the specifications. However, it is hard to believe that a neuron has a vectorial input, as the hidden units in *RBF*, according to biology, for this reason we have included the input layer in *RBF* networks.

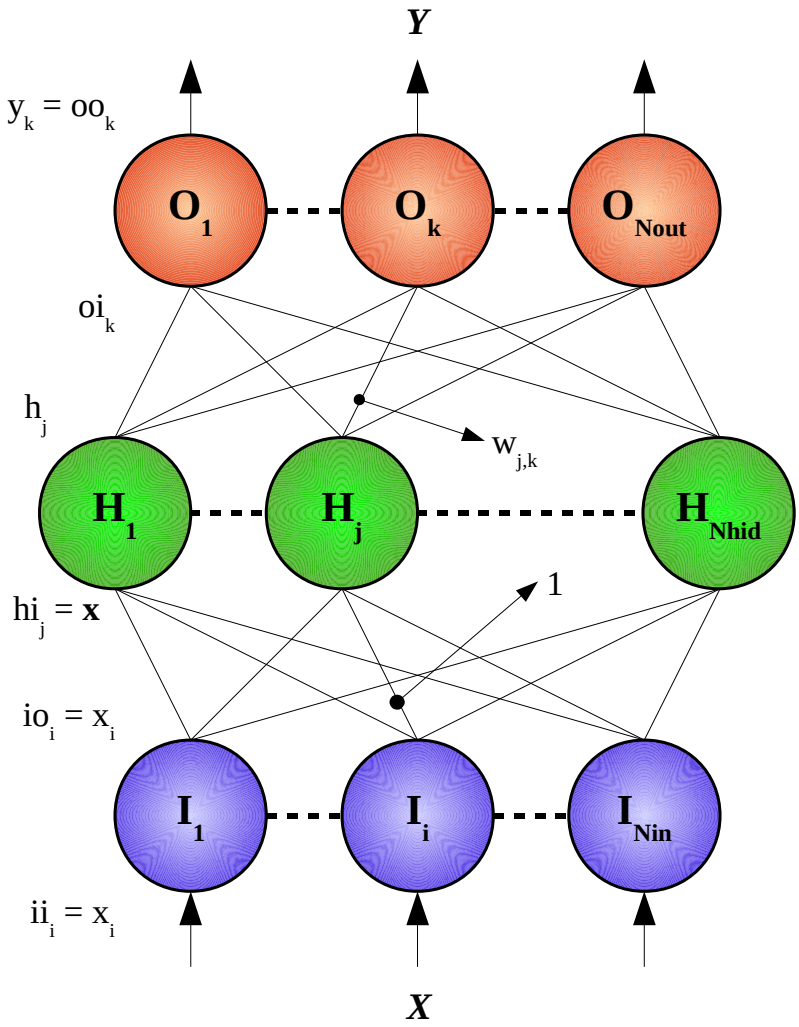


Figure 2.8: Radial Basis Functions Network Structure

The output of a *RBF* network is determined by the following equation:

$$\hat{y}_i = w_i^T \cdot h = \sum_{j=1}^{N_{clusters}} w_{i,j} \cdot h_j \tag{2.16}$$

Where h_i is the output of the i-th cluster (unit of the hidden layer).

In [29, 30] it is used a sensitivity analysis to show that the traditional *Gaussian Unit*, called *exponential generator function* in the papers, of the *RBF* network has low sensitivity for gradient descent training for a wide range of values of the widths, this parameter should be tuned carefully. As an alternative, two different transfer functions were proposed, called in the papers *lineal generator function* and *cosine generator function*. For this reason, we describe the following three equations that can be employed to calculate the output of the clusters.

- The *Exponential* generator transfer function:

$$h_i = \exp\left(-\frac{\|x - v_i\|^2}{\sigma_i^2}\right) \tag{2.17}$$

- The *Linear* generator transfer function:

$$h_i = \left(\frac{1}{\|x - v_i\|^2 + \gamma^2}\right)^{\frac{1}{m-1}} \tag{2.18}$$

- The *Cosine* generator transfer function:

$$h_i = \frac{a_i}{\sqrt{\|x - v_i\|^2 + a_i^2}} \tag{2.19}$$

Where:

- v is the center of the radial function.
- In the *Exponential generator function*, σ is the width of the gaussian function. This value is obtained by trial an error.
- In the linear generator function, γ is a parameter that has to be empirically set by trial an error and the configuration $m = 3$ has been determined in some papers as optimal.
- In the cosine generator function, a is a trainable parameter that has to be adjusted during the learning procedure.

2.3.5 Training a RBF network

The learning process of a *RBF* network is not as simple as in the case of *MF* networks. The training procedure is more complex because the weights, the centers and the widths of the gaussian functions have to be adjusted. There are two main methodologies to train a *RBF* network.

The first methodology consists in dividing the training procedure into two different steps. In the first step, the value of the centers and widths is determined by applying a common non-supervised algorithm. In the second step, the centers and widths are kept unchanged and the weights w are modified by a supervised training algorithm. The learning process is quicker when the training is divided into these two steps, but the performance of the final network highly depends on the success of determining the center and widths of the gaussian functions. Some typical two-stepped algorithms can be found in [31, 32, 33, 34].

The second methodology consists of applying a global training algorithm in order to adjust all the trainable parameters. This methodology is similar to *Backpropagation* so it has the same drawbacks, for instance, high computational cost and low training speed. However, some authors have successfully applied it [29, 35, 36, 30].

2.3.6 Training RBF with exponential generator units

After some experiments performed, the gaussian function (*Exponential generator Function*) was chosen to be employed in order to generate *RBF* networks [37]. The low sensitivity for gradient descent training in the widths of the gaussians described in [29, 30] was omitted by setting the value of the widths by a trial and error procedure. And according to our experiments with this last procedure, the performance of the three generator functions was similar and superior to the training in two steps (non-supervided, supervised) [37, 38]. So we finally decided to employ the usual gaussian transfer functions in our experiments trained by gradient descent.

Algorithm 2.2 RBF Network Training$\{T, V\}$

```
Set initial weights randomly
for e = 1 to epochs do
  for x = 1 to N_patterns do
    Adjust the value of the centers v and weights w
  end for
  Calculate MSE over validation set V
  Save epoch weights and calculated MSE
end for
Select epoch with lowest validation MSE
Assign best epoch configuration to the network
```

The equation to adjust the value of the weights of the output layer is:

$$\Delta w_p = \eta \cdot \sum_{k=1}^{N_{clusters}} \varepsilon_{p,k}^0 \cdot h_k \tag{2.20}$$

Where η is the adaptation step and ε^0 is the difference between the output of the network and the desired output and h_k is the output of the gaussian units. The equation applied to adapt the centers of the gaussian units is the following one:

$$\Delta v_q = \eta \cdot \sum_{k=1}^{N_{clusters}} \varepsilon_{p,k}^h \cdot (x_k - v_q) \tag{2.21}$$

Being ε^h determined by equations 2.22 and 2.23.

$$\varepsilon_{p,k}^h = \alpha_{q,k} \cdot \sum_{i=1}^{N_{classes}} \varepsilon_{i,k}^0 \cdot w_{iq} \tag{2.22}$$

$$\alpha_{q,k} = \frac{2}{\sigma^2} \cdot \exp\left(-\frac{\|x_k - v_q\|^2}{\sigma^2}\right) \tag{2.23}$$

Finally, the width of the gaussian is set to an equal value for all clusters and it is obtained by trial and error.

2.4 Comparing the network architectures studied

2.4.1 Experimental setup

In this first comparative study, the two included network architectures (*MF* and *RBF*) and *K-Nearest Neighbors* (*KNN*) [39] are tested and their accuracy on some classification problems is calculated. The main characteristics of this research are:

- Nineteen datasets from the *UCI Repository* are used.
- Two network architectures (*MF* and *RBF*) and *KNN* are compared.
- The training parameters of the networks are optimized by a trial and error procedure using the validation set.
- The experiments are repeated ten times with each database with different partitions of data in training, validation and test sets to get:
  - The mean value of performance.
  - The error rate by standard error theory.

The description of all the datasets used in the experiments can be found in appendix A. The training parameters for *MF* and *RBF* are in appendixes B.3 and B.4.

2.4.2 Results

In this subsection, the performance of the *MF* and *RBF* networks on each database is shown in table 2.1.

Table 2.1: Performance of the single network

Database	MF-Net	RBF-Net
aritm	75.6 ± 0.7	75.2 ± 0.6
bala	87.6 ± 0.6	89.4 ± 0.7
band	72.4 ± 1.0	72.4 ± 1.5
bupa	58.3 ± 0.6	71.9 ± 1.2
cred	85.6 ± 0.5	87.1 ± 0.5
derma	96.7 ± 0.4	96.8 ± 0.4
ecoli	84.4 ± 0.7	87.9 ± 0.9
flare	82.1 ± 0.3	81.5 ± 0.6
glas	78.5 ± 0.9	93 ± 1
hear	82.0 ± 0.9	83.4 ± 1.6
img	96.3 ± 0.2	97 ± 0.3
ionos	87.9 ± 0.7	90.6 ± 1
mok1	74.3 ± 1.1	99.6 ± 0.2
mok2	65.9 ± 0.5	90.8 ± 1
pima	76.7 ± 0.6	77.1 ± 0.8
survi	74.2 ± 0.8	76.4 ± 1.6
vote	95.0 ± 0.4	96.1 ± 0.6
vowel	83.4 ± 0.6	97.3 ± 0.3
wdbc	97.4 ± 0.3	97.1 ± 0.2

The performance shown in the previous table is calculated as the mean performance of a single network after repeating the experiments ten times as described above. The performance is the percentage of correctly classified patterns on the test set, TST, and the error rate is given by equation 2.24.

$$Error = \frac{std(Performance)}{\sqrt{n}} \tag{2.24}$$

Where *std* is the standard deviation of the performance after repeating the experiments n times, in the experiments performed n is set to 10.

At first sight, it can be observed that, in general, the *RBF* network provides similar or better results than the *MF* network. Moreover, there are some particular cases in which the performance of the *RBF* network is much better than the performance of the *MF* network. For instance, the *RBF* network is much better than the *MF* networks in datasets *bupa*, *glas* and two *mok* problems.

Furthermore, the performance of the *K-Nearest Neighbor* classifier has also been tested on the datasets previously used. *KNN* is an instance-based classifier in which classification is based on some distance or similarity function such as the *Euclidean distance*. Two instances far apart are less likely to belong to the same class than two closely situated instances. The results of this classifier are shown in table 2.2. Four different values for k have been considered.

Table 2.2: Performance of K-Nearest Neighbors

Database	$k = 1$	$k = 3$	$k = 5$	$k = 9$
aritm	61.8 ± 1.6	63.0 ± 1.3	61.7 ± 0.9	62.6 ± 1.0
bala	79.0 ± 1.1	81.8 ± 0.9	84.9 ± 0.8	87.8 ± 1.0
band	65.1 ± 1.6	70.7 ± 1.2	71 ± 2	73.1 ± 0.8
bupa	63.7 ± 1.6	61 ± 2	61.6 ± 1.8	62.4 ± 1.7
cred	83.4 ± 0.9	86.8 ± 0.6	86.3 ± 0.7	87.1 ± 0.5
derma	94.9 ± 0.8	96.2 ± 0.5	96.3 ± 0.7	95.9 ± 0.7
ecoli	81.9 ± 1.3	86.0 ± 0.8	85.4 ± 0.8	85.6 ± 1.3
flare	73.2 ± 1.0	78.7 ± 1.3	80.2 ± 0.9	80.6 ± 1.0
glas	90.2 ± 1.3	86.4 ± 1.6	86.0 ± 1.3	83.4 ± 1.4
hear	74.9 ± 1.5	78.6 ± 1.5	79.8 ± 1.7	80.8 ± 1.6
img	96.2 ± 0.2	94.9 ± 0.3	94.6 ± 0.3	94.2 ± 0.2
ionos	87.0 ± 1.1	85.6 ± 1.1	84.7 ± 1.0	83.9 ± 1.4
mok1	75.0 ± 1.3	81.6 ± 1.4	75.8 ± 1.6	72.0 ± 1.8
mok2	77.5 ± 1.6	80.0 ± 1.3	79.9 ± 1.4	75.8 ± 0.9
pima	70.5 ± 1.2	71.7 ± 0.9	72.5 ± 1.1	72.6 ± 0.8
survi	63.6 ± 1.2	68.0 ± 1.0	69.7 ± 1.2	73.1 ± 1.0
vote	93.6 ± 0.8	93.6 ± 0.5	93.5 ± 0.5	93.3 ± 0.4
vowel	96.6 ± 0.5	91.4 ± 1.0	85.1 ± 0.8	73.0 ± 1.1
wdbc	94.8 ± 0.5	96.6 ± 0.6	96.3 ± 0.5	96.6 ± 0.5

According to the previous table, the *KNN* classifiers provide similar or worse results to *MF* networks in a few cases and they perform worse than *RBF* networks in general. The main advantage of these classifiers is that they do not need to be trained. Although research on statistical or decision-based classifiers as *KNN* is also interesting, this thesis will be only focused on neural networks.

2.4.3 Analysis of the results

As it can be observed in table 2.1, the *RBF* network provides better results than the *MF* network in general. But we will consider both architectures for further experiments.

However, there are other characteristics that should be seriously considered such as the resources required to set the optimal training parameters and to train the networks. These resources are related to the main memory and computational time required to set the parameters and to train the networks.

The following table, table 2.3, shows the computational cost, the training time of the networks with optimal parameters, and the memory required for both architectures. These results have been obtained by running a single program on an *Intel Core i7* processor. In this script, the different networks have been trained in serial, never more than a network has been trained simultaneously.

Table 2.3: Training resources required

	MF		**RBF**	
dataset	**time (sec)**	**mem (MB)**	**time (sec)**	**mem (MB)**
aritm	35.8	50.5	510.6	893
bala	45.7	6.6	78	20.3
band	28.6	38.8	108.4	131.5
bupa	37.6	7.1	53.2	20.7
cred	77.1	18.7	499.6	164.4
derma	4.8	1.3	446.2	518.9
ecoli	43	7.3	4353	3372.8
flare	137.9	11.9	2214.4	387.2
glas	9.6	1.9	201	282.2
hear	16.6	1.5	120.1	60.8
img	55.2	4.6	17986.1	2830
ionos	26.3	12	336	345.8
mok1	16	1.4	145.1	39
mok2	44.5	10.4	536.4	174.7
pima	101.4	12.8	282.6	72.4
survi	75.3	9.6	133.6	17.2
vote	18.1	0.5	24.7	3.7
vowel	59.8	11.5	1776.1	887.6
wdbc	30.8	6.5	1714.3	1077.4
total	865	215.6	31520.1	11300.4

According to the previous table, training an *MF* is, in general, faster than training an *RBF* network by gradient descent. In fact, the time required to train a network for all the datasets is only 15 minutes for *MF* networks whereas the same experiment on *RBF* networks takes more than 8 hours. Moreover, the memory required for training and definitively storing an evolution of the simulation is much higher for the case of *RBF* networks. There are some datasets in which the memory required to train the *RBF* network reach a size higher than one gigabyte whereas 50 megabytes is the highest value of required memory for *MF* networks.

It is important to mention that the resources showed are related to training of a single network with optimal parameters with a *modern computer*. However, tunning the optimal parameters requires training several networks with different values of all the training parameters for a large number of iterations. In other words, setting the optimal parameters for both architectures is a long term task even if the experiments are distributed in all the computers of our laboratory.

2.5 Conclusions

The application of neural networks in order to solve classification tasks has been introduced in this chapter. In fact, two network architectures, *MF* and *RBF*, have been described and some basic experiments with them have been performed.

The results of these basic experiments show that the classifiers based on *RBF* networks may be more accurate than the classifiers based on *MF* networks. However, the computational cost and resources required to set the optimal parameters and to train a single network is much lower for *MF* networks than for *RBF* networks.

Moreover, the statistical classifier *K-Nearest Neighbors* has also been implemented and compared to *MF* and *RBF* networks. This statistical classifier is important because it provides good results, it does not depend on any training parameter and it does not require a training procedure. However, classifying a pattern is a little bit slow since the pattern has to be compared to all the patterns of the training set. In this thesis we will be focused on neural networks.

Finally, the most appropriate network architecture should be selected depending on the dataset and its application. In further experiments, both network architectures will be initially considered and applied. However, the majority of experiments will be performed with the *MF* network due to its versatility and suitability on ensembles of neural networks.

Chapter 3

Designing Ensembles and other MCS

3.1 Introduction

In this chapter, ensembles of neural networks are introduced. This chapter is organized as follows. Firstly, some theoretical concepts related to the *ensemble model* and other *Multiple Classifier Systems* will be introduced. Secondly, *Simple Ensemble* will be described. Finally, some basic experiments are performed.

3.2 Preliminar definitions

Here, some basic concepts related to the research performed will be defined.

3.2.1 Multiple Classifier Systems

A *Multiple Classifier System*, also known as *MCS*, is a complex classification system which is composed by simpler classifiers. The final output or prediction is based on the information provided by the base classifiers.

The main idea is that a classification problem may be more easily solved by a set of 'simple' classifiers than with a unique and more complex classifier. As Kittler et al. said [40, 41]:

> "the sets of patterns misclassified by the different classifiers would not necessarily overlap"

This sentence means that you could design a good classifier by creating an appropriate model which properly combines the information of different simple classifiers.

Reviewing the bibliography we can see that there are three major approaches or models when we discuss about *MCS*, which are:

- Ensembles of Classifers.
- Modular Systems.
- Multiple Level Classifiers.

Although these three approaches are important, the most important model is the *Ensemble of Classifiers* that will be applied to Neural Networks in this thesis.

3.2.2 Ensembles of Neural Networks

The Ensemble model consists in training a set of neural networks with different weight initialization or properties in the training process with the purpose of solving a problem. Furthermore, the same network architecture is used in all the networks of the ensemble and all the networks completely solve the whole problem. Finally, once all the networks are trained, a suitable combiner is applied to calculate the final output of the ensemble.

The process of designing an ensemble of neural networks consists of two main steps:

- The ensemble development.
- The determination of a suitable combiner.

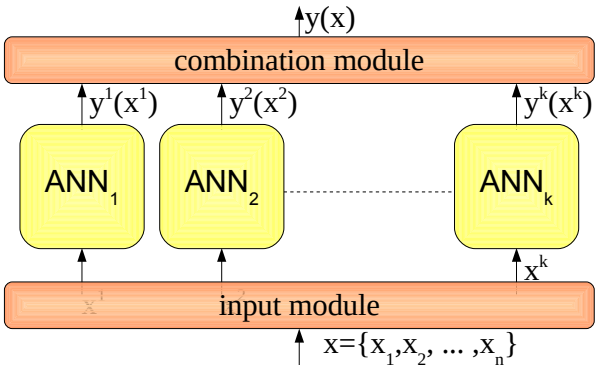


Figure 3.1: Ensemble Basic Diagram

In the first step, the development of the ensemble, the networks of the ensemble are trained according to the specifications of the design alternative. In this case, the diversity of the ensemble is generated by the method applied to build the ensemble. Chapters 4 and 5 are focused on the methods to build ensembles of neural networks.

The second step, the determination of the suitable combiner, consists in selecting the most accurate combiner for the generated ensemble. In chapter 6, procedures to combine ensembles will be described and analyzed. Further, the *Output average* described in equation 3.1 is supposed to be the first option to combine the networks of the ensemble when the building method does not require a specific combiner.

$$y^{ensemble}(x) = \frac{1}{N_{networks}} \sum_{net=1}^{N_{networks}} y^{net}(x) \tag{3.1}$$

Output average is the simplest combiner and it averages the individual classifiers outputs, y^{net}, across the different classifiers, $N_{networks}$. The notation of the variables shown in the previous chapter is modified by adding a superindex referring to the network of the ensemble. The number of networks of the ensemble is given by $N_{networks}$ or N_{nets}.

3.2.3 Diversity and Neural Networks

The On-line version of the Cambridge Dictionary defines diversity as:

Diversity *Noun.* When many different types of things or people are included in something.

In the *MCS* field, diversity is associated with the grade of dependence among the base classifiers which form the whole classification system. It is clear that a set of absolutely equal networks will commit the same errors in the problem and will not improve the performance of a single network. Measuring the dependence of the networks is not an easy task and some equations to measure the diversity of a *MCS* can be found in the bibliography [42].

In this thesis, some different ensemble methods will be analyzed in which diversity is applied in some different ways.

3.3 Why Ensembles and MCS classify better?

As mentioned in chapter 2, the *MF* network and the *RBF* network could approximate *any* function with a specified precision. Unfortunately, the 'optimal' classifier is nearly impossible to be built in the real world. Some causes of this problem could be:

- The data set is finite, there might not be enough data.
- Some important features of the problem might be omitted.
- Noisy or imprecise data might be introduced.
- Optimal training is NP-Hard for neural networks [43, 44].

Although some complex training algorithms could have been proposed to overcome this drawback, in [45, 46] Ho expressed that the best solution may consist in combining some simple classifiers in an effective way.

> "Instead of looking for the best set of features and the best classifier, now we look for the best set of classifiers and the best combination methods. One can imagine that very soon we will be looking for the best set of combination methods and then the best way to use them all. If we do not take the change to review the fundamental problems arising from this challenge, we are bound to be driven into such an infinite recurrence, dragging along more and more complicated combination schemes and theories and gradually loosing sight of the original problem."

Some authors have defended the idea of combining some classifiers, neural networks in our case, by giving theoretical demonstrations or by suggesting the main reasons why a multiple classifier system might perform better than a single one [40, 41, 47, 48, 49].

In this section the *Tumer and Ghost framework* and the *Diettrich reasonings* will be introduced in order to justify our research on ensembles of artificial neural networks for classification tasks.

3.3.1 Tumer and Ghosh framework

Tumer and Ghosh [47, 48] provided a theoretical framework for analyzing the combiner *output average* when the classifier outputs are an estimation of the posterior probabilities of each class. This framework is shown in figure 3.2.

The output of a classifier for the ith class $f_i(x)$ is given by $f_i(x) = p_i(x) + \eta_i(x)$ where $p_i(x)$ is the posterior probability of class i for a given input x, and $\eta_i(x)$ is the associated error. The *bayesian* decision would be given to class i for the pattern x if $p_i(x) > p_k(x)\ \forall\ k \neq i$.

Unfortunately, classifiers produce an output f_i and not p_i so there is an added error E_{added} over the bayesian error $E_{bayesian}$. This composed error is given by:

$$E^i = E^i_{bayesian} + E^i_{added} \tag{3.2}$$

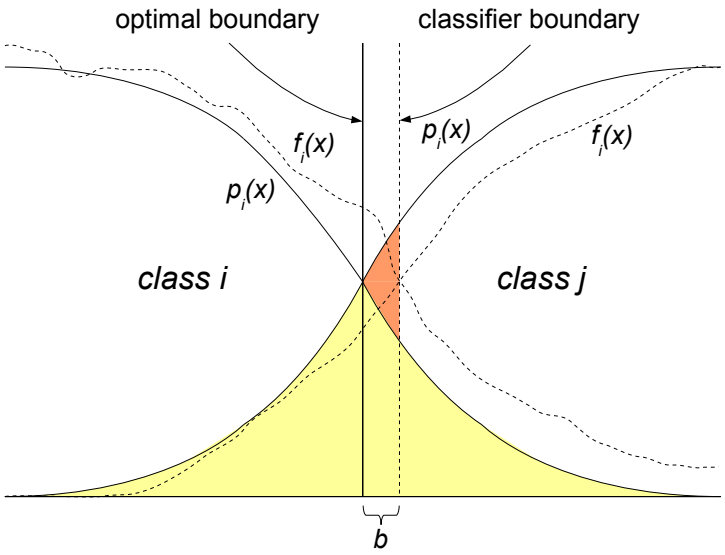


Figure 3.2: Tumer and Ghosh's Framework

In the previous figure, the yellow shaded area corresponds to the *bayesian error* whereas the orange shaded area corresponds to the *expected added error*. As we can see, the bayesian error correspond to the posterior probabilities overlap and it can not be reduced. Moreover, the expected added error is the error of a classifier in addition to the bayesian error.

Tumer and Ghosh also suggested that the expected added error of an ensemble $E_{added}^{ensemble}$ will decrease with an increase in the number of the different members ($N_{networks}$) of the ensemble as shown in the following equation:

$$E_{added}^{ensemble} = \frac{1}{N_{networks}} \cdot E_{added} \tag{3.3}$$

Unfortunately, this equation can be only applied if the members of the ensemble are totally independent. If the classifiers are correlated, the added error of an ensemble is given by:

$$E_{added}^{ensemble} = \frac{1 + \delta \cdot (N_{networks} - 1)}{N_{networks}} \cdot E_{added} \tag{3.4}$$

Where δ denotes the correlation coefficient which ranges from 0 to 1. This value denotes the degree of dependence of the networks. There are some methods and measurements to calculate this coefficient as we can see in [42].

From this framework, we can conclude that the ensemble approach is quite interesting because the error in classification decreases as new independent networks are added to the ensemble.

3.3.2 Diettrich reasonings

Dietterich identified in [49] the three major problems of a classification system based on a single classifier and how an ensemble of classifiers could overcome them. In order to define these problems, he assumed that all the possible classifiers where defined into a classifiers space H.

The statistical problem: There can be classifiers with very different final configuration and with the same accuracy. Combining predictions from these classifiers produces a smoothing in the output space and reduces the risk of choosing a single poor classifier, C_4 in figure 3.3. The main idea is that a classification system based only on one classifier has the risk of being composed by the worst possible classifier. If the classification system is composed by some classifiers (a typical *MCS)* the probability of having the behavior of the worst classifier is much lower.

The computational problem: As mentioned above, the 'optimal' classifier if nearly impossible to be built. It is known that an optimal training algorithm is *NP-hard* for the case of neural networks [43, 44]. When a training algorithm based on the gradient descent is applied, the final classifiers are susceptible to be in a local optima. If the networks of an ensemble are properly combined, the ensemble may perform better than any of the single networks without applying a complexer training algorithm

The representational problem: Although it has been suggested that the *MF* and *RBF* networks are universal and any decision boundary can be represented by applying them, the classifiers space H is limited because the training set is finite and the 'optimal' classifier, C in figure 3.3, which can exactly represents the real decision boundary could be outside H.

Figure 3.3 shows the graphical illustrations of the three arguments proposed by Dietterich. The appearance of the figure has been adapted in this thesis.

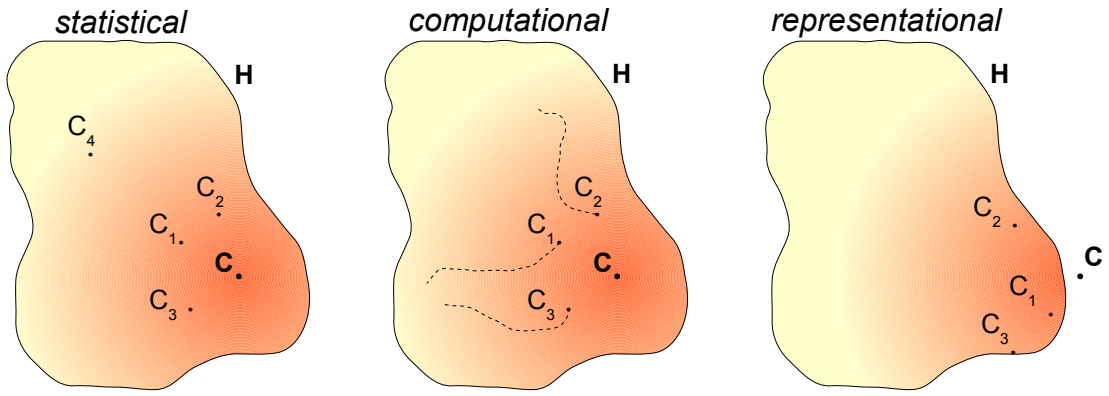


Figure 3.3: Diettrich arguments

3.4 How ensembles are grouped?

In this section we will classify the ensemble methods according to how diversity is generated. First of all, the different ways of generating diversity will be reviewed. Finally, the ensemble methods will be divided into two different approaches.

3.4.1 Sources of diversity

As we have previously mentioned, the classifiers of an ensemble are most useful when they make independent errors. Furthermore, some authors defend that the error of the *MCS* decreases as the ensemble diversity increases.

There are some sources to create different neural networks with a increase in the diversity of the system. In figure 3.4, we see some of them:

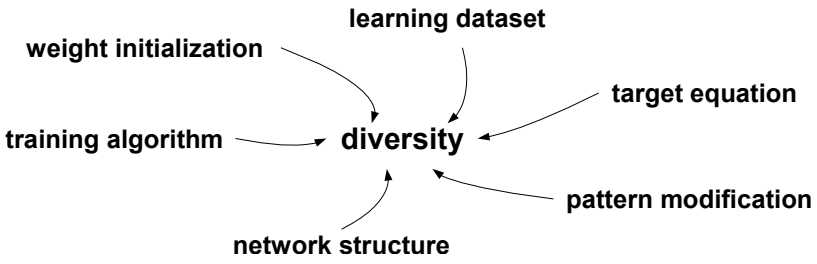


Figure 3.4: Sources of Diversity in Neural Networks

Weight initialization: If the networks of an ensemble have different starting points, initial weight configurations, then the different networks can converge to different local minima, having independent errors and the ensemble may provide a better classification than any of the individual classifiers.

Learning dataset: If the learning datasets are different in each network, then the space of possible classifiers is also different. The main problem of creating different learning datasets is that the number of patterns in a particular dataset is usually low so the different learning sets are similar.

Target equation and training algorithm: Traditionally, the target equation applied with *Backpropagation* is the *Mean Square Error* but other target equations that include information of the other networks in the ensemble can be also applied.

Pattern modification: The modification of the pattern information, input attributes or class label, can be useful in some cases.

Network structure: The space of possible classifiers also depends on the internal structure of the network. Using networks with different number of processing units can be interesting. Moreover, the use of different values in the training parameters can be useful.

Finally, we think that if all these sources of diversity are properly mixed, good classification systems can be designed.

3.4.2 Final classification

The alternatives to design ensembles of neural networks have been divided into two major groups depending on how diversity is generated. These groups are:

- Methods based on the modification of the training algorithm.
- Methods based on the modification of the learning set.

The first group based on the modification of the training algorithm also includes the methodologies based on alterations on the target equation or network structure. Whereas the second group based on the modification of the learning set includes the methodologies in which the training or validation sets or the patterns are altered.

Finally, the first category will be described and analyzed in chapter 4 whereas the second category will be reviewed in chapter 5. Before describing these alternatives to generate ensembles, *Simple Ensemble* will be studied in this chapter.

3.5 Simple Ensemble

As has been previously described, the learning process of an artificial neural network is based on minimizing a target function. A simple procedure to increase the diversity of an ensemble consists in using several neural networks with different initial values of the trainable parameters. Once the initial configuration is randomly set, the network can be trained as a single network. Finally, a combiner as the *Output average* described in equation 3.1 can be applied to combine the outputs. With this ensemble method, known as *Simple Ensemble*, the networks of the ensemble converge into different final configurations. The description of *Simple Ensemble* is shown in algorithm 3.1.

Algorithm 3.1 Simple Ensemble $\{T, V, N_{networks}\}$

```
Generate N_networks different seed values: seed_i
for i = 1 to N_networks do
    Random Generator Seed = seed_i
    ANN Training { T , V , seed_i }
end for
Save Ensemble Configuration
```

Although this is a simple method, it provides a considerable increase in performance with respect to a single network. In figure 3.5, we can see a graphical example of the relation between the number of networks in a ensemble and the accuracy of the ensemble for the case of *MF* ensembles and dataset *vowel*. In this figure, we can notice that the improvement in performance increases as new networks are added to the ensemble. Unfortunately, this improvement tends to be softer and softer as the number of networks increases, this behavior can be justified by equation 3.4 of Tumer and Ghosh. In this case, we have that there is only one source of diversity and it is limited. And the networks of the ensemble may tend to fall into a few local optima. For these reasons, there is a limit in which adding new networks to an

ensemble does not increase or increases slightly the performance of the classification system.

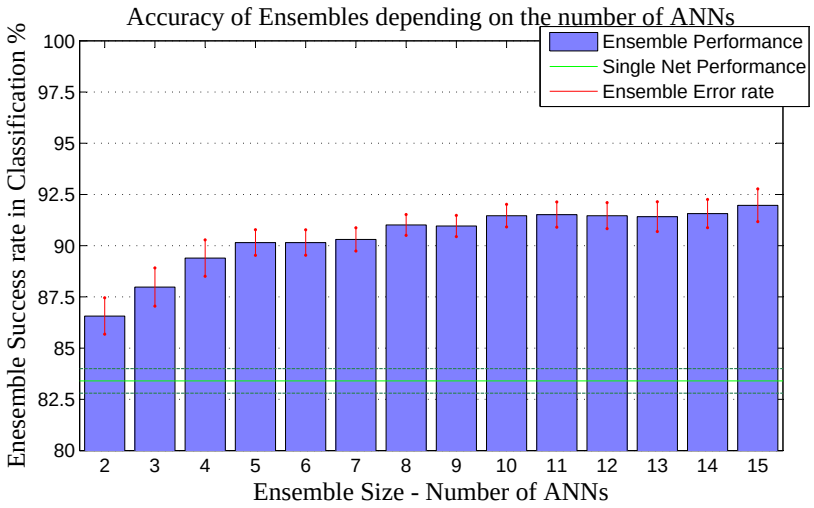


Figure 3.5: Example of the increase in performance. Ensembles of *MF* networks and dataset *vowel*.

As it has been mentioned, the networks tend to fall into a few local optima due to the characteristics of the ensemble method and to the problems derived from gradient descent methods. Figure 3.6(a) shows graphically the theoretical simple ensemble in which all the networks converge into different configurations. In the worst case, figure 3.6(b), all the networks converge to the same local optima so the errors are correlated and the networks are not independent. Finally, it also shows how the networks converge in a typical simple ensemble, figure 3.6(c).

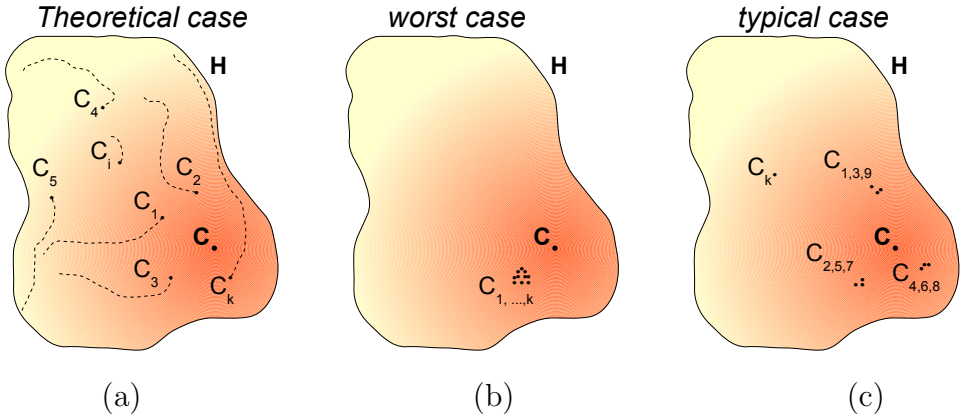


Figure 3.6: Representation of the Theoretical, Worst and Typical Simple Ensemble

Finally, the use of different initial values of the trainable parameters is assumed to be used in all the methods to build ensembles of neural networks.

3.6 First experiments on ensembles

3.6.1 Experimental setup

In this section, *Simple Ensemble* is tested with the two reviewed network architectures, *MF* and *RBF*. The purpose of these experiments is to test in which architectures the use of ensembles improves the performance with respect to a single network. The main characteristics of this research are:

- Nineteen datasets from the *UCI Repository*.
- Two network architectures employed: *MF* and *RBF*
- Optimized training parameters.
- *Simple Ensemble* applied to generate the ensembles.
- One combiner applied: Output average.
- Experiments repeated ten times with different partitions in training, validation and test sets:
  - Mean value of performance.
  - Error rate by standard error.
- Three general measurements applied to the comparison.
  - Mean *Increase of Performance*.
  - Mean *Percentage of Error Reduction*.
  - *Paired Student's t-test*.

The description of the nineteen datasets used in the experiments can be found in appendix A. The optimized training parameters for the *MF* and *RBF* networks are in appendixes B.3 and B.4. Finally, the general measurements applied to test the behavior of *Simple Ensemble* are described in section 3.6.3.

3.6.2 Raw results

In this subsection, the performance of *Simple Ensemble*, using a *MF* and *RBF* network, on each database is shown along with its error rate in tables 3.1 and 3.2 respectively. We consider as performance the percentage of correctly classified samples in the test set and the error in the performance is given by equation 2.24. Moreover, the results obtained with a single network are also included.

At first sight, it can be observed that the analysis of the raw results is a tedious procedure because the amount of data to process is too high and it is difficult to obtain general conclusions. This data should be simplified without loosing accuracy about the general behavior of the ensembles. This is the main reason why the *Increase of Performance*, the *Percentage of Error Reduction* and the *Paired Student's t-test* are applied in this thesis. These measurements will be described in subsection 3.6.3.1.

Table 3.1: Performance - *Simple Ensemble* of *MF* networks

Database	1-net	3-net	9-net	20-net	40-net
aritm	75.6 ± 0.7	73.4 ± 1	73.8 ± 1.1	73.8 ± 1.1	73.8 ± 1.1
bala	87.6 ± 0.6	96 ± 0.5	95.8 ± 0.5	95.8 ± 0.6	95.9 ± 0.5
band	72.4 ± 1.0	73.5 ± 1.2	72.9 ± 1.5	73.8 ± 1.3	73.8 ± 1.3
bupa	58.3 ± 0.6	72.3 ± 1.2	72.4 ± 1.1	72.3 ± 1.1	72.7 ± 1.1
cred	85.6 ± 0.5	86.5 ± 0.7	86.4 ± 0.7	86.6 ± 0.7	86.5 ± 0.7
derma	96.7 ± 0.4	97.2 ± 0.7	97.5 ± 0.7	97.3 ± 0.7	97.6 ± 0.7
ecoli	84.4 ± 0.7	86.6 ± 0.8	86.9 ± 0.8	86.9 ± 0.8	86.9 ± 0.7
flare	82.1 ± 0.3	81.8 ± 0.5	81.6 ± 0.4	81.5 ± 0.5	81.6 ± 0.5
glas	78.5 ± 0.9	94 ± 0.8	94 ± 0.7	94 ± 0.7	94.2 ± 0.6
hear	82.0 ± 0.9	82.9 ± 1.5	83.1 ± 1.5	83.1 ± 1.5	82.9 ± 1.5
img	96.3 ± 0.2	96.5 ± 0.2	96.7 ± 0.3	96.7 ± 0.2	96.8 ± 0.2
ionos	87.9 ± 0.7	91.1 ± 1.1	90.3 ± 1.1	90.4 ± 1	90.3 ± 1
mok1	74.3 ± 1.1	98.3 ± 0.9	98.8 ± 0.8	98.3 ± 0.9	98.3 ± 0.9
mok2	65.9 ± 0.5	88 ± 2	90.8 ± 1.8	91.1 ± 1.1	91.1 ± 1.2
pima	76.7 ± 0.6	75.9 ± 1.2	75.9 ± 1.2	75.9 ± 1.2	75.9 ± 1.2
survi	74.2 ± 0.8	74.3 ± 1.3	74.2 ± 1.3	74.3 ± 1.3	74.3 ± 1.3
vote	95.0 ± 0.4	95.6 ± 0.5	95.6 ± 0.5	95.6 ± 0.5	95.6 ± 0.5
vowel	83.4 ± 0.6	88 ± 0.9	91 ± 0.5	91.4 ± 0.8	92.2 ± 0.7
wdbc	97.4 ± 0.3	96.9 ± 0.5	96.9 ± 0.5	96.9 ± 0.5	96.9 ± 0.5

Table 3.2: Performance - *Simple Ensemble* of *RBF* networks

Database	1-net	3-net	9-net	20-net	40-net
aritm	75.2 ± 0.6	75.2 ± 1	75.4 ± 0.9	75.4 ± 0.9	75.3 ± 0.9
bala	89.4 ± 0.7	89.6 ± 0.7	89.7 ± 0.7	89.7 ± 0.7	89.7 ± 0.7
band	72.4 ± 1.5	72.9 ± 1.5	73.5 ± 1.6	74 ± 1.4	74.7 ± 1.4
bupa	71.9 ± 1.2	71.7 ± 1.3	71.9 ± 1.3	71.9 ± 1.4	72.1 ± 1.2
cred	87.1 ± 0.5	87 ± 0.5	87.2 ± 0.5	87.2 ± 0.6	87.2 ± 0.6
derma	96.8 ± 0.4	96.8 ± 0.5	97.2 ± 0.5	97.3 ± 0.5	97.2 ± 0.5
ecoli	87.9 ± 0.9	88.2 ± 0.9	88.2 ± 0.9	87.9 ± 1	88.1 ± 0.9
flare	81.5 ± 0.6	81.6 ± 0.6	81.6 ± 0.5	81.9 ± 0.5	81.7 ± 0.5
glas	93 ± 1	93.2 ± 1	93.4 ± 0.9	93.2 ± 1	93.2 ± 1
hear	83.4 ± 1.6	83.6 ± 1.7	82.7 ± 1.9	82.5 ± 1.8	83.2 ± 1.7
img	97 ± 0.3	96.9 ± 0.3	97 ± 0.3	97 ± 0.3	96.9 ± 0.3
ionos	90.6 ± 1	90.7 ± 1.1	90.6 ± 1.1	90.9 ± 1.1	90.7 ± 1.1
mok1	99.6 ± 0.2	99.6 ± 0.3	99.8 ± 0.3	99.8 ± 0.3	99.8 ± 0.3
mok2	90.8 ± 1	90.3 ± 1.3	91.4 ± 1.3	91.4 ± 1.2	91.5 ± 1.2
pima	77.1 ± 0.8	77.3 ± 0.9	77.3 ± 0.9	77.4 ± 0.9	77.4 ± 0.9
survi	76.4 ± 1.6	75.6 ± 1.5	75.6 ± 1.5	75.6 ± 1.5	75.7 ± 1.5
vote	96.1 ± 0.6	96.3 ± 0.6	96.3 ± 0.7	95.9 ± 0.6	96 ± 0.6
vowel	97.3 ± 0.3	97.2 ± 0.4	97.3 ± 0.3	97.3 ± 0.4	97.2 ± 0.3
wdbc	97.1 ± 0.2	97.2 ± 0.3	97.3 ± 0.3	97.1 ± 0.4	97.1 ± 0.3

3.6.3 General measurements

3.6.3.1 Measurements based on the performance of a single network

In the current and future experiments, the *Increase of Performance*, henceforth IoP, and the *Percentage of Error Reduction*, henceforth PER, of the results with respect to a single network have been calculated in order to perform an exhaustive comparison. The IoP value is an absolute measurement whereas the PER value is a relative measurement. Both of them can be used to compare easily ensemble methods. Moreover, they are based on the mean performance of the ensemble after repeating the experiments ten times. Finally, figure 3.7 shows a graphical representation of the general measurements IoP and PER.

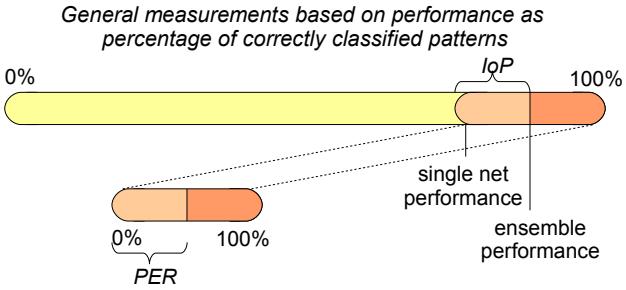


Figure 3.7: Graphical representation of IoP and PER

The IoP is described in equation 3.5 and its value denotes the increase of performance of the ensemble with respect to the performance of a single network.

$$IoP = Performance_{Ensemble} - Performance_{SingleNet} \tag{3.5}$$

Where $Performance$ in equation 3.5, and in the following equations, is the percentage of correctly classified patterns in the test set of the single network or the ensemble.

The PER value, described in equation 3.6, ranges from 0%, where there is no improvement by the use of an ensemble method with respect to a single network, to 100%, where the error have been totally reduced and the performance of the ensemble is 100%. The PER value has been often used to compare ensemble methods and other algorithms [50, 51, 52, 53, 54].

$$PER = 100 \cdot \frac{Error_{SingleNet} - Error_{Ensemble}}{Error_{SingleNet}} \tag{3.6}$$

where:

$$Error = 100 - Performance \tag{3.7}$$

A negative value of IoP or PER means that a single network performs better than the ensemble.

Finally, the mean IoP and the mean PER across all databases is calculated to obtain a global measurement used to compare the ensemble methods. All the results shown in this thesis are based in these measurements.

3.6.3.2 The Paired T-Test

The mean IoP and the mean PER are the two measurements used to rank the ensemble methods in this thesis. Unfortunately, the ensembles can not be statistically compared with them so the *Paired T-Test* will be also applied in some cases.

This test determines whether two alternatives or procedures differ from each other in a significant way under the assumptions that the paired differences are independent and identically normally distributed. The measurement value, t, is given by equation 3.8.

$$t = (\overline{X} - \overline{Y}) \sqrt{\frac{n \cdot (n-1)}{\sum_{i=1}^{n} (\widehat{X}_i - \widehat{Y}_i)^2}} \tag{3.8}$$

where:

$$\widehat{X}_i = X_i - \overline{X}_i \qquad \widehat{Y}_i = Y_i - \overline{Y}_i \tag{3.9}$$

The t-value represents, in this case, the ratio between the increase of performance and the variability or dispersion of the performance of a method X with respect to another method Y. A positive t-value means that X performs better than Y whereas a negative t-value means the opposite. However, the t value is not enough to perform a statistical comparison so the table of Student's t-distribution and confidence intervals are applied to determine the significance level denoted by α. Commonly, two methods are statistically different if $\alpha \leq 0.05$.

The vectors X and Y contain the values of the performance of the classifiers for each experiment and dataset. In this thesis, the experiments are always repeated 10 times with different partitions of the datasets on 19 classification problems so these vectors have 190 elements.

Finally, the *t-test* is commonly used in machine learning in order to determine whether one learning algorithm is statistically better than another on a given task [55, 56, 57] and it has been implemented in the *Waikato Environment for Knowledge Analysis* (*WEKA*) suite along with learning algorithms and ensemble methods [58].

3.6.4 Analysis of the general results

Performing a deep analysis of the complete results generated is a complex task. Moreover, to have a general idea of the behavior of the ensembles and extracting conclusions is not trivial. For these reasons the discussion of the results has been divided into three analyses.

Firstly, the results of *Simple Ensemble* are compared to the results provided by the *Single Network* in order to determine if this basic ensemble improves the results of the single network. The *MF* and *RBF* networks are used to perform this first study.

Secondly, we want to test if the networks of the ensemble differ on the patterns correctly classified as Kittler said [40, 41]. This is a measurement of the diversity of the different networks in the ensemble.

Finally, the results obtained by *Simple Ensemble* of *MF* networks are compared to the results obtained by *Simple Ensemble* of *RBF* networks. Although both ensembles can be compared analyzing the raw results, the *T-Test* is applied to determine if the results are statistically different or not.

3.6.4.1 First analysis: Comparing *Simple Ensemble* to *Single Nets*

Comparing the performance provided by a single network to the performance of ensembles composed by 3, 9, 20 and 40 networks is not an easy task with the information provided by tables 3.1 and 3.2. The general measurements *mean IoP* and *mean PER* are calculated in order to have general information about the improvements done by the ensembles with respect to the single network.

Table 3.3 shows the general results related to the ensembles of *MF* and *RBF* networks trained with *Simple Ensemble*. These results can be easily derived from tables 3.1 and 3.2 by applying the equations of the general measurements shown in section 3.6.3.1. With this procedure, simplified information about the behavior of the ensembles is obtained.

Table 3.3: Simple Ensemble - General results

	Simple ensemble* of *MF			***Simple ensemble* of *RBF***	
Size	**mean** IoP	**mean** PER	**Size**	**mean** IoP	**mean** PER
3-net	4.97	21.84	**3-net**	0.02	0.15
9-net	5.27	23.65	**9-net**	0.14	3.71
20-net	5.34	23.73	**20-net**	0.14	3.06
40-net	5.43	24.64	**40-net**	0.23	3.29

In the previous table, the information of *Simple Ensemble* considering 4 different ensemble sizes and two network architectures is shown. In the case of *MF* networks, there is a considerable increase of performance with respect to a single *MF* network. For example, the PER values range from 22% to 25%.

In the case of *RBF* networks, unfortunately, the performance of the ensemble is close to the performance of the single *RBF* network (low general results). There is not a high improvement of performance neither in general nor in any database. Moreover, the *mean IoP* and the *mean PER* are not completely correlated, for instance the PER value for 40 networks is lower than for 9 networks whereas the IoP value for 40 networks is greater than the value for 9 networks. Maybe this is due to the similarity of the ensembles with respect to the single network, the ensemble provides slightly better, or worse, results than the *Single Network* and the mean IoP is close to 0. However a slight increase, or decrease, of performance has a specific PER depending on the classification problem so it may differ to IoP in this case.

For this reason, the statistical test was also applied to compare the results provided by a single RBF network to the results provided by the *Simple Ensemble* of *RBF* networks.

Table 3.4: *T-Test* of *Single-Network* versus *Simple Ensemble* for *RBF* networks

Size	t-value	α
3-net	-0.17	0.87
9-net	-1.24	0.22
20-net	-1.24	0.22
40-net	-2.00	0.05

According to the previous table, the results provided by the single *RBF* network can only be statistically improved by ensembles of 40 *RBF* networks. We need to train an ensemble of 40 *RBF* networks in order to obtain a clear statistically significant increase of performance of 0.23% with respect to a single *RBF* network. We think that training 40 *RBF* networks is not worth because the increase of performance is low and the computational cost is quite high.

We consider that the ensemble model fits better on the *MF* network because the mean IoP and PER with respect to the single network is considerably higher than for the case of *RBF* network.

3.6.4.2 Second analysis: Diversity on the classified patterns

The raw results previously shown (tables and 3.1 and 3.2) do not provide information about how individual patterns have been classified because it only shows the count of the correctly classified patterns. Moreover, a pattern that have been correctly classified by a network may not be correctly classified by the ensemble because the output provided by the ensemble corresponds to the simple average among the outputs of all the networks of the ensemble. Finally, the networks of the ensembles can provide similar performance without sharing the same set of correctly classified patterns.

Comparing one-by-one how the patterns are classified is a nearly impossible task because there is a high number of patterns in each dataset and the experiments have been repeated ten times. One way to determine if the networks of an ensemble differ on classifying patterns is applying the union of the correctly classified patterns among all the networks of the ensemble and then counting the number of patterns of this new "set". If this new count is considerably greater than the performance of the ensemble, it means that the networks of the ensemble do not correctly classify the same patterns.

In order to evaluate if the networks of the generated ensembles differ on classifying patterns, the union previously mentioned has been calculated for *Simple Ensemble* of *MF* and *RBF* networks. Table 3.5 shows the performance for these ensembles of 40 networks and the mean number of patterns which has been correctly classified by, at least, one network of the ensemble. This *measurement* has been denoted as *union* in the table.

Table 3.5: Number of patterns correctly classified at least by one network

	MF Net		RBF Net	
Database	**ensemble 40 net**	**union**	**ensemble 40 net**	**union**
aritm	73.8 ± 1.1	79 ± 2	75.3 ± 0.9	80.5 ± 1.1
bala	95.9 ± 0.5	99.1 ± 0.2	89.7 ± 0.7	91.4 ± 0.9
band	73.8 ± 1.3	80.9 ± 1.6	74.7 ± 1.4	87.5 ± 1.1
bupa	72.7 ± 1.1	82.6 ± 1.3	72.1 ± 1.2	76.9 ± 1.6
cred	86.5 ± 0.7	88.6 ± 0.9	87.2 ± 0.6	89.3 ± 0.5
derma	97.6 ± 0.7	98.7 ± 0.5	97.2 ± 0.5	98.2 ± 0.5
ecoli	86.9 ± 0.7	91.9 ± 0.7	88.1 ± 0.9	89.9 ± 0.9
flare	81.6 ± 0.5	84.0 ± 1.0	81.7 ± 0.5	84.1 ± 0.6
glas	94.2 ± 0.6	97.8 ± 0.8	93.2 ± 1	95.2 ± 1.2
hear	82.9 ± 1.5	85.8 ± 1.4	83.2 ± 1.7	87.8 ± 1.3
img	96.8 ± 0.2	99.20 ± 0.12	96.9 ± 0.3	97.4 ± 0.3
ionos	90.3 ± 1	95.6 ± 1.0	90.7 ± 1.1	94.0 ± 1.0
mok1	98.3 ± 0.9	100.00 ± 0.00	99.8 ± 0.3	100.00 ± 0.00
mok2	91.1 ± 1.2	99.3 ± 0.2	91.5 ± 1.2	97.5 ± 0.7
pima	75.9 ± 1.2	78.6 ± 1.5	77.4 ± 0.9	79.0 ± 0.8
survi	74.3 ± 1.3	77.5 ± 1.5	75.7 ± 1.5	80.3 ± 1.7
vote	95.6 ± 0.5	95.6 ± 0.5	96 ± 0.6	97.0 ± 0.6
vowel	92.2 ± 0.7	99.5 ± 0.3	97.2 ± 0.3	98.3 ± 0.3
wdbc	96.9 ± 0.5	97.1 ± 0.4	97.1 ± 0	97.7 ± 0.3

According to table 3.5, the networks of the ensembles differ on classifying patterns independently of the network architecture chosen. There are some datasets in which the number of patterns correctly classified by, at least, one network of the ensemble is much greater than the number of patterns correctly classified by the ensemble. This behavior can be clearly seen, for instance, in database *bupa* for *MF* networks and in *band* for *RBF* networks. It also means that, maybe, the *Output Average* may not be the best combiner and that combining ensembles could be more interesting that though.

Furthermore, the diversity of the networks seems to be higher in the case of *MF* networks than *RBF* networks. We can see that the *union* of *MF* is higher than the *union* of *RBF* in 11 of 19 datasets. Moreover, the difference between the performance of the ensemble and "union" tends to be higher for *MF* networks. These results corroborate the affirmation that the ensemble model is more appropriate for *MF* networks, at least in the case of *Simple Ensemble*.

3.6.4.3 Third analysis: *MFSE* versus *RBFSE*

In this last analysis, the results provided by *Simple Ensemble* of *MF* networks, *MFSE*, are compared to the results obtained by *Simple Ensemble* of *RBF* networks, *RBFSE*. The raw results, tables 3.1 and 3.2, showed that the performance of *MFSE* is similar to the performance of *RBFSE* in general. The majority of datasets have similar performance independently of the network architecure choosen to build the

Simple Ensemble. Moreover, there are two datasets, *bala* and *vowel*, whose performance depends on the network architecture. Concretely, *MFSE* performs much better than *RBFSE* for dataset *bala* whereas for dataset *vowel* it is the opposite.

However, there is not any fact that clearly affirm that *MFSE* and *RBFSE* provide similar results. For this reason, the *T-Test* was applied to compare the results provided by *MFSE* and *RBFSE*, the statistical results are shown in table 3.6.

Table 3.6: *T-Test* - *MFSE* versus *RBFSE*

Size	t-value	α
3-net	-2.06	0.04
9-net	-1.74	0.08
20-net	-1.59	0.11
40-net	-1.65	0.10

According to the previous table, the results provided by *MFSE* and *RBFSE* are statistically similar for the cases of ensembles of 9, 20 and 40 networks. The differences are not statistically significant but *RBFSE* performs slightly better according to the t-value. Although *RBFSE* tends to slightly outperform *MFSE*, the time required to generate the networks in *RBFSE* is much higher. There is only one case, ensembles of 3 networks, in which *RBFSE* is statistically better than *MFSE*.

In the rest of this thesis, we will be centered on the analysis of ensembles of *MF* networks for several reasons. Firstly, we consider that ensembles of *MF* networks are interesting because the *MF* network is one of the most extended network architecture. And second, it seems that the ensemble model fits better on *MF* networks because the performance with respect to a single *MF* network is clearly improved by the ensemble.

3.7 Conclusions

There are some alternatives in order to generate a multiple classifier system. In this chapter the ensemble approach has been introduced and a first research has been performed. Concretely, a basic ensemble method has been applied on *MF* and *RBF* networks and important conclusions can be derived from this first study on ensembles of neural networks.

The first conclusion of this chapter is that, according to the experiments done, the simple ensemble approach fits better on *MF* networks than on *RBF*. The increase of performance obtained by the ensembles with respect to the single network is considerably higher for *MF* networks than for *RBF* networks. Moreover, with the ensembles of 3 *MF* networks the increase with respect to a single *MF* network is considerably high whereas the single *RBF* network is only statistically improved by the ensembles of 40 networks. Maybe, the ensembles of *RBF* networks require much more than 40 networks to considerably improve the single network.

Secondly, the networks of an ensemble do not classify patterns in the same way. There are patterns correctly classified by a particular network which are not correctly classified by other networks or the whole ensemble. Concretely, there are some datasets in which the percentage of patterns which have been correctly classified by, at least, one network of the ensemble is much higher than the performance of the ensemble. And this effect, the diversity of the networks in the ensemble seems to be higher in ensembles of *MF* networks. Besides that, the *Output average* is a 'democratic' combiner in which all the networks have the same weight on the final output, so the final output correspond to the decision taken by the majority of networks. For this reason, a deeper analysis on combining neural networks is highly suggested. There are more factors apart from the outputs and the performance of the network which should be considered when the networks of an ensemble are combined.

Thirdly, the differences between the results provided by medium and high sized ensembles of *MF* networks and the results of the ensembles of *RBF* networks of the same sizes are not statistically significant. However, the results are slightly better for the case of ensembles of *RBF* networks.

This thesis will focus on an analysis on ensembles of *MF* networks mainly because of the following reasons:

- It is easier to improve the results of a single network and generate a better classifier according to the first experiments showed in this chapter.
- The *MF* structure is simpler than *RBF* and therefore the procedure to set specific parameters and the algorithm to train the networks are, both, also simpler and they have less computational cost for the *MF* network.
- The computational cost of training a *RBF* network by gradient descent is considerably higher than for the *MF* network.
- The *MF* network is more widely used than *RBF* in classification.

So, the *MF* architecture will be initially considered for further experiments related to ensembles of neural networks and the *RBF* will also be only used in determined cases.

Furthermore, we performed some preliminary experiments on well-known ensemble methods with *RBF* networks whose results were published in [59, 60]. In those experiments, we trained ensembles of *RBF* networks with complex ensemble methodologies such as *Cross Validation Committee*, *Bagging* and *Adaptive Boosting*. The results showed that, in general, the performance of these *advanced* ensemble methods were worse than the performance provided by *Simple Ensemble* of *RBF* networks. Those important ensemble methods will be reviewed and analyzed in the following chapters.

In conclusion, further experiments will be focused on two major areas: Analyzing methods to generate ensembles of neural networks and studying combiners which successfully provide a global output, maximizing the performance of the ensemble.

The *RBF* network will not be utilized in the analysis and development of ensemble methods. But it will be present in the combiners study and, moreover, in further experiments based on more complex models such as *Stacked Generalization* (chapter 9) and *Mixture of Experts* (chapter 10).

Chapter 4

A comparative study of ensemble methods (I)

4.1 Introduction

In this chapter, some methods to design and build ensembles of neural networks will be reviewed and analyzed.

As was previously mentioned, an *Ensemble of Neural Networks* is a well-known type of multiple classifier systems. Moreover, there are some possible sources of diversity and, therefore, there is an important number of different alternatives to design ensembles of neural networks. In order to perform a readable comparison of ensemble variants, the comparison has been split into two chapters. Concretely, the research performed here is focused on the ensembles based on modifications of the training algorithm. The other alternatives, in which the training and validation sets are modified, will be analyzed in the following chapter.

This chapter is organized as follows. Firstly, the reviewed ensemble methods will be described in section 4.2. Secondly, the experimental setup along with the used databases are introduced in section 4.3. Thirdly, the results and their discussion are shown in section 4.4.

4.2 Analyzed Methods

The ensembles that modify the training algorithm are described in this section. They have been divided into two different subgroups depending on how the training algorithm is altered.

The alternatives that *indirectly* change the training algorithm are included in the first subgroup. These ensembles modify the target equation and indirectly change the training algorithm because the adaptation of the trainable parameters is altered. They introduce communication among the different networks of the ensemble inside the error function and their purpose is to build cooperative systems. The following methods will be described in subsection 4.2.1.

- *Cooperative Ensemble Learning System.*
- *Decorrelated.*

The variants that *directly* change the training algorithm are included in the second subgroup. They modify the structure of the training algorithm in order to build an accurate classifier. These ensembles are often complex and require specific training parameters. The methods listed below will be described in subsection 4.2.2.

- *Ensembles Voting On-Line.*
- *Observational Learning Algorithm.*
- *Adaptive Training Algorithm.*
- *Evolutionary Ensemble with Negative Correlation Learning.*

4.2.1 Methods that modify the target equation

In this subsection, some ensemble alternatives that modify the target equation and the training algorithm will be shown.

4.2.1.1 Cooperative Ensemble Learning System

Cooperative Ensemble Learning System, henceforth *CELS*, is an ensemble variant proposed by Liu and Yao that modifies the target equation and, therefore, the learning algorithm [61]. In this methodology, all the networks of the ensemble are trained in parallel. The parameters of the whole networks are adapted in each epoch of the algorithm. Moreover, a *penalty term* is added to the MSE equation in order to produce biased individual networks whose errors tend to be negatively correlated. The new target is described in equation 4.1.

$$Error_{net}(x) = \sum_{c=1}^{N_{classes}} \left(\frac{1}{2} \cdot (d_c(x) - y_{c,net}(x))^2 + Penalty_{c,net}(x) \right) \tag{4.1}$$

Where *Penalty* is described in equation 4.2 and denotes the correlation between a network with respect to the other classifiers of the system. According to [62], the effectiveness of the ensemble can be reduced if the networks are correlated. For this reason this term is added and minimized during the training.

$$Penalty_{c,net}(x) = \lambda \cdot (y_{c,net}(x) - d_c(x)) \cdot \sum_{\substack{i=1 \\ i \neq net}}^{N_{nets}} (y_{c,i}(x) - d_c(x)) \tag{4.2}$$

In equation 4.2, λ denotes the weight of the penalty. The value of this parameter depends on the classification problem so it must be set empirically by trial and error.

Algorithm 4.1 CELS $\{T, V, Parameters\}$

```
for net = 1 to N_networks do
    Set initial values of the trainable parameters for net-network
end for
for e = 1 to epochs do
    for x = 1 to N_patterns do
        for net = 1 to N_networks do
            Adjust the trainable parameters
        end for
    end for
    Calculate MSE of the ensemble over validation set V
    Save ensemble configuration and calculated MSE
end for
Select epoch with minimum validation MSE
Assign best epoch configuration to the ensemble and save it
```

In *CELS*, the equations applied to adjust the trainable parameters must be calculated according to the new target function. In appendix B.2.1 we have included the new equations for *MF* networks.

4.2.1.2 Decorrelated version 1

Decorrelated version 1, henceforth *DECOv1*, is also an ensemble that only modifies the target equation so the training algorithm is slightly altered [62]. In *DECOv1* the networks are trained in serial and the main purpose is to penalize an individual classifier for being correlated with the previously trained one. For this reason Rosen added a *penalty term* to the MSE equation as in (4.3):

$$Error_{net}(x) = \sum_{c=1}^{N_{classes}} \left(\frac{1}{2} \cdot \left(d_c(x) - y_{c,net}(x)\right)^2 + Penalty_{c,net}(x) \right) \tag{4.3}$$

In this case, *Penalty* is described in equation 4.4 and denotes the correlation degree between a network and the previously trained one. Its value increases if the present and the previous networks incorrectly classify the pattern in the same way.

$$Penalty_{c,net}(x) = \lambda \cdot \left(d_c(x) - y_{c,net-1}(x)\right) \cdot \left(d_c(x) - y_{c,net}(x)\right) \tag{4.4}$$

Where λ denotes the weight of the penalty term which must be set empirically by trial and error because it depends on the classification problem. The structure of this ensemble method is shown in algorithm 4.2. In this detailed description we can realize that the first difference between *CELS* and *DECOv1* is that the networks in *DECOv1* are not trained in parallel.

Algorithm 4.2 DECOv1 $\{T, V, Parameters\}$

```
Generate N_networks with different seed values: seed_net
for net = 1 to N_networks do
    Random Generator Seed = seed_net
    Set initial weights randomly
    for e = 1 to epochs do
        for x = 1 to N_patterns do
            Calculate output of the previous network
            Adjust the value of the trainable parameters
        end for
        Calculate MSE over validation set V
        Save epoch weights and calculated MSE
    end for
    Select epoch with minimum validation MSE
    Assign best epoch configuration to the network and save it
end for
```

Finally, we have included the new equations to adjust the weights for *MF* networks in appendix B.2.2.

4.2.1.3 Decorrelated version 2

Decorrelated v2, henceforth *DECOv2*, is an ensemble alternative which was also proposed by Rosen in [62]. In this case, the penalty term is only added to odd networks. Here, the target is given by the following equation:

$$Error_{net}(x) = \sum_{c=1}^{N_{classes}} \left(\frac{1}{2} \cdot \left(d_c(x) - y_{c,net}(x)\right)^2 + Penalty_{c,net}(x) \right) \tag{4.5}$$

And, the *penalty term* is given now by:

$$Penalty_{c,net}(x) = \begin{cases} \lambda \cdot (d_c(x) - y_{c,net-1}(x)) \cdot (d_c(x) - y_{c,net}(x)) & \text{if } net \text{ is odd} \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

The purpose of this penalty is the same than in *DECOv1*, however it is only applied to odd networks. In *DECOv2*, λ also denotes the weight of the new term and it also depends on the problem. As in *DECOv1*, a trial and error procedure must be used in order to empirically set its value.

The description of this ensemble method is shown in algorithm 4.3. The basic structure of *DECOv2* is quite similar to the description of *DECOv1* shown in algorithm 4.2.

Algorithm 4.3 DECOv2$\{T, V, Parameters\}$

```
Generate N_networks with different seed values: seed_net
for net = 1 to N_networks do
  Random Generator Seed = seed_net
  Set initial weights randomly
  for e = 1 to epochs do
    for x = 1 to N_patterns do
      Calculate output of the previous network if net is odd
      Adjust the value of the trainable parameters
    end for
    Calculate MSE over validation set V
    Save epoch weights and calculated MSE
  end for
  Select epoch with minimum validation MSE
  Assign best epoch configuration to the network and save it
end for
```

The equations applied to adjust the trainable parameters in *DECOv1* can also be found in appendix B.2.2.

4.2.2 Methods that directly modify the training algorithm

In this subsection, some ensemble variants that directly modify the training algorithm will be shown.

4.2.2.1 Ensembles Voting On-Line

Ensembles Voting On-Line, henceforth *EVOL*, is an ensemble method proposed by Auda in [63]. In this case, a specific training algorithm is introduced to generate ensembles. Its whole description is shown in algorithm 4.4.

Algorithm 4.4 EVOL $\{T, V, Parameters\}$

for $net = 1$ to $N_{networks}$ **do**
  Set initial weight values for network net
end for
for $e = 1$ to $epochs$ **do**
  for $x = 1$ to $N_{patterns}$ **do**
    for $net = 1$ to $N_{networks}$ **do**
      Calculate:
        $error^{net}(x) = \frac{1}{2} \cdot \sum_{c=1}^{N_{classes}} \left(y_c^{net}(x) - d_c(x)\right)^2$
        $miss^{net}(x) = \begin{cases} 0 & \text{if } class(y^{net}(x)) = class(d(x)) \\ 1 & \text{otherwise} \end{cases}$
    end for
    Calculate ensemble output
      $\overline{y}(x) = voting\left(y\left(x\right)\right)$
    while x is incorrectly classified by the ensemble **do**
      Select individual network net with lowest $error^{net}(x)$ and $miss^{net}(x) = 1$
      while x is incorrectly classified by the network net **do**
        Adjust the value of the trainable parameters of network net with x
        Recalculate: $y^{net}(x)$, $error^{net}(x)$ and $miss^{net}(x)$
      end while
      Recalculate ensemble output
    end while
  end for
  Calculate MSE of the ensemble over validation set V
  Save epoch weights and calculated MSE
end for
Select epoch with minimum validation MSE
Assign best epoch configuration to the ensemble
Save ensemble configuration

The combiner *voting* is used to fuse the outputs of the networks. In *voting*, each network (net) provides a vote to a class (c) given by the highest output of the classifier, y_c^{net}. The class which has been most often voted by all the networks is assigned to the pattern. The number of votes, $\overline{y}$, is calculated with equation 4.7.

$$\overline{y}_c(x) = \sum_{net=1}^{N_{nets}} vote_c^{net} \tag{4.7}$$

Where $vote_c^{net}$ is the vote value of an individual network, net, on a given class, c. Then, the class c with highest $\overline{y}_c(x)$ is assigned to the pattern, x, according to the following equation:

$$Class(x) = \underset{c=1,\ldots,N_{classes}}{\arg\max} \; (\overline{y}_c(x)) \tag{4.8}$$

The technique proposed by Auda is complex and its computational cost is considerably high. All the networks are trained in parallel because the parameters of all the networks are adapted in each iteration.

In *EVOL*, we train the ensemble for some epochs. In each epoch, we present all the patterns from the training set to the ensemble. Concretely, for each pattern, x, from the original training set, T, we do the following three steps.

Firstly, the pattern x is presented to all the networks of the ensemble. For each network, net, of the ensemble, we calculate the *Mean Squared Error* (*MSE*) of the pattern x. With this procedure, we calculate the error vector, $error^{net}(x)$. Moreover, the vector of *missclassified* patterns, $miss$, can be calculated along with the error for pattern x. An element of this vector, $miss^{net}(x)$, denotes if the pattern, x, has been correctly, or incorrectly, classified by the network, net.

Secondly, the output of the ensemble is calculated, $\overline{y}(x)$. To obtain this 'final' output, we use the *voting* scheme. With this procedure, the pattern, x, corresponds to the class, c, which has been the most often voted class by the individual networks. This final output is important because the third step of the training will not be done if the pattern is correctly classified.

Thirdly, the parameters of the networks will be adapted if the pattern, x, is not correctly classified by the ensemble. This step is an iterative training which finishes when the ensemble correctly classifies the pattern x. In each iteration, we adjust the network, net, with lowest error, $error^{net}(x)$, if it has incorrectly classified the pattern, $miss^{net}(x)$ equal to 1. After adapting the network, we recalculate the output of this network, $y^{net}(x)$, its error value, $error^{net}(x)$ and its *missclassification* value, $miss^{net}(x)$. This iteration finishes when the network correctly classifies the pattern. Then, this third step finishes if the ensemble also correctly classifies the pattern. If the ensemble does not correctly classify the pattern, the following network with lower $error^{net}(x)$ and that incorrectly classifies the pattern is chosen and adapted.

Once all the patterns have been *presented*, the *MSE* on the validation set is calculated and the weight configuration of the ensemble is temporally saved.

Finally, after the last epoch, the ensemble configuration of the best epoch, the one with lowest validation error, is chosen as the final ensemble.

4.2.2.2 Observational Learning Algorithm

Observational Learning Algorithm, henceforth *OLA*, is an ensemble alternative based on a social learning theory proposed by Jang [64]. In *OLA*, all the networks of the ensemble are trained simultaneously in a parallel learning procedure. The parameters of all the networks are adapted in each iteration of the training algorithm. According to the reference, *OLA* can improve the generalization of the ensemble with the use of a virtual set when the original training set has insufficient or noisy data.

Firstly, each network, *net*, is trained with a specific training set, T^{net}, for a determined number of epochs. The specific training sets, T^{net}, are generated by *Bootstrap* [65]. With this procedure, the patterns of the specific training set, T^{net}, are randomly sampled with replacement from the original training set, T. The size of the specific sets is equal to the size of the original set. After adapting the networks on the specific training set, T^{net}, a *virtual* set is generated for each network, VT^{net}. Then the network parameters are adjusted using this new set.

The patterns of the virtual set are the patterns of the specific training set. However, random noise is added to the input parameters of each pattern using a normal distribution with low variance. Furthermore, the desired output (target) of these patterns is also modified. The new desired output of a virtual pattern v is given by:

$$d_c(v) = \begin{cases} 1 & \text{if } class(average(\widehat{y}^{-net}(v))) = c \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

For a given virtual pattern, v, its new desired output, $d(v)$, corresponds to the output provided by using *Output average* over all the networks of the ensemble except one, the network which is being trained on this virtual set, *net*. The description of *OLA* is shown in algorithm 4.5

Algorithm 4.5 OLA $\{T, V, Parameters\}$

for $net = 1$ to $N_{networks}$ **do**
  Set initial weight values for network *net*
  Create specific training set T^{net} by randomly sampling from T with replacement
end for
for $i = 1$ to *epochs* **do**
  for $net = 1$ to $N_{networks}$ **do**
    Adjust the trainable parameters of network *net* on T^{net}
  end for
  for $net = 1$ to $N_{networks}$ **do**
    Create the virtual training set VT^{net}
    Adjust the trainable parameters of network *net* on VT^{net}
  end for
  Calculate MSE over validation set V and save weight configuration
end for
Select epoch with lowest validation MSE and assign its weights to the ensemble

4.2.2.3 Adaptive Training Algorithm

Adaptive Training Algorithm, henceforth *ATA*, is a complex ensemble learning procedure proposed by Wanas in [66] and it is introduced in algorithm 4.6.

Algorithm 4.6 ATA $\{T, V, Parameters\}$

P =any constant (usually between 10% to 40%)
$\Gamma = 0.0001$
for $net = 1$ to $N_{networks}$ **do**
  $Converged_{net} = 0$
  Set initial weight values for network net
  Generate the specific training set T^{net} by *Cross-Validation*
  Initialize *Confidence Factor* variables: $CF_{net} = CFbest_{net} = CFprev_{net} = 0$
end for
for $net = 1$ to $N_{networks}$ **do**
  for $e = 1$ to $epochs$ **do**
    Adjust the trainable parameters of network net on T^{net}
  end for
end for
while $(\sum_{net=1}^{N_{networks}} Converged_{net}) < N_{networks}$ **do**
  for $net = 1$ to $N_{networks}$ **do**
    if $Converged_{net} = 0$ **then**
      $Perf_{net}$ = Percentage of correctly classified patterns on validation set, V, by network net
      $CFprev_{net} = CF_{net}$
      $CF_{net} = Perf_{net}$
      $\delta = P \cdot CF_{net}$
      if $CF_{net} > CFbest_{net}$ **then**
        $CFbest_{net} = CF_{net}$
        The network configuration is temporally stored
      end if
      if $|CF_{net} - CFprev_{net}| > \Gamma$ **then**
        Generate the training set MT^{net}
        for $e = 1$ to $epochs$ **do**
          Adjust the trainable parameters of network net on MT^{net}
        end for
      else
        $Converged_{net} = 1$
        Store definitively the network configuration
      end if
    end if
  end for
end while

The complex training algorithm of this ensemble method is divided into two steps.

In the first step, the networks are trained for a few epochs on the specific training sets. In the original paper, there was not any direct reference about creating the different training sets so we decided to apply *K-fold cross-validation* in order to generate them because it satisfies the requirements related to the specific training sets of the ensemble. With this procedure, the original training set, T, is divided into k subsets, T_i, and the specific training sets, T^{net}, are given by equation 4.10:

$$T^{net} = \bigcup_{\substack{i=1 \\ i \neq net}}^{N_{networks}} T_i \tag{4.10}$$

In the second step, the non-converged networks are iteratively trained for a preset number of epochs on a special training set MT^{net} described below. The ensemble training finishes when all the networks have converged.

Before the training on the special set, the performance of the network on the validation set is calculated. In this method the performance is the percentage of correctly classified patterns. Then the *confidence factor* of the network CF_{net} is stored as *previous confidence factor* $CFprev_{net}$ and the calculated performance is assigned as the new *confidence factor* of the network CF_{net}. If this new factor is higher than the *best confidence factor* $CFbest_{net}$, the network configuration is temporally stored and the value of the *confidence factor* is assigned to the *best confidence factor* because the current network is better than the previous one temporally stored.

At this point, we calculate if a network *net* has converged. The network *net* converges if the difference between the *confidence factor* and the *previous confidence factor* is lower than a preset *threshold* value Γ.

If the network has not converged, it is trained with the special training set MT^{net}. According to the original reference, the special training set of a network is composed of two parts. The first part are the patterns from the specific training set T^{net} with the same classes as the patterns from the validation set which were incorrectly classified by the network *net*. We will denote R the number of these patterns. The second part, contains a percentage of patterns, given by equation 4.11, whose classes were correctly classified by the network generated in the previous iteration, *net*. In this equation, P and δ are calculated as in algorithm 4.6.

$$Percentage = 0.01 \cdot R \cdot (P + \delta) \tag{4.11}$$

If the network has converged, the temporally stored configuration, $CFbest_{net}$, is set to the final network. Furthermore, this network will not be used in further iterations.

Two different versions of this technique have been implemented in this thesis. In the first one, *ATA-LE*, the networks are trained with specific training sets, T^{net} and MT^{net}, and the final network configuration always corresponds to the last epoch of the training procedure. In the second one, *ATA-BE*, the networks are also trained with these sets but the network configuration is given by the epoch with lowest MSE on the validation set. The first version is the original method whereas the second one is proposed by us in order to avoid overfitting during the training procedure.

4.2.2.4 Evolutionary Ensemble with Negative Correlation Learning

Genetic algorithms and other evolutionary theories are important alternatives to design ensembles of neural networks [67]. *Evolutionary Ensemble with Negative Correlation Learning*, *EENCL* [68], was proposed by Liu et al. and it is an important technique to generate ensembles.

The basic description of this method is shown in the next algorithm.

Algorithm 4.7 EENCL $\{T\ ,\ V\ ,\ Parameters\}$

Set initial weight values for network *net* $\forall net \in [1...N_{networks}]$
Ensemble training with Negative Correlation Learning, *NCL*
for $g = 1$ to $N_{generations}$ **do**
  Select $N_{descents}$ descent networks
  Mutate $N_{descents}$ descent networks
  Train the descents by applying NCL
  Calculate network *net* fitness $\forall net \in [1...N_{networks} + N_{descents}]$
  Select $N_{networks}$ best networks according to fitness values
  Generate ensemble with best networks
end for
Generate final ensemble

The basic structure of this ensemble is decomposed in the following main steps:

- Basic training algorithm, *NCL* in this case.
- Selection of the descents.
- Mutation of the selected descents.
- Evaluation of the networks, in this case based on *fitness*.
- Replacement strategy.
- Combination of the networks.
- Building the final ensemble.

EENCL was proposed by the same author who proposed *CELS*. For this reason, the base training algorithm applied, *NCL*, is similar to *CELS*. In this case, the ensemble methodology is not only given by the training algorithm because all the other steps previously listed are also important.

4.2.2.4.1 Negative Correlation Learning

Negative Correlation Learning, henceforth *NCL*, is the algorithm proposed to adjust the weights of the networks in this evolutive technique. This procedure is applied at the beginning of the ensemble algorithm in order to establish the first generation of networks. It is also used to train the mutated descents generated in each generation of the ensemble method.

NCL is a training algorithm in which the target equation and the basic training algorithm described in chapter 2 are modified. In fact, all the networks of the ensemble are trained in parallel. Moreover, a *penalty term* is added to the MSE equation in order to generate negatively correlated networks, being the new target equation the following:

$$Error_{net}(x) = \sum_{c=1}^{N_{classes}} \left(\frac{1}{2} \cdot (d_c(x) - y_{c,net}(x))^2 + Penalty_{c,net} \right) \tag{4.12}$$

And the weighted penalty error is given by:

$$Penalty_{c,net}(x) = \lambda \cdot (y_{c,net}(x) - \overline{y}_c) \cdot \sum_{\substack{i=1 \\ i \neq net}}^{N_{networks}} (y_{c,i}(x) - \overline{y}_c) \tag{4.13}$$

Where λ denotes the weight of the penalty term and $\overline{y}$ denotes the output of the ensemble when the *Output average* is applied to combine the networks. The value of λ depends on the problem and a trial and error procedure is used to empirically set its value. *Penalty* denotes the correlation between a network with respect to the other networks of the ensemble. According to [62], this term is added in the error and minimized during the training because the accuracy of an ensemble can be reduced when the correlation degree among the networks is high.

The equations applied to adjust the trainable parameters must be calculated according to the new target function. These new equations are included in appendix B.2.3 for the case of *MF* networks. The description of *NCL* is shown in *algorithm* 4.8.

Algorithm 4.8 NCL $\{T, V, Parameters\}$

```
Initialize trainable parameters for all networks
for e = 1 to N_epochs do
  for j = 1 to N_networks do
    Adjust the trainable parameters on the training set T for j-network
  end for
end for
Save ensemble configuration
```

As we can see, *EENCL* is composed by two training algorithms. The main technique establishes the evolutive process whereas *NCL* is used to train the networks and adapt the network parameters.

4.2.2.4.2 Selecting descents

In *EENCL*, the procedure applied to select the descent networks is quite easy. At the beginning of each generation, a few networks of the ensemble are randomly picked out with replacement in order to generate the descents of the generation. All the networks have the same probability of being selected.

4.2.2.4.3 Mutation of the descent networks

Once the descents have been selected, their configuration is altered in two steps. Firstly, some noise is added to the trainable parameters, weights of the networks. This noise is given by the standard normal distribution $N(0,1)$. Then, *NCL* is used to train only the descents. But in this step, the information provided by the original networks and the descents is used to calculate the error according to equation 4.12.

4.2.2.4.4 Fitness Evaluation

Fitness is introduced in order to obtain the suitability of the network with respect to the ensemble. It is given by the following algorithm:

Algorithm 4.9 $Fitness\{ensemble, N_{networks}, N_{descents}\}$

$Fitness_{net} = 0 \; \forall \; net \in [1 \ldots N_{networks} + N_{descents}]$
for $x = 1$ **to** $N_{patterns}$ **do**
  for $net = 1$ **to** $N_{networks} + N_{descents}$ **do**
    Classify pattern x with network net
  end for
  Let $Total$ be the number of networks which correctly classify x
  for $net = 1$ **to** $N_{networks} + N_{descents}$ **do**
    $Fitness_{net} = Fitness_{net} + \begin{cases} \frac{1}{Total} & \text{if } x \text{ is correctly classified by } net \\ 0 & \text{otherwise} \end{cases}$
  end for
end for

If a pattern, x, is learned correctly by n networks, each of these n networks receives $1/n$ fitness. A network which correctly classifies a hard to learn pattern (n is low) is given better fitness, and it has better chance to continue in the ensemble. This fitness encourages each individual net to cover different patterns in the training set.

4.2.2.4.5 Replacement strategy

Once the fitness of each network is calculated, the final ensemble has to be built. In *EENCL* the $N_{networks}$ networks with highest fitness are selected to form the ensemble of the next generation.

4.2.2.4.6 Combination of the networks

The combiner *Output average* used in *Simple Ensemble* is also used to fuse the outputs provided by the networks and calculate the final output vector $\overline{y}$.

4.2.2.4.7 Building the final ensemble

Although in the original reference the final ensemble corresponds to the system obtained at the last generation of the evolutionary algorithm, we have also selected the *best generation*, the generation with lowest *MSE* on the validation set, in order to built the final ensemble. For this reason we will evaluate two different evolutionary ensemble methods, *EENCL-LG* which corresponds to the last generation and *EENCL-BG* which is the version we propose based on selecting the best generation.

4.3 Experimental Setup

The comparative research of this chapter is concerned with ensembles in which the training procedure is, directly or indirectly, modified.

Concretely, eight different alternatives have been reviewed. The different models can not be directly compared with the information and results provided in the original references because each author has used a specific set of databases and test setup in order to validate the results of the proposed methods. Moreover, non public databases are sometimes used to test new ensemble variants.

We consider that the comparison we want to perform can be useful in order to determine the ensemble methods which provide the best performance in general. In the comparison, we will use an experimental framework that can be reproduced by any researcher interested in ensembles of neural networks.

In this research, we want to obtain the performance of the analyzed alternatives on some different cases. These cases are different sizes of the number of networks in the ensemble and different partitions of the databases. The main characteristics of the performed research are:

- Nineteen classification problems from the *UCI Repository*.
- The eight methods reviewed are applied to generate the ensembles.
- One network architecture for the networks in the ensemble: *MF*.
- Four different ensemble sizes: 3, 9, 20 and 40 networks.
- Optimized training parameters.
- One combiner applied, *Output average*, except for *EVOL* (*Voting*).
- The experiments have been repeated ten times with different partitions of training, validation and test sets in order to obtain:
  - The mean value of performance.
  - The error rate by standard error theory.
- Three general measurements applied to the comparison:
  - Mean *Increase of Performance*.
  - Mean *Percentage of Error Reduction*.
  - *Paired Student's t-test*.

The description of the nineteen databases used in the experiments can be found in appendix A. The optimized training parameters of the networks are in appendix B.3, whereas the specific parameters of the ensembles are in appendix B.5. Finally, the general measurements applied to compare the ensembles analyzed are the mean *Increase of Performance* (equation 3.5), the *Percentage of Error Reduction* (equation 3.6) and the *Paired Student's t-test* (section 3.6.3.2).

4.4 Results and discussion

The main results related to this ensemble comparison are shown in this section. Concretely, table 4.1 shows the *mean IoP* and the *mean PER* of the ensemble methods analysed in this chapter for four different ensemble sizes.

Table 4.1: Ensemble Comparison 1 - General Results

	Mean *IoP*				**Mean *PER***			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Simple Ensemble	4.97	5.27	5.34	5.43	21.84	23.65	23.73	24.64
CELS	4.72	5.42	5.46	5.53	21.51	23.73	25.75	26.35
DECOv1	5.48	5.78	5.85	5.83	24.76	26.6	26.99	27.03
DECOv2	5.46	5.53	5.58	5.78	24.87	25.71	25.92	26.4
EVOL	−10.71	−16.09	−19	−21.15	−144.86	−211.67	−271.14	−290.72
OLA	1.01	−2.52	−3.05	−2.85	−16.26	−58.59	−64.29	−61.41
ATA-LE	4.58	4.91	5.01	4.77	19.44	22.05	22.18	20.22
ATA-BE	4.42	5.24	5	5.02	19.97	23.4	22.58	21.59
EENCL-LG	4.41	4.06	4.52	4.2	16.87	14.93	15.99	17.71
EENCL-BG	4.89	4.82	4.67	5.07	20.42	19.6	19.02	22.36

According to the results shown in the previous table, *Decorrelated* provides the best results. Although both versions of *Decorrelated* provide similar results, *DECOv1* is slightly better than *DECOv2*.

Similarly, *CELS* also provides good results for any ensemble size and it works better than *Simple Ensemble* for 9, 20 and 40 network ensembles. However, *CELS* is slightly worse than the two implementations of *Decorrelated* for any ensemble size.

Moreover, *EVOL* and *OLA* perform worse than a single *MF* network because the *PER* value is always negative. *EVOL* provides a mean decrease of performance higher than 10%. In the original references [63, 64], the authors obtained good results but they were only tested on one dataset. The *Cleveland Heart Disease* was used to evaluate *EVOL* and an artificial classification problem was used to test *OLA*. Although these methods had good performance on the dataset employed in the original references, this does not mean that they have good performance on other classification problems because they were only tested in one dataset and their results can not be generalized.

Although *ATA* provides good results, its performance is lower than the performance provided by *Simple Ensemble*. The version proposed in this thesis, *ATA-BE*, provides slightly better results than the original implementation, *ATA-LE*, in a wide number of cases. In the original reference [66], *ATA* was tested on three datasets: a *20-class problem* from [69], the *Glass Dataset* [70] and the *Satimage database* [70]. According to the results shown in the reference, *ATA* performed better than a normal ensemble on the first and second datasets. However, the normal ensemble performed better than *ATA* for the third dataset.

In the case of *EENCL* [68], it provides good results but they are also lower than the results provided by *Simple Ensemble*. Moreover, the version proposed in this thesis, *EENCL-BG*, performs clearly better than the original proposal, *EENCL-LG*. In the original reference, two databases from *UCI repository* were used in the experiments. In both datasets, *ENNCL* performed better than a simple ensemble of *MF* networks.

The general performance shown in this research for *ATA* and *ENNCL* may be considered more accurate than the performance shown in the original references [66, 68] because the number of networks in the ensemble and datasets used here are higher. Maybe, an experimental setup with only two datasets, as used in the original references, may not provide the general performance of a method.

As mentioned above, *CELS*, *DECOv1* and *DECOv2* perform better than *Simple Ensemble*. Unfortunately, the general measurements applied do not provide any statistical information so the *T-Test* was applied three times. Firstly, it has been used to compare these three ensembles to *Simple Ensemble*, table 4.2. Secondly, the three methods have been statistically compared among them, table 4.3. Thirdly, the t-test will be applied to compare the original versions of *ATA* and *ENNCL* to the versions we propose of them in this Thesis, table 4.4.

Table 4.2: Statistical results of the best methods versus *Simple Ensemble*

Methods	**measure**	3-**net**	9-**net**	20-**net**	40-**net**
CELS vs SE	t-**value**	−1.15	0.86	0.66	0.47
	α	0.25	0.39	0.51	0.64
DECOv1 vs SE	t-**value**	3.05	3.55	3.97	3.35
	α	0.0026	0.0005	0.0001	0.001
DECOv2 vs SE	t-**value**	2.89	2.39	2.59	3.47
	α	0.0044	0.018	0.011	0.0006

According to the results shown in the previous table, both version of *Decorrelated* improve the results provided by *Simple Ensemble*, *SE*, and the differences are statistically significant. In both versions, it can be seen that the values of α are quite low so the results provided by them are quite different when compared to *SE*. Moreover, *CELS* provides slightly better results than *SE* for the cases of 9, 20 and 40 networks but the differences between them are not statistically significant.

Table 4.3: Statistical results among the best methods

Methods	**measure**	3-**net**	9-**net**	20-**net**	40-**net**
CELS vs DECOv1	t-**value**	−3.63	−2.83	−2.15	−1.4
	α	0.0004	0.0052	0.033	0.16
CELS vs DECOv2	t-**value**	−3.32	−0.72	−0.77	−1.09
	α	0.0011	0.47	0.44	0.28
DECOv1 vs DECOv2	t-**value**	0.13	1.92	1.92	0.44
	α	0.9	0.056	0.057	0.66

According to the results of table 4.3, both versions of *Decorrelated* perform better than *CELS* because the t-value is always negative in the first and second row of table 4.3. The differences between *DECOv1* and *CELS* are statistically significant for ensembles of 3, 9, 20 networks and the differences of *DECOv2* and *CELS* are only statistically significant for the case of ensembles of 3 networks. Although *DECOv1* and *DECOv2* are similar in general, the differences between them are almost significant for the cases of ensembles of 9 and 20 networks because α is close to 0.05 and *DECOv1* performs better than *DECOv2*.

For the ensembles based on *ATA* and *EENCL*, the original implementation according to the references has been compared to the version we have proposed to avoid overfitting in table 4.4. For this reason, *ATA-LE* is statistically compared to *ATA-BE* and a similar comparison is done for *EENCL-LG* with respect to *EENCL-BG*.

Table 4.4: Statistical results of original *ATA* and *EENCL*

Methods	**measure**	**3-net**	**9-net**	**20-net**	**40-net**
ATA-LE vs ATA-BE	t**-value**	$+0.93$	-1.58	-0.04	-1.22
	α	0.35	0.12	0.97	0.22
EENCL-LG vs EENCL-BG	t**-value**	-2.89	-3.88	-0.8	-3.86
	α	0.0043	0.0001	0.43	0.0002

According to the results of table 4.4, the differences between both versions of *ATA* are not statistically significant but the t-value shows that our version is slightly better for ensembles of 9 to 40 networks.

Moreover, the proposed version of *EENCL*, *EENCL-BG*, is in general better than the original method and their differences are statistically significant except for ensembles of 20 networks where α is high.

4.5 Conclusions

In this chapter a comparison among traditional ensemble methods based on modifications of the training algorithm has been performed. Important conclusions can be derived from this first comparison.

Firstly, not all the ensemble alternatives proposed in the bibliography improve the accuracy of *Simple Ensemble*. Most of them provide results which are quite similar to the results provided by *Simple Ensemble*. Even more, there are two of them, *Ensembles Voting On-Line* and *Observational Learning Algorithm*, in which the ensembles perform worse than a single *Multilayer Feedforward* network. The results of these two ensemble variants were not expected since they reported good performance in the original references. Concretely, *EVOL* could draw accurately the decision boundaries because each network focused on different regions in the input space according to the original reference. In addition, the original experiments performed in *OLA* showed that this ensemble alternative could give better generalization performance than *Simple Ensemble* and *Bagging* in regression and classification tasks. Maybe, the use of only one classification problem, as reported in their original references,

is not enough in order to perform a comparison with other classifiers such as *Simple Ensemble* because the general behavior of a classification system could not be obtained with only one dataset.

Secondly, only three methods *CELS*, *DECOv1* and *DECOv2* improved the results provided by *Simple Ensemble*. Unfortunately, *CELS* only seems to be slightly better than *Simple Ensemble* because the differences between them are not statistically significant. The two versions of *Decorrelated* are clearly and statistically better than *Simple Ensemble*. Moreover, *DECOv1* is better than *CELS* in most of the cases and it is only slightly better than *DECOv2* according to the statistical tests.

Thirdly, the version proposed in this thesis of *ATA*, *ATA-BE*, is slightly better than the original implementation, *ATA-LE*. Moreover, the alternative we have proposed of *EENCL*, *EENCL-BG*, is also statistically better than the original version, *EENCL-LG*. Selecting the final network configuration according to the lowest MSE on the validation set is an important improvement which is highly suggested instead of selecting the configuration associated to the last iteration, epoch or generation.

However, the original and proposed versions of these two ensembles, *ATA* and *EENCL*, perform slightly worse than *Simple Ensemble* according to the results of our experiments. In the original references, the original implementations tend to perform better than other ensembles such as *Simple Ensemble*. These comparisons were done with a reduced number of datasets and, maybe, their general behavior can not be obtained by only two or three classification problems.

Finally, both versions of *Decorrelated* can be considered the best ensemble methods among the alternatives analyzed in this chapter. They will be considered for the list of best ensembles and *DECOv1* will be used in the combiners study. In the best cases, the increase of performance is close to 6% and the percentage of error reduction is around 27% with respect to a *Single MF Network*. The results shown in this chapter have been published in references [MFC1, MFC2, MFC3] detailed in the conclusions chapter.

Chapter 5

A comparative study of ensemble methods (II)

5.1 Introduction

In this chapter, some ways to design and build ensembles of neural networks will be analyzed and compared. Concretely, the research performed here is focused on ensembles which modify the learning set.

The current research is organized as follows. Firstly, the reviewed ensemble alternatives will be described in section 5.2. Secondly, the setup of the experiments is introduced in section 5.3. Moreover, the results and their discussion are shown in section 5.4.

5.2 Analyzed Methods

In this section, ensembles which are based on modifications of the learning set are reviewed. Since there are several approaches to modify the learning set, the different alternatives have been split into the following three different subgroups: Models that *Statically* modify the learning set, alternatives that *Dynamically* modify the learning set and *Boosting* variants.

The models that modify the learning set statically are those that randomly generate the training sets for the different networks before training the networks of the ensemble. Any previous study on the training set patterns is not required. Then, each network is independently trained using the corresponding training set as an independent network. The following methods will be described in subsection 5.2.1.

- *Bagging* and its variants.
- Different versions of *Cross-Validation Committee*.
- *Disjoint partitions*.
- *Overlapping partitions*.

The design alternatives that modify the learning set dynamically are those that modify the learning set of the networks during training. The training algorithm of this kind of ensembles should be modified so they were included in the previous chapter, concretely in subsection 4.2.2. We consider in this category:

- *Observational Learning Algorithm*.
- *Adaptive Training Algorithm*.

The variants based on boosting are those that construct a sequence of networks in which their successive training sets are overfitted with hard to learn patterns by the previous networks. A sampling distribution is used to generate the different training sets. The following *Boosting* methods will be reviewed in subsection 5.2.2.

- *Boosting 3*.
- *Adaptive Boosting and its variants*.
- *Arcing Classifiers*.

5.2.1 Methods that statically modify the learning set

5.2.1.1 Bagging

Bootstrap Aggregating, henceforth *Bagging*, is the simplest method that modifies the learning set. *Bagging* was proposed by Breiman in [71]. In this model, each training set is generated for each network by randomly sampling patterns with replacement from the original training set. According to reference [48], the generated training sets should double the original training set size.

5.2.1.2 Bagging with noise

Bagging with noise, henceforth *Bagnoise*, is a variant of *Bagging*. In this case, the size of the generated training sets is ten times the size of the original training set. Furthermore, the input patterns contains random noise generated by a normal distribution with low variance. *Bagnoise* was proposed by Raviv in [72].

5.2.1.3 Cross Validation Committee version 1

Cross-Validation Committee version 1, henceforth *CVCv1*, is an ensemble in which the training set is randomly split into some subsets of the same size, T_i. The number of subsets corresponds to the number of networks of the ensemble. The training set of the network, T^{net}, is given by equation 5.1.

$$T^{net} = \bigcup_{\substack{i=1 \\ i \neq net}}^{N_{networks}} T_i \tag{5.1}$$

In figure 5.1, a graphical diagram related to the generation of the different training sets is shown. A 5-network ensemble is supposed in the figure. The validation set is kept unchanged for all the networks in the ensemble.

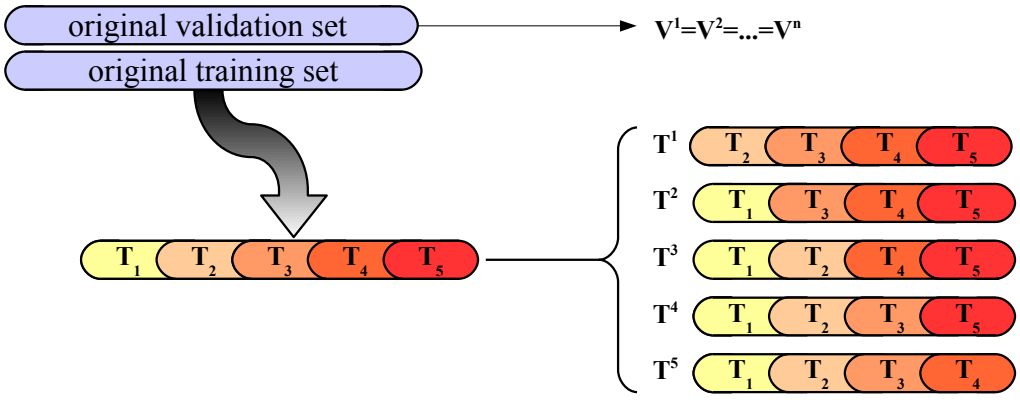


Figure 5.1: Generation of the different training sets for Cross-Validation version 1

This ensemble method, also called *Cross-Validation leaving-one-out*, is fully described in references [73, 74].

5.2.1.4 Cross Validation Committee version 2

The second version of Cross-Validation Committee is a variant of *CVCv1*. In this case, the original learning set, composed with the patterns from the training and validation sets, is split into k different subsets of the same size, L_i. The new training set of a network, T^{net}, is generated as in *CVCv1* but the omitted subset, L_{net}, is used for validation, V^{net}. These specific sets are given by the following equations:

$$T^{net} = \bigcup_{\substack{i=1 \\ i \neq net}}^{N_{networks}} L_i \qquad V^{net} = L_{net} \tag{5.2}$$

In figure 5.2, a graphical diagram related to the generation of the different sets is shown. In this case, there are specific training and validation sets for each network. As in the previous method, the example is given by an ensemble of 5 networks.

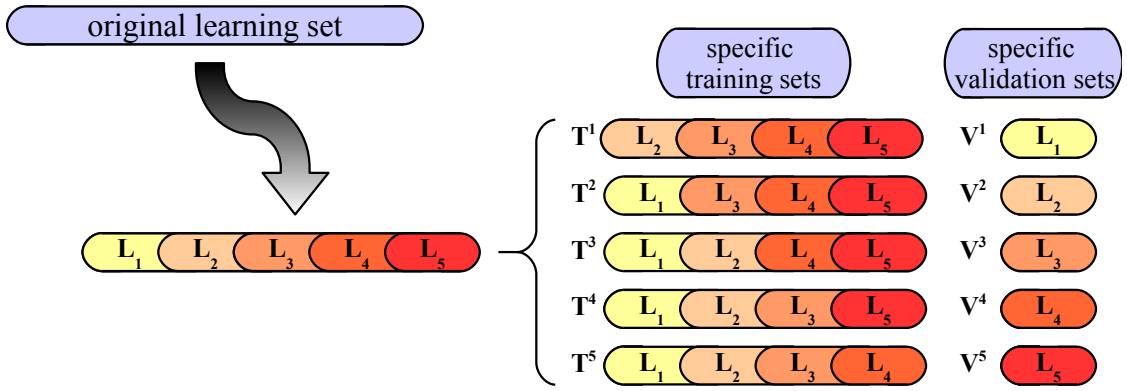


Figure 5.2: Generation of the different training sets for Cross-Validation version 2

The whole description of *CVCv2* can be found in references [75, 76].

5.2.1.5 Cross Validation Committee version 2.5

One important characteristic of the two previous methods is that the subset size inversely depends on the number of networks. There are some problems derived from this fact. On the one hand, if the ensemble is composed by a reduced number of networks, the generated training sets are smaller than usual whereas the generated validation sets are larger and the performance of the individual networks usually decreases because of the use of a small training set. On the other hand, if the ensemble is composed by a high number of networks, the generated training sets are similar to the original training set in *CVCv1* and similar to the original learning set in *CVCv2* and the diversity and performance of the ensemble decreases. Moreover, the validation sets are smaller than the original validation set, being insignificant in some cases.

In this new method proposed by us, *Cross Validation Committee v2.5*, the number of subsets generated, $N_{subsets}$, does not depend on the ensemble size. In this way, the training and validation sets can almost keep their original sizes and the problems derived can be avoided if $N_{subsets}$ is properly set. To set its value, we have generated ensembles with different values of $N_{subsets}$, between 3 and 10, and the best results were obtained when it was set to 5. Moreover, the size of the specific training and validation sets is similar to the size of the original sets with this value. For these reasons, we have finally set $N_{subsets}$ to 5 in all the experiments done.

The training and validation sets in this version of *CVC* are given by the following equations:

$$T^{net} = \bigcup_{\substack{i=1 \\ i \neq index_{net}}}^{N_{networks}} L_i \qquad V^{net} = L_{index_{net}} \tag{5.3}$$

Where the index related to a specific neural network, $index_{net}$, depends on the modulus after division of the number of the network, net, and the total number of subsets. Its value is given by:

$$index_{net} = 1 + modulus\left(\frac{net - 1}{N_{subsets}}\right) \tag{5.4}$$

The graphical diagram related to the generation of the different training sets is shown in figure 5.3. In this case, a n-network ensemble is supposed in the figure and $N_{subsets}$ is 5 as set in our experiments.

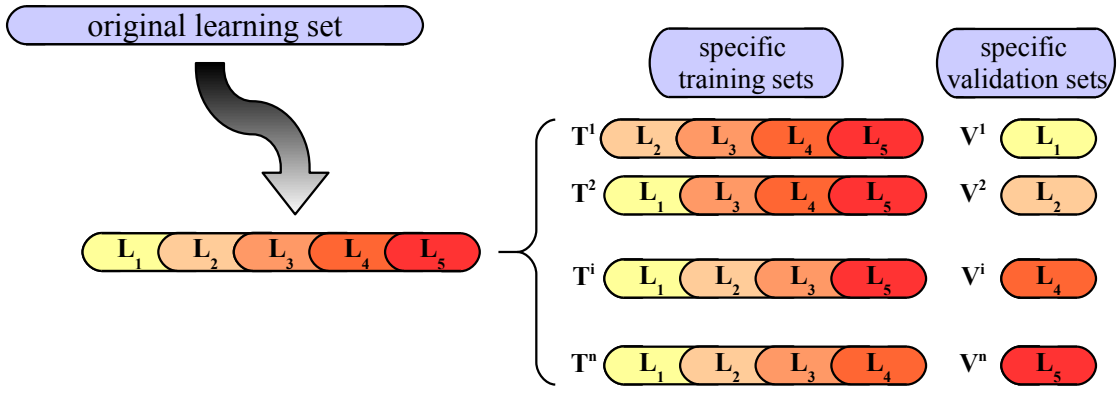


Figure 5.3: Generation of the different training sets for Cross-Validation version 2.5

Although the problems related to previous versions are avoided, the training and validation sets are repeated every five networks. It can be considered a disadvantage because the final network configuration depends on the training and validation sets so the diversity among networks may be reduced.

5.2.1.6 Cross Validation Committee version 3

In the previous ensemble, *CVCv2.5*, the generated training and validation sets can be shared by some networks. To avoid this minor problem, we propose *Cross Validation Committee v3* (*CVCv3*). In this proposed version, the learning set is also split into $N_{subsets}$ but the training and validation sets are given by equation 5.5. In this case, $N_{subsets}$ is now set to 10.

$$T^{net} = \bigcup_{\substack{i=1 \\ i \neq index_{net,1} \\ i \neq index_{net,2}}}^{N_{subsets}} L_i \qquad V^{net} = L_{index_{net,1}} \cup L_{index_{net,2}} \tag{5.5}$$

Where the indexes related to a neural network, $index_{net,1}$ and $index_{net,2}$, are randomly set with the constraint that the different networks have different training and validation sets. Here, we show an example of these indexes.

	$index_{net,1}$	$index_{net,2}$
net_1	1	6
net_2	2	7
...		
net_n	5	8

In order to explain better *CVCv3*, a graphical diagram related to the generation of the different sets is shown in figure 5.4. As in *CVCv2.5*, an n-network ensemble is supposed in the figure but, in this case, there are 10 subsets in this method which is the value used in our experiments.

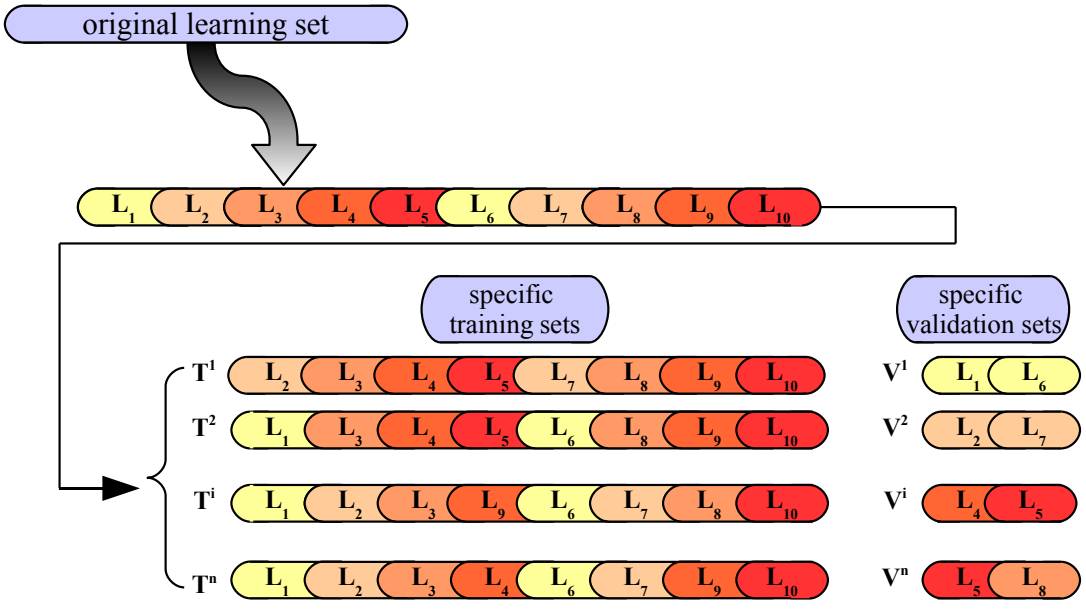


Figure 5.4: Generation of the different training sets for Cross-Validation version 3

5.2.1.7 Disjoint Partitions

Disjoint Partitions, henceforth *DP*, is an ensemble method in which the training set is randomly split into k disjoint subsets of the same size. Each network is trained with the corresponding disjoint subset. The union of the different disjoint subsets is the original training set.

This ensemble was proposed in [77], a graphical example is shown in figure 5.5. Where the training set for the first network is $T^1 = [x_5, x_9]$, the training set for the second network is $T^2 = [x_2, x_7]$ and so on.

At first sight, this procedure has two disadvantages. Firstly, the training set of the different networks is very small and the performance of the individual networks decreases. Secondly, the validation set is the same for all the networks and the diversity of the ensemble can be also decreased for this reason.

5.2.1.8 Disjoint Partitions with Replications

Disjoint Partitions with Replications, henceforth *DPR*, is an *evolution* of *DP* in which the training set is also split into k disjoint subsets. For each disjoint subset, elements from the same subset are randomly selected and added to this subset until its size is equal to the size of the original set.

Then, each network is trained with its corresponding subset. Although new elements are added to the disjoint sets, there is not any overlap among them. The intersection between two disjoint sets is the void set.

This method was also proposed in [77], a graphical example is also shown in figure 5.5. Under our point of view, *DPR* increases the sizes of the training sets with respect to *DP*. But the diversity of the particular training set is the same and the validation sets are equal for all networks.

5.2.1.9 Overlapping Partitions

Overlapping Partitions, henceforth *OP*, is an ensemble method which slightly differs from *DP*. In *OP*, the training set is also randomly split into k disjoint partitions but a certain percentage of patterns are randomly selected from the disjoint partitions. They will form a *common* set of patterns and these patterns are excluded from the original disjoint subsets. The union of the selected patterns in the *common* set with the remaining disjoint subsets will form the overlapping subsets.

Finally, each network is trained with the corresponding overlapping subset. The union of the different overlapping subsets is the original training set whereas the intersection of the different overlapping sets are the patterns of the *common* set randomly selected from the disjoint partitions.

Figure 5.5 shows a graphical example of this ensemble proposed by Dara in [77].

5.2.1.10 Overlapping Partitions with Replications

Overlapping Partitions with Replications, henceforth *OPR*, is an ensemble method similar to *Overlapping Partitions*. But in this case, the patterns of each overlapping subset are replicated until the size of the overlapping subset is equal to the size of the original set. It is a mixture between *OP* and *DPR*. Then, each network is trained with the corresponding overlapping subset with replications. This model was also proposed in [77], and its graphical example is also shown in figure 5.5.

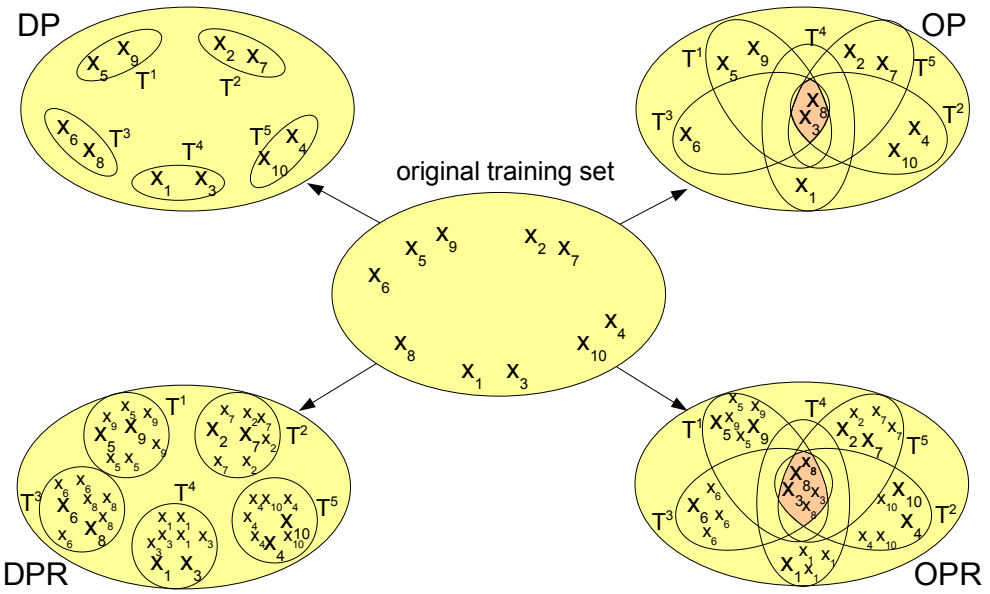


Figure 5.5: Generating the training set for Disjoint and Overlapping partitions

5.2.2 Boosting methods

In this subsection some boosting variants will be reviewed. The origins of Boosting are dated in 1988 when Kearns openly asked 'Can a set of weak learners create a single strong learner?' [78]. The early *Boosting models* proposed by Freund and Yoav were the affirmative answer of this question [79].

Most of the methods reviewed in this thesis are improved versions of *Adaptive Boosting* [79] in which some modifications have been proposed by different authors to design better ensembles. In fact, in this thesis we also propose modifications of *Adaptive Boosting*. In this section, the *classic* boosting implementations are reviewed but the boosting alternatives proposed by us in this thesis will be described and analyzed in chapter 8.

5.2.2.1 Boosting 3

Boosting 3, henceforth *Boost3*, is a method to design an ensemble of 3 networks as described in [80]. Each network is trained on different training sets based on the performance of the previous network.

The first network is trained on the original training set T. Once it has been trained, it is tested on the same set marking their patterns with information related to the correct or incorrect classification by the network. Then a new training set of the same size is built T^2. In this new set, half of the patterns corresponds to patterns which were incorrectly classified by the previous network and the other half corresponds to patterns which were correctly classified. This dataset is built by randomly selecting patterns with replacement from the original training set with the parity restriction commented.

Then, the second network is trained with the new training set T^2. Once the network has been trained, it is tested again on the original training set marking their patterns with information related to the classification by the second network. If a pattern is not correctly classified by the first and second network then it is added to a new training set T^3.

Finally, once the new training set T^3 is generated, the third network is trained using this new set. The ensemble learning procedure finishes when the third network has been trained.

5.2.2.2 Adaptive Boosting

Adaptive Boosting, henceforth *Adaboost*, is an important ensemble alternative proposed by Freund and Schaphire in [79]. *Adaboost* constructs a sequence of networks in which the training sets of the successive classifiers are overfitted with hard to learn patterns by the previous networks.

In *Adaboost*, the successive training sets are generated by randomly sampling with replacement, but the probability of selecting a pattern, x, depends on a sampling probability distribution, called $Dist$, where each pattern has a particular probability of being selected, $Dist_x$. This sampling distribution, $Dist$, is updated after training each network and the probability of selecting a pattern increases if the trained network does not correctly classify the pattern, $h^{net}(x)$ is not equal to $d(x)$, whereas the probability decreases if the pattern is correctly classified.

A basic training diagram of *Adaboost* is shown in figure 5.6 and the description is shown in algorithm 5.1.

Adaboost uses a specific model to combine the output provided by the networks of the ensemble. This combination method, henceforth *boosting combiner*, is described in equation 5.6.

$$h(x) = \underset{c=1,..,N_{classes}}{\arg\max} \sum_{net:h^{net}(x)=c}^{N_{networks}} \log \frac{1-\epsilon^{net}}{\epsilon^{net}} \tag{5.6}$$

Where ϵ^{net} and h^{net} are described in algorithm 5.1 and *arg max* is the maximum function. The final hypothesis, $h(x)$, is calculated by a weighted voting system based on the error of all the networks, ϵ^{net}, and their predictions, h^{net}. Each network, *net*, assigns a vote value, $\log \frac{1-\epsilon^{net}}{\epsilon^{net}}$, to the predicted class, $h^{net}(x)$ equal to c. If the error of the network is low, its vote value will be high and this network will have more importance in the final ensemble classification. Finally, the class c with highest final vote value, arg max of the summatory, is associated to the pattern.

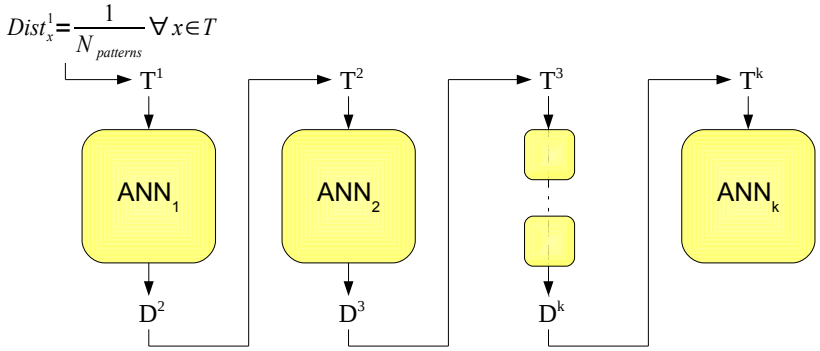


Figure 5.6: Adaboost training basic diagram

Algorithm 5.1 Adaboost $\{T, V, N_{networks}\}$

Initialize Sampling Distribution *Dist*:
$Dist_x^1 = 1/N_{patterns}\ \forall x \in T$
for $net = 1$ to $N_{networks}$ **do**
Create T' sampling from T using $Dist^{net}$
MF Network Training T' , V
Calculate misclassified vector:

$$miss_x^{net} = \begin{cases} 1 & \text{if } h^{net}(x) \neq d(x) \quad x \text{ is incorrectly classified by } net \\ 0 & \text{otherwise} \quad x \text{ is correctly classified by } net \end{cases}$$

$h^{net}(x)$ is the class associated to the pattern x according to the output of *net*
$d(x)$ is the target associated to the pattern x
Calculate error:
$\epsilon^{net} = \sum_{x=1}^{N_{patterns}} Dist_x^{net} \cdot miss_x^{net}$
Update sampling distribution:

$$Dist_x^{net+1} = Dist_x^{net} \cdot \begin{cases} \frac{1}{(2 \cdot \epsilon^{net})} & \text{if } miss_x^{net} \\ \frac{1}{2 \cdot (1-\epsilon^{net})} & \text{otherwise} \end{cases}$$

end for

Although *Adaboost* is considered one of the most important ensembles, some authors like Breiman [81], Kuncheva [82] or Oza [83] have deeply analyzed variants and successfully improved it. Better results are obtained by modifying the equation used to update the sampling distribution, *Dist*, or by adding new constraints to the original algorithm. Unfortunately, any deep research on comparing different combination methods for boosting ensembles has not been done yet.

5.2.2.3 Averaged Boosting

Oza proposed a variant of *Adaboost* in [83] called *Averaged Boosting*, henceforth *Aveboost*. In *Aveboost*, the sampling distribution related to a neural network, $Dist^{net+1}$, depends on three factors.

- The sampling distribution of the previous network, $Dist^{net}$.
- The equation used to update the distribution of *Adaboost*, equation 5.8.
- The number of networks previously trained, *net*.

The basic algorithm structure and the specific combiner are keep unchanged with respect to the algorithm described in *Adaboost*.

In *Aveboost*, the equation to update the sampling distribution is given by:

$$Dist_x^{net+1} = \frac{net \cdot Dist_x^{net} + C_x^{net}}{net + 1} \tag{5.7}$$

where:

$$C_x^{net} = Dist_x^{net} \cdot \begin{cases} \frac{1}{2 \cdot \epsilon^{net}} & \text{if } x \text{ is incorrectly classified by } net \\ \frac{1}{2 \cdot (1 - \epsilon^{net})} & otherwise \end{cases} \tag{5.8}$$

As it can be observed in equation 5.7, the weighted average highly depends on the number of networks previously trained, *net*. The strength of the sampling distribution update decreases as the value of *net* increases. The limit of $Dist^{net+1}$ when *net* increases in equation 5.7 is equal to $Dist^{net}$, so when *net* increases the sampling distribution is kept unchanged in the successive networks.

The major problem of *Adaboost* is that the ensemble tends to be unstable after training some networks because the training sets are overfitted quickly by hard to learn patterns. In contrast, in *Aveboost*, the values of the sampling distribution can not shoot up, or down, quickly because a weighted average is applied to update the sampling distribution. In this case, the update of the sampling distribution tends to be softer and softer as new networks are added to the ensemble.

5.2.2.4 Averaged Boosting version 2

Ninkunj C. Oza proposed *Averaged Boosting version 2*, *Aveboost2*, in [84] one year after describing *Aveboost*. In this variant, some new variables are introduced with respect to the original boosting description. The purpose of the new variables, β^{net} and γ^{net}, are to optimize the training and to improve the specific combiner. It is described in algorithm 5.2.

Algorithm 5.2 Aveboost2 $\{T, V, N_{networks}\}$

Initialize Sampling Distribution $Dist$:
$Dist_x^1 = 1/N_{patterns}\ \forall x \in T$
for $net = 1$ to $N_{networks}$ **do**
Create $T^{'}$ sampling from T using $Dist^{net}$
MF Network Training T' , V
Calculate misclassified vector:

$$miss_x^{net} = \begin{cases} 1 & \text{if } h^{net}(x) \neq d(x) \quad x \text{ is incorrectly classified by } net \\ 0 & \text{otherwise} \quad x \text{ is correctly classified by } net \end{cases}$$

$h^{net}(x)$ is the class associated to the pattern x according to the output of net
$d(x)$ is the target associated to the pattern x
Calculate error ϵ and other measures:
$\epsilon^{net} = \sum_{x=1}^{m} Dist_x^{net} \cdot miss_x^{net}$
$\beta^{net} = \frac{\epsilon^{net}}{1-\epsilon^{net}}$
$\gamma^{net} = \frac{2\cdot(1-\epsilon^{net})\cdot net+1}{2\cdot\epsilon^{net}\cdot net+1}$
Update sampling distribution:

$$w_x^{net} = Dist_x^{net} \cdot \begin{cases} 1 & \text{if } x \text{ incorrectly classified} \\ \beta^{net} & \text{otherwise} \end{cases}$$

$C_x^{net} = \frac{w_x^{net}}{\sum_{i=1}^{N_{networks}} w_x^i}$
$Dist_x^{net+1} = \frac{net\cdot Dist_x^{net}+C_x^{net}}{net+1}$
end for

Aveboost2 uses a specific method to combine the output provided by the networks of the ensemble, the *Aveboost2 combiner*, which is described in equation 5.9.

$$h(x) = \underset{c=1,\ldots,N_{classes}}{\arg\max} \sum_{net:h^{net}(x)=c}^{N_{networks}} \log\left(\gamma^{net} \cdot \frac{1-\epsilon^{net}}{\epsilon^{net}}\right) \tag{5.9}$$

The *Aveboost2 combiner* is similar to the one used in *Adaboost*. However, in this case the vote value given by a network *net* will also depend on the number of networks previously trained. The value of γ^{net} increases as the number of networks added to ensemble increases, and therefore the final vote provided by the network also increases with *net*. In this way, the output provided by the firsts networks of the ensemble are penalized.

5.2.2.5 Totally Corrective Adaboost version 1

Totally Corrective Adaboost version 1, henceforth *TCA v1*, is another variant of *Adaboost*. In this case an optimization procedure is applied to calculate the values of the sampling distribution. Algorithm 5.3 shows the basic design of the ensemble structure whereas algorithm 5.4 shows the optimization procedure applied to update the sampling distribution in *TCAv1*.

Algorithm 5.3 TCAv1 $\{T, V, N_{networks}\}$

Initialize Sampling Distribution $Dist$:
$Dist_x^1 = 1/N_{patterns}\ \forall x \in T$
for $net = 1$ to $N_{networks}$ **do**
Create $T^{'}$ sampling from T using $Dist^{net}$
MF Network Training T' , V
Calculate misclassified vector:

$$u_x^{net} = \begin{cases} -1 & \text{if } h^{net}(x) \neq d(x) \quad x \text{ is incorrectly classified} \\ +1 & \text{otherwise} \quad\quad\quad\quad\ x \text{ is correctly classified} \end{cases}$$

$h^{net}(x)$ is the class associated to the pattern x according to the output of net
$d(x)$ is the target associated to the pattern x
Generate the new sampling distribution: Dist Update TCA $\{Dist, net\}$
end for

As we can see in algorithm 5.3, the basic structure of *TCAv1* is similar to the algorithm described for *Adaboost*. However, in *TCA*, the misclassification vector *miss* is replaced by the **u** vector whose values are now 1 or -1. Moreover, the procedure to update the sampling distribution is more complex because it is not given by a simple equation. In *TCAv1*, the optimization procedure described in algorithm 5.4 is used to obtain the new values of the sampling distribution.

Algorithm 5.4 Dist Update TCAv1 $\{Dist, net\}$

$\widehat{\mathbf{d}}_\mathbf{1} = Dist^1$
$v_1^{net} = \max\limits_{q_j \in \{1,2...,net\}} \left| \widehat{\mathbf{d}}_\mathbf{j} \cdot \mathbf{u}^{\mathbf{q_j}} \right|$
$Converged =$ **false**
$j = 1$
while $not(Converged)$ **do**
$q_j = \arg\max\limits_{q_j \in \{1,2...,net\}} \left| \widehat{\mathbf{d}}_\mathbf{j} \cdot \mathbf{u}^{\mathbf{q_j}} \right|$
$\widehat{\alpha}_j = \ln\left(\frac{1+\widehat{\mathbf{d}}_\mathbf{j} \cdot \mathbf{u}^{\mathbf{q_j}}}{1-\widehat{\mathbf{d}}_\mathbf{j} \cdot \mathbf{u}^{\mathbf{q_j}}}\right)$
$\widehat{\mathbf{d}}_{\mathbf{j+1}} = \widehat{\mathbf{d}}_\mathbf{j} \cdot \exp\left(-\widehat{\alpha}_j \cdot \mathbf{u}^{\mathbf{q_j}}\right)$
$Normalize\left(\widehat{\mathbf{d}}_{\mathbf{j+1}}\right)$
$v_{j+1}^{net} = \max\limits_{q_j \in \{1,2...,net\}} \left| \widehat{\mathbf{d}}_{\mathbf{j+1}} \cdot \mathbf{u}^{\mathbf{q_j}} \right|$
if $\left| v_{j+1}^{net} - v_j^{net} \right| < \Gamma$ **then**
$Converged =$ **true**
$Dist^{net+1} = \widehat{\mathbf{d}}_{\mathbf{j+1}}$
else
$j = j + 1$
end if
end while

In algorithm 5.4, the new distribution, $\widehat{\mathbf{d}}_\mathbf{1}$, corresponds to the initial configuration in which all the patterns have the same probability of being selected, $Dist^1$. Moreover, the value v_1^{net} is also calculated as the maximum value of the multiplication between $\widehat{\mathbf{d}}_\mathbf{1}$ and each individual $\mathbf{u}$ vector.

Then, an iterative procedure is applied to adapt the new distribution until the procedure converges. In each iteration, the new distribution, $\widehat{\mathbf{d}}$, is adapted as follows. Firstly, we select the network, q_j, which provides the highest value of the multiplication of two vectors, the distribution which is being updated, $\widehat{\mathbf{d}}_\mathbf{j}$, and its $\mathbf{u}$ vector ($\mathbf{u}^{\mathbf{q_j}}$). Once the "best" network has been selected, network q_j, (according to the bibliography, q_j has been used to define this "winner" network and the variable for the arg max function), the distribution is updated according to the following equation:

$$\widehat{\mathbf{d}}_{\mathbf{j+1}} = \widehat{\mathbf{d}}_\mathbf{j} \cdot \exp\left(-\widehat{\alpha}_j \cdot \mathbf{u}^{\mathbf{q_j}}\right) \tag{5.10}$$

where:

$$\widehat{\alpha}_j = \ln\left(\frac{1 + \widehat{\mathbf{d}}_\mathbf{j} \cdot \mathbf{u}^{\mathbf{q_j}}}{1 - \widehat{\mathbf{d}}_\mathbf{j} \cdot \mathbf{u}^{\mathbf{q_j}}}\right) \tag{5.11}$$

After updating $\widehat{\mathbf{d}}$, it must be normalized. The summatory of the probabilities of all the patterns must be one according to this ensemble. For this reason, all the individual values of $\widehat{\mathbf{d}}_{\mathbf{j+1}}$ are divided by the summatory of all the values of this vector.

Finally, v_{j+1}^{net} is calculated as the maximum value associated to the multiplication between two vectors, the new distribution $\widehat{\mathbf{d}}_{\mathbf{j+1}}$ and each individual $\mathbf{u}$ vector. The procedure finishes and the new sampling distribution is set ($Dist^{net+1}$) if the difference between v_{j+1}^{net} and its previous value, v_j^{net}, is lower than a threshold, Γ. This stopping criteria was introduced by Oza in [83].

TCAv1 is described and analyzed in [83] as a modified version of the corrective boosting method proposed in [85].

5.2.2.6 Totally Corrective Adaboost version 2

Totally Corrective Adaboost v2, henceforth *TCAv2*, is a slight modified version of *TCAv1* in which the procedure to optimize the sampling distribution also corresponds to algorithm 5.4 but it is slightly modified. Concretely, the following stopping criteria:

if $\left|v_{r,j+1}^{net} - v_{r,j}^{net}\right| < \Gamma$ **then**

Is replaced by:

if $((\left|v_{j+1}^{net} - v_j^{net}\right| < \Gamma)$ *or* $(((j) \geq N_{patterns})$ *and* $(v_{j+1}^{net} > v_j^{net})))$ **then**

In this case, the iterative procedure also finishes if the number of iterations, j, is greater or equal than $N_{patterns}$ and v_{j+1}^{net} is greater than v_j^{net}. This new stopping criteria was also proposed by Oza.

5.2.2.7 Aggressive Boosting

Aggressive Boosting, henceforth *Aggreboost*, is the first of three boosting alternatives proposed by Kuncheva in [82]. According to [82, 86], the equation applied to update sampling distribution of *Aggreboost* is equal to the equation of *Adaboost* described in [87, 88]. However, this new equation introduces the reinitialization of the sampling distribution when the error, ϵ, is not in the range $(0, \ldots, 0.5)$. The description of *Aggreboost* is shown in algorithm 5.5.

Algorithm 5.5 Aggreboost $\{T, V, N_{networks}\}$

Initialize Sampling Distribution $Dist$: $Dist_x^1 = 1/N_{patterns}\ \forall x \in T$
for $net = 1$ to $N_{networks}$ **do**
  Create T' sampling from T using $Dist^{net}$
  MF Network Training T' , V
  Calculate misclassified vector:

$$miss_x^{net} = \begin{cases} 1 & \text{if } h^{net}(x) \neq d(x) \quad x \text{ is incorrectly classified} \\ 0 & \text{otherwise} \quad\quad\quad\quad\ x \text{ is correctly classified} \end{cases}$$

  $h^{net}(x)$ is the class associated to the pattern x according to the output of net
  $d(x)$ is the target associated to the pattern x
  Calculate error and other parameters:
  $\epsilon^{net} = \sum_{x=1}^{m} Dist_x^{net} \cdot miss_x^{net}$
  $\beta^{net} = \sqrt{\frac{1-\epsilon^{net}}{\epsilon^{net}}}, \epsilon^{net} \in (0, 0.5)$
  $\xi_x^{net} = 2 \cdot miss_x^{net} - 1$
  if $\epsilon^{net} \notin (0, 0.5)$ **then**
    Reinitialize Sampling Distribution $Dist$: $Dist_x^{net} = 1/N_{patterns}\ \forall x \in T$
  end if
  Update sampling distribution:
  $Dist_x^{net+1} = Dist_x^{net} \cdot (\beta^{net})^{\xi_x^{net}}$
  Normalize($Dist^{net+1}$)
end for

The variants proposed by Kuncheva use the specific combiner described in equation 5.12, henceforth *Kuncheva's combiner*, which is quite similar to the one used in *Adaboost*. In this case, the square root is used in the equation to calculate the vote values.

$$h(x) = \underset{c=1,\ldots,classes}{\arg\max} \sum_{net:h^{net}(x)=c}^{N_{networks}} \ln \sqrt{\frac{1-\epsilon^{net}}{\epsilon^{net}}} \tag{5.12}$$

5.2.2.8 Conservative Boosting

Conservative Boosting, henceforth *Conserboost*, is the second ensemble proposed by Kuncheva in [82]. *Conserboost* applies a softer equation to update the sampling distribution. In this alternative, the value of the sampling distribution is only updated

for the misclassified patterns. In addition, this variant also allows the reinitialization of the sampling distribution as in *Aggreboost*. The description of *Conserboost* corresponds to algorithm 5.5 but the value of the parameter ξ is given by:

$$\xi_x^{net} = miss_x^{net} \tag{5.13}$$

5.2.2.9 Inverse Boosting

Inverse Boosting, henceforth *Inverboost*, is the last boosting model proposed by Kuncheva in [82]. Opposite to *Adaboost*, *Inverboost* applies an equation to update the sampling distribution in which the hard to learn patterns tend to disappear from the training set. So the different learning sets are specialized on easy to learn patterns. In this variant, the classifiers tend to be more and more similar, decreasing the diversity in the process.

The basic description of *Inverboost* also corresponds to the description of *Aggreboost*, the only difference between them is the value of the parameter ξ which, in this case, is given by:

$$\xi_x^{net} = -miss_x^{net} \tag{5.14}$$

It is though that this variant was due to a missprint in the description of the algorithm described in the references [89, 90]. However, this method is included in our comparison because it can be useful for further experiments and research.

Although the three boosting alternatives proposed by Kuncheva appeared previously in the bibliography [89, 90], most of the time they were simply called *Adaboost* and any distinction among them had not been done. Kuncheva was the first researcher in naming, distinguishing and analising them.

5.2.2.10 Arcing Classifiers

Breiman proposed an ensemble called *ARC-x4* in the paper *Arcing Classifiers* [81]. *ARC-x4* can not be considered as a pure boosting method because the sampling distribution of the i-th network, $Dist_x^i$, only depends on the number of times the pattern, x, has been missclasified by the $i-1$ previous trained networks, equation 5.15. Moreover, the procedure to update the sampling distribution is quite different.

Sometimes, *ARC-x4* is cataloged as a boosting variant because they share the same spirit which is overfitting the training sets with hard to learn patterns.

In this case, the values of the sampling distribution, *probabilities* in the reference, are calculated by equation 5.15 after training a network.

$$Dist_x^{net+1} = \frac{(1 + \sum_{n=1}^{net} miss_x^n)^4}{\sum_{i=1}^{Npatterns} \left(1 + \sum_{n=1}^{net} miss_i^n\right)^4} \tag{5.15}$$

Finally, the *Boosting combiner* was not used to combine the networks in the original reference. Instead, the *Voting* scheme was the combiner applied to obtain the final hypothesis of the ensemble system as in *EVOL*. This combiner was described in subsection 4.2.2.1 and it will be fully analyzed in the next chapter.

5.3 Experimental Setup

The purpose of this comparative research is to obtain the performance of the ensemble methods in which the learning set is modified. As was commented, the main problem of the research in this field is that there is not a common test framework so it is not possible to compare the different alternatives from the results provided in the original references.

Although some problems related to this research were described in the previous chapter, there are specific problems related to the ensemble models included in this chapter. As Kuncheva said in [82], some variants derived from *Boosting* were named with its generic name and a comparison differentiating the may be important.

The main characteristics of the performed research are:

- Nineteen datasets from the *UCI Repository* are employed.
- Twenty different alternatives are applied to generate the ensembles.
- One network architecture is used, the *MF* network.
- Four different ensemble sizes are considered: 3, 9, 20 and 40 networks.
- We have utilized optimized training parameters.
- One combiner is applied for each ensemble:
  - *Output average* in *CVC* versions, *DP*, *OP* and *Boosting3*.
  - Specific combiners in boosting.
  - *Voting* in *ARCx4*.
- The experiments have been repeated ten times with different partitions of training, validation and test sets in order to obtain:
  - The mean value of performance.
  - The error rate by standard error theory.
- Three general measurements were applied for the comparison:
  - Mean *Increase of Performance*.
  - Mean *Percentage of Error Reduction*.
  - Paired Student's t-test.

As mentioned in the previous chapter, the description of the datasets from the *UCI* repository are in appendix A. The optimized training parameters of the networks are in appendix B.3 whereas the specific parameters of the ensembles are in appendix B.5. Finally, the general measurements utilized to compare the ensemble models, mean *IoP* and mean *PER* can be found in equations 3.5 and 3.6 respectively. Although the results and their analysis of this chapter are based on these measurements, the *Paired Student's t-test* will also be used.

5.4 Results and discussion

The main results related to this ensemble comparison are shown in this section. Firstly, table 5.1 shows the mean *IoP* and the mean *PER* of *Simple Ensemble* and the ensemble methods analyzed in this chapter for four different ensemble sizes.

Table 5.1: Ensemble Comparison 2 - General Results

	Mean *IoP*				Mean *PER*			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Simple Ensemble	4.97	5.27	5.34	5.43	21.84	23.65	23.73	24.64
Bagging	5.09	5.78	5.95	5.76	22.93	27.55	28.25	27.69
Bagnoise	3.95	4.03	3.98	4.18	13.32	12.44	13.12	13.66
CVC	4.38	5.53	5.35	5.53	20.66	24.3	24.23	24.21
CVCv2	5.48	5.95	5.98	5.85	25.73	27.09	25.34	23.7
CVCv2.5	5.48	6.17	5.94	5.94	24.25	27.85	26.65	27.55
CVCv3	5.56	5.96	6.17	6.16	24.06	26.59	28.35	28.68
DP	2.83	−2.14	−5.16	−8.15	11.18	−21.36	−51.46	−80.4
DPR	2.93	−1.27	−4.99	−7.89	12.54	−13.11	−42.97	−70.71
OP	3.47	0.12	−4.63	−9.06	13.38	−10.89	−43.14	−87.71
OPR	3.96	1.48	−3.3	−7.67	15.29	−2.16	−33.38	−74.78
Boosting	4.36	–	–	–	16.35	–	–	–
Adaboost	3.69	4.55	4.93	4.98	15.4	19.5	22.96	24.54
Aveboost	4.33	5.57	5.95	6.04	18.26	26.11	27.12	26.53
Aveboost2	4.02	4.59	4.88	5	17.67	22.34	22.99	23.32
TCAv1	2.18	2.88	2.04	0.19	−6.52	−3.97	−7.12	−35.58
TCAv2	3.01	2.63	2.26	1.19	6.59	−1.64	−12.94	−21.21
Aggreboost	3.56	4.79	5.53	5.83	14.82	20.26	25.27	26.56
Inverboost	2.89	0.95	−1.23	−2.76	9.2	−3.12	−15	−24.18
Conserboost	4.42	5.31	5.65	6.06	19.72	25.63	26.62	27.84
ARCx4	4.06	3.78	4.8	5.24	18.01	17.69	22.56	24.37

On the one hand, according to the previous general results, the ensemble alternatives which improve *Simple Ensemble* are: *Bagging*, the whole versions of *Cross-Validation Committee* and a few variants based on *Adaptive Boosting* for the case of a high number of networks in the ensemble from 9 to 40 networks (*Aveboost*, *Aggreboost* and *Conserboost*).

On the other hand, *DP*, *DPR*, *OP*, *OPR* and some *Boosting variants* (*Inverboost*, *TCAv1* and *TCAv2*) perform worse than *Simple Ensemble* and, in most of the cases, perform worse than a single *MF* network.

The results provided by *DP*, *OP* and variants were expected because the size of the training set or the number of different patterns used in training was so small. In the original reference [77], they were tested using two artificial databases. Concretely, the *80-D Correlated Gaussian* with 80 classes from [91] and the *20-class Gaussian* from [69]. In both datasets, each class had a total number of 100 patterns so the total number of patterns used for training was high. Moreover, the authors generated

ensembles of 6 to 9 networks. Although the results provided by the authors were positive, they tested the four methods on artificial datasets with a large number of patterns and using small ensembles. With that experimental setup, all the networks of their ensembles had enough patterns for training. We consider that the general behavior shown in this thesis for the four models is more accurate than in the original references because the experimental setup we have used, with several normal datasets and quite different sizes of the ensembles, is more complete.

Moreover, the results provided by *Inverboost* were also expected because the patterns close to the class boundaries tend to be removed on the successive training sets and it was suggested that this algorithm was due to a missprint [82]. The results provided by both versions of *TCA* were not expected because the optimization procedure used seems to be quite elaborated. We do not know their behavior according to its authors because they did not include any empirical result in the original reference [85]. However, Oza commented in [83] that *TCA* may not perform well.

However, with the general measurements there is not any statistical information about the dissimilarity of *Simple Ensemble* and the best alternatives. For this reason, the *t-test* was applied to compare *Simple Ensemble* to these alternatives described in this chapter in order to test if they statistically improve the results provided by *Simple Ensemble*. The results of the tests are in table 5.2. Moreover, the different versions of *Cross-Validation Committee* are statistically compared among them in table 5.3, whereas the best *Boosting* variants are statistically compared to *Adaboost*, table 5.4.

Table 5.2: Statistical results of the best methods versus *Simple Ensemble*

Methods	measure	3-net	9-net	20-net	40-net
SE vs Bagging	t-**value**	−0.57	−2.64	−4.01	−2.20
	α	0.57	0.0091	0.000086	0.029
SE vs CVCv1	t-**value**	2.66	−1.53	0.16	−0.56
	α	0.0085	0.13	0.87	0.58
SE vs CVCv2	t-**value**	−2.19	−3.71	−3.54	−2.19
	α	0.03	0.00027	0.00051	0.029
SE vs CVCv2.5	t-**value**	−2.22	−4.45	−3.45	−2.61
	α	0.027	0.000014	0.0007	0.01
SE vs CVCv3	t-**value**	−2.20	−3.33	−4.61	−3.88
	α	0.029	0.001	0.0000073	0.00014
SE vs Aveboost	t-**value**	2.48	−1.37	−3.01	−2.78
	α	0.014	0.17	0.0029	0.006
SE vs Aggreboost	t-**value**	4.17	1.53	−0.83	−1.77
	α	0.000047	0.13	0.41	0.078
SE vs Conserboost	t-**value**	1.88	−0.14	−1.33	−2.99
	α	0.061	0.89	0.19	0.0031

In the previous table, *Simple Ensemble*, *SE*, is statistically compared to the best ensemble methods according to the mean IoP and PER of table 5.1. Firstly, *Bagging* and all the versions based on *CVC*, except *CVCv1*, perform better than *Simple Ensembles* and their differences are statistically significant, except for the case of 3-nets and *Bagging*.

Secondly, *Aveboost* and *Conserboost* improve the results provided by *Simple Ensemble* and the differences between them and *Simple Ensemble* are significant for the case of ensembles of 40 networks. In this case, ensembles of 40 networks, *Aggreboost* only performs slightly better than *SE* because α is greater than 0.05. Moreover, the differences between *Aveboost* and *Simple Ensemble* are also statistically significant for ensembles of 20 networks. Finally, the three boosting variants perform statistically worse than *Simple Ensemble* for the case of ensembles of 3 networks.

The *CVC* versions have been statistically compared among them in table 5.3 because each newer version described is an improvement of the previous version. For this reason, it has been considered that any version should be compared to the previous ones.

Table 5.3: Statistical results among *CVC* methods

Methods	measure	3-net	9-net	20-net	40-net
CVCv1 vs CVCv2	t-value	−4.42	−1.95	−3.37	−1.39
	α	0.000016	0.053	0.00092	0.17
CVCv1 vs CVCv2.5	t-value	−4.06	−3.36	−3.37	−2.11
	α	0.000071	0.00095	0.00090	0.036
CVCv1 vs CVCv3	t-value	−3.81	−1.98	−4.48	−3.24
	α	0.00019	0.049	0.000013	0.0014
CVCv2 vs CVCv2.5	t-value	−0.01	−1.43	0.25	−0.51
	α	0.99	0.15	0.80	0.61
CVCv2 vs CVCv3	t-value	−0.37	−0.08	−1.29	−1.99
	α	0.71	0.94	0.20	0.048
CVCv2.5 vs CVCv3	t-value	−0.37	1.23	−1.66	−1.56
	α	0.71	0.22	0.10	0.12

Firstly, *CVCv1* is improved by the other versions of *Cross-Validation Committee*. However, there are two cases in which the differences with respect to *CVCv2* are not statistically significant, ensembles of 9 and 40 networks. These statistical values may be due to the size of the training and validation sets in medium and high sized ensembles generated by *CVCv2*. As we pointed out, their sizes depend on the number of networks of the ensemble.

Secondly, *CVCv2* is slightly improved by *CVCv2.5* and *CVCv3* in most of the cases but the differences are not statistically significant. Minor changes are applied to both ensemble methods with respect to *CVCv2*. Fortunately, ensembles of 40 networks generated by *CVCv3* can improve statistically the results of *CVCv2*. As has been mentioned, diversity is low in *CVCv2* because the specific training sets are similar

and the specific validation sets are nearly insignificant when the number of networks is high. This problem does not occur in *CVCv3* so this last version of *CVC* can provide quite good results in high sized ensembles.

Finally, *CVCv2.5* and *CVCv3* have similar general performance but *CVCv3* performs slightly better. The statistical test shows that the differences between them are not significant. *CVCv2.5* should be considered a *bridge* version between *CVCv2* and *CVCv3* and it should not be seriously considered for further experiments.

In the case of boosting variants, only four models have been included in the statistical comparison. The original *Adaptive Boosting* has been compared to the best boosting variants. Moreover, *Aveboost* has been compared to *Conserboost* because they are the two best boosting alternatives.

Table 5.4: Statistical results among *Boosting* methods

Methods	measure	3-net	9-net	20-net	40-net
Adaboost vs Aveboost	t-**value**	−2.27	−3.27	−3.20	−3.51
	α	0.024	0.0013	0.0016	0.00056
Adaboost vs Aggreboost	t-**value**	−0.39	−1.46	−2.31	−3.20
	α	0.7	0.15	0.022	0.0016
Adaboost vs Conserboost	t-**value**	−2.55	−2.66	−2.58	−3.54
	α	0.012	0.0086	0.011	0.0005
Aveboost vs Conserboost	t-**value**	−0.35	1.66	1.75	−0.09
	α	0.72	0.098	0.081	0.93

The results provided by the statistical tests in table 5.4 show that *Aveboost* and *Conserboost* clearly improve *Adaboost*. Moreover, the dissimilarity of *Conserboost* and *Aveboost* with respect to *Adaboost* increases as new networks are added to the ensemble according to the t-value. Furthermore, *Aggreboost* is only slightly better than *Adaboost* for low and medium sized ensembles, 3 or 9 networks in the ensemble, because the differences are not statistically significant. The probability of reinitializing the sampling distribution in low and medium sized ensembles is very low so *Adaboost* and *Aggreboost* have both, in general, the same behavior on these cases and the difference between them becomes clear for 20 and 40 networks in the ensemble. In these two cases, *Aggreboost* is better than *Adaboost* and α denotes that the differences between them are statistically significant.

Finally, *Aveboost* performs better than *Conserboost* in two cases and in the other cases *Conserboost* performs better. The statistical test shows that the differences between them are not significant. But in ensembles of 9 and 20 networks, the differences are almost significant because α is slightly higher than 0.05. Although these two methods provide similar results this does not mean that they classify exactly the same patterns.

5.5 Conclusions

Some interesting information can be derived from the results shown in the previous tables. First of all, considering the best performing methods, the mean increase of performance with respect to a single network is higher than 6% and the percentage of error reduction is between 28% and 29%. The best results obtained here are slightly higher that the best results shown in the previous chapter.

Secondly, the best ensemble version of *Cross-Validation* is *CVCv3* and it has been proposed by us. In *CVCv3*, the number of partitions does not depend on the number of networks and, moreover, the networks do not share the same validation set. So the size of training and validation sets are kept in a reasonable value and the diversity of the ensemble is improved. Although *CVCv2.5* has also good performance, the ensembles of 40 networks generated by *CVCv3* are the only ensembles which statistically improve the results provided by the traditional models based on *Cross-Validation* which are commonly applied in the literature, *CVCv1* and *CVCv2*. It is important because, as was mentioned before, this concrete implementation, *CVCv3*, has been proposed in this thesis. Furthermore, *CVCv2* and *CVCv2.5* also provides good results and their differences with respect to *Simple Ensemble* are statistically significant.

Thirdly, the *Boosting* variants are accurate. *Conserboost* and *Aveboost* provide the best results in this class of ensembles whereas *TCA* and *Inverboost* provide the worst ones. *Conserboost* and *Aveboost* are two different approaches in which the sampling distribution is updated by successfully applying a softer equation. The results of *Inverboost* were expected because hard to learn patterns disappear from training and Kuncheva described it as a *missprint*. Although the optimization procedure to update the sampling distribution proposed in *TCA* is elaborated, the results show that *TCAv1* and *TCAv2* should not be considered. Unfortunaltely, *TCA* was not empirically tested in the original reference so we do not know its behavior on the experiments performed by the researchers if they were done. Furthermore, Oza reached similar conclusions about *TCA* in [83].

Moreover, *Bagging* also provides good results and it is better than *Simple Ensemble* in all cases. Furthermore, *Bagging* performs better than *Boosting* in a wide number of cases. However, the best results provided by *Bagging* are overcame by *Conserboost* in the case of 40 networks in the ensemble. Unfortunately, *Bagnoise* does not improve the results obtained by *Simple Ensemble*.

In addition, the implementations based on disjoint and overlapping partitions do not provide good results. The size of the specific training sets inversely depends on the number of networks of the ensemble so the performance of the individual networks in the ensemble usually decreases with the size of the training set. The size of the specific training sets is much lower and there are not enough samples to successfully train a network. In the original references, they provided good results because they were tested on large datasets and they used 'small' ensembles. So, they had enough

patterns to successfully train all the networks of the ensemble. However, there are some classification problems in which a large dataset can not be generated and, therefore, *DP*, *DPR*, *OP* and *OPR* are not suggested to be applied. For this reason, we consider that the general behavior shown in this thesis is more accurate for these four ensembles.

Finally, the best ensemble methods are: *Bagging*, *CVCv3*, *Aveboost*, *Conserboost*. These four alternatives should be seriously considered when ensembles of neural networks are designed. However, *Bagging* is special because the training set is doubled so the computational training requirements are also doubled. The ratio between performance and requirements of *Bagging* should be seriously considered when the ensembles are being designed. Finally, the origins of the four best models are quite different so it may be interesting to design new methods in which some different ensemble approaches are combined. Although diversity is generated in all of these alternatives by modifying the learning sets, the modifications applied are quite different among them.

To conclude this chapter, is important to remark that the best ensemble alternatives according to the two ensemble comparisons done in this thesis are should be strongly considered. *Bagging*, *CVCv3*, *Aveboost* and *Conserboost* are the best methods according to the current research and both versions of *Decorrelated* were highly suggested in the previous chapter. They should be used to solve classification problems because they provide excellent general results and their origins, or sources of diversity, are quite different among them. For instance, *CVCv3* uses slightly different training and validation sets, *Conserboost* overfits softly the networks with hard to learn patterns and *Decorrelated* modifies the target equation in order to reduce the correlation between networks.

The results shown in this chapter have also been published in references [MFC1, MFC2, MFC3] which are detailed in the last chapter of this thesis.

Chapter 6

A comparative study of ensemble combiners

6.1 Introduction

In this chapter, some methods to combine the networks of an ensemble will be analyzed and compared. Concretely, the research performed here is focused on combiners applied to ensembles previously trained with *Simple Ensemble*, *Decorrelated*, *Cross Validation Committee v2* and *Conservative Boosting*. In this way, four different ensemble alternatives have been used to test the different combiners analyzed in this chapter. We have selected them because they are the most representative ones. Although *Cross Validation Committee v2* is outperformed by *Cross Validation Committee v3*, we have chosen *CVCv2* because it was the ensemble used in the first combiner study published in [76]. Moreover, *Simple Ensembles* of *RBF* networks are also studied. The comprehension and analysis of this chapter would have been much more difficult if we were included more ensemble alternatives in the comparison.

The current research is organized as follows. Firstly, the studied combiners will be reviewed in section 6.2. Then, the setup of the experiments is described in section 6.3, whereas their discussion is shown in section 6.4.

6.2 Analysed Combiners

The combination of the information provided by the networks of an ensemble is an important step in the process to build the final classifier [92, 93, 76]. Selecting the most appropriate way to fuse the outputs can highly improve the performance of the system. Although the most commonly applied alternative is *Output average*, there are some important models which will be studied in this chapter.

Since there are some different approaches to combine the networks, we have grouped them into the following categories in this thesis:

- *Combiners based on averaging.*
- *Voting schemes.*
- *Competitive combiners.*
- *Other combiners.*
- *Feature Based Combiners.*
- *Two-Layered MCS.*

The models based on averaging are those in which a simple average is applied in order to calculate an averaged output vector, $\widehat{y}$, or ensemble hypothesis, $h(\widehat{y}(x))$ or $Class(\widehat{y}(x))$, without using any previous knowledge on the dataset. The voting schemes apply a voting procedure to select the most appropriate class, the class with highest number of votes is assigned to the pattern.

The competitive alternatives usually use prior knowledge on the dataset in order to select the most relevant networks for a determined pattern whereas a small weight is assigned to the less relevant networks in the final hypothesis. Moreover, there are

other ways to fuse the networks which are based on more complex operators such as the *Choquet Integral* or the *Zimmermann Compensatory Operator*. Furthermore, we will analyze two combiners which depend on the data (pattern) which is being classified.

Finally, some two layered *MCS*, such as *Mixture of experts* [94] or *Stacked Generalization* [95], can be adapted to combine ensembles if the networks of the ensemble are "assigned" to the "first *layer*" of these systems and they are kept unchanged during the training procedure.

6.2.1 Combiners based on averaging

The alternatives based on averaging are those in which a simple average is calculated for each output of the networks without using any knowledge on the dataset. The main problem of this approach is that the best and the worst networks have the same influence on the final decision.

6.2.1.1 Output average

This approach, *Output average* or *Ensemble averaging*, simply averages the individual outputs across the different classifiers. The final output is given by:

$$\widehat{y}_c(x) = \overline{y}_c(x) = \frac{1}{N_{nets}} \cdot \sum_{net=1}^{N_{nets}} y_c^{net}(x) \tag{6.1}$$

Then, the class, c, yielding the maximum of the averaged values, $\widehat{y}_c$, is assigned to the pattern.

$$Class(x) = \underset{c=1,\ldots,N_{classes}}{\arg\max} \; (\widehat{y}_c(x)) \tag{6.2}$$

Where *arg max* returns the argument, class c, with highest output value $\widehat{y}_c(x)$.

6.2.1.2 Dynamically Averaged Networks version 1

Dynamically Averaged Networks version 1, henceforth *DANv1*, was proposed by Jimenez [96]. In this case, a weighted average is applied in order to obtain the output vector of the ensemble. To perform this special average, the weights are proportional to the certainties of the respective network output.

$$\widehat{y}_c(x) = \overline{y}_c(x) = \frac{1}{N_{nets}} \cdot \sum_{net=1}^{N_{nets}} w_c^{net} \cdot y^{net}(x) \tag{6.3}$$

Where the values of the weights, w, are given by:

$$w_c^{net} = \frac{C_c^{net}}{\sum_{i=1}^{N_{nets}} C_c^i} \tag{6.4}$$

$$C_c^{net} = \begin{cases} y_c^{net}(x) & if \; y_c^{net}(x) \geq 0.5 \\ 1 - y_c^{net}(x) & otherwise \end{cases} \tag{6.5}$$

We may notice in the previous equations that the weights depend on the output values. When an output of a class, c, is close to 0 the probability that the pattern, x, does not belong to that class is the highest one. If the same output is close to 1, the probability that the pattern belongs to that class is also high. Finally, when an output is close to 0.5, the probability that the pattern belongs or not to the class c are similar and close to 50%.

The author proposes that the certainties of the outputs are given by the outputs themselves. The certainty of a specific output, c, for a given network, net, is C_c^{net} and it reaches the highest value when the output, y_c^{net}, is close to 0 or 1 whereas the lowest certainty value, 0.5, is given when the output is also 0.5.

As in *Output average*, the class, c, yielding the maximum of the averaged values, $\widehat{y}_c$, is assigned to the pattern.

6.2.1.3 Dynamically Averaged Networks version 2

Dynamically Averaged Networks version 2, henceforth *DANv2*, is an evolution of *DANv1* which was also proposed by Jimenez [97]. Although, the special average previously described is also applied, the weights are now given by:

$$w_c^{net} = \frac{\exp\left(-\gamma \cdot (C_c^{net})^2\right)}{\sum_{i=1}^{N_{nets}} \exp\left(-\gamma \cdot (C_c^i)^2\right)} \tag{6.6}$$

Where γ depends on the problem and it has to be set by trial and error. Moreover, the second difference with respect to *DANv1* is that the weights do not linearly depend on the output values because the certainties are squared and the exponential function is also introduced into the equation.

6.2.2 Voting schemes

In this subsection, the voting schemes will be reviewed. The combiners analyzed are quite different, but the use of a counter is the philosophy shared by them.

6.2.2.1 Majority Voting

In *Majority voting* each classifier, net, provides a vote to a class, c, given by the highest output of the classifier. This approach will be named *Voting* henceforth.

$$vote_c^{net}(x) = \begin{cases} 1 & \text{if } y_c^{net} \text{ is the highest value in } y^{net} \\ 0 & \text{otherwise} \end{cases} \tag{6.7}$$

Then, the summatory of all the votes for class c is calculated according to the following equation:

$$v_c(x) = \sum_{net=1}^{N_{nets}} vote_c^{net}(x) \tag{6.8}$$

Finally, the class which has been most often voted by the classifiers, the one with highest $v_c(x)$, is assigned to the pattern, x, as in equation 6.9.

$$Class(x) = \underset{c=1,\ldots,N_{classes}}{\arg\max} \; (v_c(x)) \tag{6.9}$$

The main problem of *Voting* is that the information provided by the network is reduced to a single vote so the probabilistic information related to each output is omitted.

6.2.2.2 Nash Vote

In *Nash Vote* [98], henceforth *Nash*, each classifier, *net*, assigns a real-valued vote between zero and one for each candidate output or class, c. The vote value, $vote_c^{net}(x)$, is given by the value of the output.

$$vote_c^{net}(x) = y_c^{net}(x) \tag{6.10}$$

The product is applied to all the votes in order to calculate the final output or *nash value*.

$$nash_c(x) = \prod_{net=1}^{N_{nets}} vote_c^{net}(x) \tag{6.11}$$

Finally, the class c which has the highest *nash value*, $nash_c(x)$, is assigned to the pattern, x, as shown in equation 6.12

$$Class(x) = \underset{c=1,\ldots,N_{classes}}{\arg\max} \; (nash_c(x)) \tag{6.12}$$

The main problem of *Nash* is that a single network with low performance can drastically alter the output of the ensemble because of the use of the product. For instance, if a classifier assigns the value 0 to a class, the *nash* value of this class is also 0 even if the other classifiers assign the value 1 to the same class.

Although the previous example can be possible, the probability of having the worst case is very low. The *Nash vote* can be seen as a voting scheme with censor votes.

6.2.2.3 Borda Count

Borda Count [99, 76], henceforth *Borda*, is a ranking-based combiner. In this case, the classes are ranked in descending order according to the output values. The first element of the ranked list has the highest output value and the lowest value is set to the last element.

Each network of the ensemble, *net*, assigns to a class, c, as many votes as the number of classes are ranked below c by the network. For a given class, c, its number of votes

corresponds to the number of classes whose output value is lower than the output value of class c. For example, a network with 5 different outputs will give to the class with highest output, the *borda count* of 4.

Firstly, equation 6.13 is used to determine if a class j is ranked below another class i in network net.

$$v_{i,j}^{net}(x) = \begin{cases} 1 & \text{if } y_i^{net}(x) \text{ is greater than } y_j^{net}(x) \text{ in network } net \\ 0 & \text{otherwise} \end{cases} \tag{6.13}$$

Then, the individual *Borda* values, B_c^{net}, are calculated with equation 6.14. Each value denotes the number of classes ranked below a class, c, by a network, net.

$$B_c^{net}(x) = \sum_{\substack{i=1 \\ i \neq c}}^{N_{classes}} \left(v_{c,i}^{net} \right) \tag{6.14}$$

Once all these values are calculated, the *Borda Count* related to the ensemble, $Borda$, is calculated with equation 6.15. This final count is the summatory of the individual values of a class, c, for all the networks in the ensemble.

$$Borda_c(x) = \sum_{net=1}^{N_{nets}} \left(B_c^{net}(x) \right) \tag{6.15}$$

The class, c, with highest count, $Borda_c(x)$, is assigned to the pattern as given in the following equation:

$$Class(x) = \underset{c=1,\ldots,N_{classes}}{\arg\max} \left(Borda_c(x) \right) \tag{6.16}$$

6.2.3 Competitive combiners

In this subsection some competitive alternatives will be described. They are considered *competitive* because a "switch control" is applied to select the best network for each case. Moreover, external information can be used to prior the networks.

6.2.3.1 Winner Takes All

In *Winner Takes All*, henceforth *WTA*, the class with overall maximum output across all classifiers and outputs is selected as the correct class, it is calculated with equation 6.17.

$$Class(x) = \underset{\substack{net=1,\ldots,N_{classes} \\ c=1,\ldots,N_{classes}}}{\arg\max} y_c^{net} \tag{6.17}$$

Here, the final output vector only depends on the network which provides the maximum overall output and the information provided by the other networks is omitted. The aim of this method is to identify the most appropriate network by the highest output and switch control to it.

6.2.3.2 Weighted average based on Correlation Matrix

In this approach, henceforth *W.Ave*, some weight values are introduced to the outputs of the different networks prior to averaging. The weights try to minimize the difference between the output of the ensemble and the *desired or true* output (target) on patterns from the learning set. The weights can be estimated from the error correlation matrix, equation 6.18. Each value of the matrix, $C_{i,j}$ denotes the correlation of the error between two networks, i and j.

$$C_{i,j} = \frac{1}{N_{patterns}} \cdot \sum_{x=1}^{Npatterns} \left(\left(y^i(x) - d(x) \right) \cdot \left(y^j(x) - d(x) \right) \right) \tag{6.18}$$

Then equation 6.19 is used to calculate the weights, w, of the special average. The weights depend on the inverse of the correlation matrix, denoted with IC on the equations.

$$w^{net} = \frac{\sum_{n=1}^{N_{nets}} IC_{net,n}}{\sum_{m=1}^{N_{nets}} \sum_{n=1}^{N_{nets}} IC_{m,n}} \tag{6.19}$$

Finally, the final output vector for a pattern, x, according to this weighted average, $\widehat{y}(x)$, is given by the following equation:

$$\widehat{y}(x) = \sum_{net=1}^{N_{nets}} \left(w^{net} \cdot y^{net} \right) \tag{6.20}$$

According to the original references, in all the previous equations, it has been supposed that the networks have only one output. A complete specification with another index term, the output class, can induce to errors so a simpler specification was published in the original reference.

In the classification problems used in this thesis, there are as many outputs as classes have the database. To calculate the weighted average for a classification problem with $N_{classes}$, there will be a correlation matrix, C^c, and its inverse, IC^c, for each class, c. In the equations we refer to the inverse of the correlation matrix as IC instead of C^{-1} because adding the superindex class, c, to C^{-1} could have induced to error.

Therefore, there will also be a specific weight value, w_c^{net}, for each network *net* and class c. Finally, the single output, $\widehat{y}(x)$, will be a vector of $N_{classes}$ elements, $\widehat{y}_c(x)$.

$$\widehat{y}_c(x) = \sum_{net=1}^{N_{nets}} \left(w_c^{net} \cdot y_c^{net} \right) \tag{6.21}$$

The class with highest final value, $\widehat{y}_c(x)$, is assigned to the pattern as given in the following equation:

$$Class(x) = \underset{c=1,\ldots,N_{classes}}{\arg\max} \left(\widehat{y}_c(x) \right) \tag{6.22}$$

The full description of *Weighted Average* can be found in [73, 98, 76]. It is considered competitive because the information provided by the worst networks are nearly omitted, they have low weights in the final output, whereas the information of the best performing networks have high weights on the final output. Prior knowledge, the error of the patterns from the training set, is used to set these weights.

6.2.3.3 Bayesian Combination

Bayesian Combination, henceforth *Bayes*, was originally proposed in [100]. According to the original reference it is based on the *Belief value*. This value is the normalized probability that the pattern x belongs to class c. Equation 6.23 is used to calculate its value.

$$Belief_c(x) = \frac{\prod_{net=1}^{N_{nets}} p\left(x \in c | class(y^{net}(x)) = j^{net}\right)}{\sum_{i=1}^{N_{classes}} \prod_{net=1}^{N_{nets}} p\left(x \in i | class(y^{net}(x)) = j^{net}\right)} \tag{6.23}$$

Where $p\left(x \in c | class(y^{net}(x)) = j^{net}\right)$ is the probability that the pattern x belongs to a class, c, when the output provided by the network *net* predicts the pattern as class j^{net}. This value is calculated with the normalized value from the confusion matrix C in reference [101].

$$p\left(x \in c | class(y^{net}(x)) = j\right) = \frac{C_{c,j}^{net}}{\sum_{i=1}^{N_{classes}} C_{i,j}^{net}} \tag{6.24}$$

In this combiner, C is the *confusion matrix*, it is not the *correlation matrix* described in *Weighted Average*. The *confusion matrix* is calculated by equation 6.25:

$$C_{i,j}^{net} = \sum_{l=1}^{N_{patterns}} v_{i,j}^{net}(l) \tag{6.25}$$

Where:

$$v_{i,j}^{net}(l) = \begin{cases} 1 & \text{if } Class(y^{net}(l)) = i \text{ and } Class(d(l)) = j \\ 0 & otherwise \end{cases} \tag{6.26}$$

The confusion matrix of a network, $C_{i,j}^{net}$, denotes the total number of patterns from the training set which have been cataloged to belong to the class i but they belong to the class j according to the *desired output*, d, or *target*.

Finally, the class, c, with highest value, $Belief_c(x)$ is assigned to the pattern as shown in equation 6.27.

$$Class(x) = \underset{c=1,\ldots,N_{classes}}{\arg\max} \; (Belief_c(x)) \tag{6.27}$$

This combiner is also competitive because the *belief values* associated to a network depend on how the networks classify patterns from the training set. In this combiner, the control is switched to those networks which can classify better a concrete pattern and the "worst" networks are nearly omitted. To perform this task, prior knowledge on the training set is used.

6.2.4 Other combiners

In this section, some *complex* alternatives which do not belong to the previous groups are introduced. These combiners are based on complex procedures which are not trivial. Moreover, some concepts such as the *Choquet Integral* or the *Zimmermann's Compensatory Operator* are analyzed to describe better the new combiners.

6.2.4.1 Choquet Integral

Choquet Integral, henceforth *Choquet*, was proposed in [102, 103, 104] and is based on Sugeno's *k-fuzzy measure* [105]. With this model, the class, c, with highest *Choquet Integral* value (Cg_c) is assigned to the pattern, x, as in equation 6.28.

$$Class(x) = \underset{c=1,2,\ldots,N_{classes}}{\arg\max} \; Cg_c(x) \tag{6.28}$$

The fuzzy densities described in equation 6.29 and the following two definitions are required in order to calculate the values of the integral Cg.

$$g_{net} = d_s \cdot \frac{p_{net}}{\sum_{n=1}^{N_{nets}} p_n} \tag{6.29}$$

Where p_{net} is the percentage of correctly classified patterns on the validation set. Moreover, the sum of the densities should correspond to a desired sum d_s. This last parameter has to be set by a trial and error procedure. Finally, these calculated densities are important because they represent the degree of importance of the networks in the final classification.

First Definition:

Let $g : 2^R \rightarrow [0, \ldots, 1]$ be defined as density function:

1. $g(0) = 0$ and $g(N_{nets}) = 1$
2. if $A, B \subset 2^{N_{nets}}$ and $A \subset B \rightarrow g(A) \leq g(B)$
3. if $A_n \subset 2^{N_{nets}} / \forall n \ 1 \leq n < \infty$ and $\{A_n\}$ is monotonous
   - $\lim_{n\rightarrow\infty} g(A_n) = g\left(\lim_{n\rightarrow\infty} A_n\right)$

Generally, the density value related to the union of the two sets can not be directly calculated and the following equation should be applied to obtain these densities.

$$g(A \cup B) = g(A) + g(B) + \lambda \cdot g(A) \cdot g(B) \quad (6.30)$$
$$\forall A, B \subset \Re / A \cap B = 0 \text{ and } \lambda > -1$$

Let $R = \{net_1, \ldots, net_{N_{nets}}\}$ be a finite set composed by the neural networks of the ensemble, and $g_i = g(\{net_i\})$ is the density value of network calculated by:

$$g(A_1) = g(\{net_1\}) = g_1 \quad (6.31)$$
$$g(A_i) = g_i + g(A_{i-1}) + \lambda \cdot g_i \cdot g(A_{i-1}), \forall i / 1 < i < N_{nets} \quad (6.32)$$

The λ value is calculated by solving equation 6.33. The final λ value must be different to 0 and be in the range $[-1, \ldots, \infty]$.

$$\lambda + 1 = \prod_{net=1}^{N_{nets}} (1 + \lambda \cdot g_{net}) \quad (6.33)$$

In the first definition, the equations used to calculate the values of $g(A_i)$ have been described. These values are used in the second theorem to calculate the final value of Cg. As in *Weighted average*, it has been supposed that the networks have only one output because the description for the n class version is more complex and may induce to error.

Second Definition:

Let define the *discrete Choquet Integral* as:

$$Cg(x) = Cg\{h(net_1), \ldots, h(net_{N_{nets}})\} = \sum_{i=1}^{N_{nets}} \{h(net_i) - h(net_{i-1})\} \cdot g(A_i) \quad (6.34)$$

Being:

$$A_i = \{net_1, \ldots, net_i\} \quad (6.35)$$
$$h(net_0) = 0 \quad (6.36)$$

For this reason:

$$0 \leq h(net_1) \leq \ldots \leq h(net_{N_{nets}}) \leq 1 \quad (6.37)$$

In the previous equations, $h(net_i)$ corresponds to the output of the network, $y^i(x)$.

In order to combine an ensemble in which each network provides an output vector of $N_{classes}$ elements. The previous equations will be also used to calculate the *Choquet Integral* $Cg_c(x)$ for each class, c, but the value of $h(net_i)$ will be given by $y_c^i(x)$.

With the description of the fuzzy densities, g in 6.29, and the two previous definitions we can calculate the *Choquet Integral* and, therefore, implement the combiner. However, this procedure is complex. For this reason, we introduce the following algorithm in which each step is described.

Algorithm 6.1 Choquet Inegral

Calculate the performance of all the networks: p_{net}
Calculate the fuzzy densities of all the networks with equation 6.29: g_{net}
Calculate the roots, λ, of equation 6.33
Calculate $g(A_{net})$ for all the networks, net, of the ensemble (eq.6.31 and 6.32)
Calculate the *Choquet Integral* for each class, c, as in equation 6.34

According to the algorithm, the performance (as percentage of correctly classified patterns on the validation set) and the fuzzy densities are calculated. Their values are stored in p_{net} and g_{net} respectively.

Then, the λ value is calculated with equation 6.33. The degree of this equation corresponds to the number of networks in the ensembles, N_{nets}, because of the product in the equation (the product of $(1 + \lambda \cdot g_{net})$ is repeated N_{nets} times). Although we can get some different roots for this equation, only one is valid for the combiner. This value must accomplish the restrictions shown in the first definition. In the experiments we have always obtained only one valid λ value because the *desired sum*, d_s in equation 6.29, has been properly set. After some experiments, we set the value of d_s with equation 6.38 as we describe in the appendix B.6.

$$d_s = 0.95 \cdot \frac{\sum_{net=1}^{N_{nets}} p_{net}}{\max_{net=1}^{N_{nets}} p_{net}} \tag{6.38}$$

Once we obtain a valid λ, the values of $g(A_i)$ are directly calculated as in equations 6.31 and 6.32. At this point, the fuzzy densities, g_{net}, must be differentiated from the fuzzy measurement, $g(A_i)$ or $g(\{net_i\})$. Although they do not represent the same, they have a common name, g, in the original references.

Finally, the *Choquet Integral* is calculated for each class, c, with the values of $g(A_i)$ and the outputs of the networks, $h(net_i)$ are really $y_c^i(x)$. However, the index vector net_i has been permuted to accomplish with restriction shown in equation 6.37. All the networks are sorted in ascending order according to the output values. The first network has the lowest output value and the last one has the highest value.

Some concepts related to this combiner and the previous definitions can be found in [102, 103, 106].

6.2.4.2 BADD Defuzzyfication Strategy

BADD Defuzzyfication Strategy, or simply *BADD*, is a weighted average system based on the *Basic Defuzzification Distribution* strategy described in [107]. In this case, the training set of a network, T^{net}, is represented by a reference point v_{net}. To calculate this reference point, the procedure described in algorithm 6.2 has been used.

Algorithm 6.2 Calculate reference point of the training set { *net* , T^{net} }

```
Initialize the reference point v_net(0) with small values
for ite = 1 to N_ite do
  for x = 1 to N_patterns do
    v^net = v^net + c_ite · [x − v^net]
  end for
end for
```

In the previous algorithm, the initial reference point of a network, v^{net} is randomly set with small values close to 0. Then, the patterns, x, from the training set, T^{net} are used to adapt it for some iterations. Each adaptation makes the reference point to approach to the pattern, x, but it will never exactly correspond to the pattern, x, because a slowly decreasing sequence of learning coefficients, c_{ite}, is used to weight the update. The values of c_{ite} are between 0 and 1, and they decrease as the index increases. In this thesis we have used the following equation to calculate this decreasing sequence.

$$c_{ite} = 1 - 0.9 \cdot \frac{ite}{N_{ite}} \tag{6.39}$$

The importance of a network, *net*, in the final classification depends on the distance between the pattern which is being classified, x, and its euclidean distance to the reference point, v_{net}. Equation 6.40 is used to calculate $\mu^{net}(x)$ which represents the membership degree of the pattern, x, with respect to the specific training set of the network, *net* .

It is important to mention that two networks with have used the same training set can achieve the same reference point. For instance, this behavior occurs for *Simple Ensemble* because the same base training and validation sets are shared by all the network of the ensemble. In *Cross-Validation Committee version 2*, each network has specific training and validation sets so the reference points are slightly different among them.

$$\mu^{net}(x) = \frac{1}{1 + (d(x, v_{net}))^P} \tag{6.40}$$

In the last equation, P is a constant and $d(x, v_{net})$ denotes the euclidean distance between x and v_{net}. In equation 6.41, the parameter δ determines the type of defuzzification applied. These parameters, P and δ, were set, respectively, to 2 and 3 as it was done in the literature.

Then, equation 6.41 is used to obtain the final output.

$$\widehat{y}(x) = \frac{\sum_{net=1}^{N_{nets}} \left((\mu^{net}(x))^{\delta} \cdot y^{net}(x) \right)}{\sum_{net=1}^{N_{nets}} (\mu^{net}(x))^{\delta}} \tag{6.41}$$

Finally, the class yielding the highest final value, $\widehat{y}_c(x)$, is assigned to the pattern as in the other alternatives based on averaging (for instance, as in equation 6.22 of *Weighted Average*).

6.2.4.3 Combination by Zimmermann's Compensatory Operator

This alternative is based in the *Zimmermann's compensatory operator* described in [108]. The output of the ensemble is given by the following equation

$$\widehat{y}_c(x) = \left(\prod_{net=1}^{N_{nets}} \left(y_c^{net}(x) \right)^{w_c^{net}} \right)^{1-\gamma_c} \cdot \left(1 - \prod_{net=1}^{N_{nets}} \left(1 - y_c^{net}(x) \right)^{w_c^{net}} \right)^{\gamma_c} \tag{6.42}$$

The parameter γ represents the degree of compensation between the union and intersection parts of the operator whereas w represents the weight value associated to the networks. An optimization procedure is applied to set both parameters but there are two constraints to consider in this process:

$$\sum_{n=1}^{N_{nets}} w^n = N_{nets} \tag{6.43}$$

$$\gamma \in [0, 1] \tag{6.44}$$

However, the previous constraints can be avoided if equations 6.45 and 6.46 are applied to calculate both parameters as described in [109]. In this way, two variables a and b must be adapted along with a vector d with N_{nets} elements. They must be calculated and adjusted for each output class c. The advantage of use the new variables instead of directly calculating the weights and γ is that they are not limited by any constraint so the procedure to adapt them can be easily done.

$$w_c^{net} = N_{nets} \cdot \frac{(d_c^{net})^2}{\sum_{n=1}^{N_{nets}} (d_c^n)^2} \tag{6.45}$$

$$\gamma_c = \frac{a_c^2}{a_c^2 + b_c^2} \tag{6.46}$$

The value of the parameters a_c, b_c and d_c should be determined by an optimization procedure. In this thesis, a gradient-descent algorithm has been applied to set these parameters without considering any constraint as suggested in [76]. Moreover, this algorithm is similar to *Backpropagation* because it is an iterative procedure in which all the patterns from the training set are presented to the combiner and the parameters are adjusted to minimize an error function. The basic description of this procedure is in algorithm 6.3.

Algorithm 6.3 Determining Zimmerman parameters { a , b , d }

```
Randomly set initial values of a and b in range described in (6.47)
for net = 1 to N_nets do
  Set initial values d^net according to network performance (equation 6.48)
end for
for i = 1 to N_ite do
  for x = 1 to N_patterns do
    for c = 1 to N_classes do
      Adjust the value of the parameter a_c with equations 6.50 and 6.55
      Adjust the value of the parameter b_c with equations 6.50 and 6.56
      for net = 1 to N_nets do
        Adjust the value of the parameter d_c^net with equations 6.50 and 6.57
      end for
    end for
  end for
end for
Store final Zimmerman parameters values
```

Before starting the iterative procedure, a_c and b_c parameters are initialized with small values close to 0. The initial values of, d_c^{net}, corresponds to the performance of the network on the validation set, the performance is the percentage of correctly classified patterns.

$$a_c, b_c \in [-0.05, 0.05] \tag{6.47}$$

$$d_c^{net} = perf_{net}^{val} \tag{6.48}$$

Then the iterative procedure is repeated for some iterations. In each iteration, all the patterns from the training set are presented to the combiner. In each presentation, all the parameters are adjusted according to the equations from 6.50 to 6.57. A minimization procedure has been applied to *optimize* these parameters. Concretely, the *Mean Square Error* is minimized as we did in *Backpropagation*. This equation is reproduced here in equation 6.49.

$$error_c(x) = (\widehat{y}_c(x) - d_c(x))^2 \tag{6.49}$$

Where $\widehat{y}_c(x)$ is the output of pattern x for class c provided by the *Zimmermann's operator*, equation 6.42, and $d(x)$ is its *desired output vector* or *target*.

The values of a_c, b_c and d_c^{net} are adapted according to the following equation.

$$param_c(t+1) = param_c(t) + step \cdot \frac{\partial error_c(x)}{\partial param_c} \tag{6.50}$$

Where *param* refers to the parameter which is being adapted and *step* is the adaptation step. In this thesis, this step has been set to 0.1 after a trial and error procedure. Moreover, t refers to the current value of the parameter and $t+1$ denotes the same parameter after adjusting it.

Before showing the final derivatives for each parameter, we introduce the following equations in order to simplify their description:

$$P1_c = \prod_{net=1}^{N_{nets}} \left(y_c^{net}(x)^{w_c^{net}} \right) \tag{6.51}$$

$$P2_c = \prod_{net=1}^{N_{nets}} \left(1 - y_c^{net}(x)^{w_c^{net}} \right) \tag{6.52}$$

And:

$$Part1_c = (P1)^{1-\gamma_c} \tag{6.53}$$

$$Part2_c = (1 - P2)^{\gamma_c} \tag{6.54}$$

In the previous equations, $Part1_c$ and $Part2_c$ are the first and second part of the equation 6.42. As we mentioned, they are only used to simplify the final equations.

Finally, the final derivatives used to adjust the parameters are given by the following equations:

$$\frac{\partial error_c(x)}{\partial a_c} = 2 \cdot (y_c(x) - d_c(x)) \cdot y_c(x) \cdot \frac{2 \cdot a_c \cdot b_c^2}{(a_c^2 + b_c^2)^2} \cdot \cdot (\ln(P1_c) \cdot \ln(1 - P2_c)) \tag{6.55}$$

$$\frac{\partial error_c(x)}{\partial b_c} = 2 \cdot (y_c(x) - d_c(x)) \cdot y_c(x) \cdot \frac{2 \cdot a_c^2 \cdot b_c}{(a_c^2 + b_c^2)^2} \cdot \cdot (\ln(1 - P2_c) \cdot \ln(P1_c)) \tag{6.56}$$

$$\frac{\partial error_c(x)}{\partial d_c^{net}} = 2 \cdot (y_c(x) - d_c(x)) \cdot \cdot \left(\frac{\partial Part1_c(x)}{\partial d_c^{net}} \cdot Part2_c + Part1_c \cdot \frac{\partial Part2_c(x)}{\partial d_c^{net}} \right) \tag{6.57}$$

Where:

$$\frac{\partial Part1_c(x)}{\partial d_c^{net}} = (1-\gamma)\cdot(P1_c)^{1-\gamma}\cdot\frac{2\cdot N_{nets}\cdot d_c^{net}}{\left(\sum_{n=1}^{N_{nets}}(d_c^n)^2\right)^2}\cdot \cdot\left(\sum_{n=1}^{N_{nets}}\left(log(y_c^{net}(x))\cdot(d_c^n)^2\right)+\left(\sum_{n=1}^{N_{nets}}(d_c^n)^2\right)\cdot log(y_c^{net})\right) \tag{6.58}$$

$$\frac{\partial Part2_c(x)}{\partial d_c^{net}} = \left(\sum_{n=1}^{N_{nets}}\left(log(1-y_c^{net}(x))\cdot(d_c^n)^2\right)+\left(\sum_{n=1}^{N_{nets}}(d_c^n)^2\right)\cdot log(1-y_c^{net})\right)\cdot \cdot(-\gamma)\cdot(1-P2_c)^{\gamma-1}\cdot P2_c\cdot\frac{2\cdot N_{nets}\cdot d_c^{net}}{\left(\sum_{n=1}^{N_{nets}}(d_c^n)^2\right)^2} \tag{6.59}$$

6.2.5 Feature Based Combination

In *Feature Based Combination*, the input data space is partitioned into N_{reg} regions represented by a reference point v_i. The regions and their reference points are calculated with the *Frequency-Sensitive Competitive Learning* algorithm, *FSCL*, described in algorithm 6.4.

Algorithm 6.4 Frequency-Sensitive Competitive Learning

Initialize the reference points with small random values $v_i(0), i = 1, 2, ..., k$
Initialize f: $f_{reg,0} = 0$, $reg = 1, 2, ..., N_{reg}$, $\beta = 0.1$, and $\gamma = 0.05$
for $ite = 1$ **to** N_{ite} **do**
  for $x = 1$ **to** $N_{patterns}$ **do**
    Find the reference point v_j **Euclidially closest to** x
    for $reg = 1$ **to** N_{reg} **do**
      $z_{reg} = \begin{cases} 1 & \textbf{\textit{if}}\ \textbf{reg=j} \\ 0 & \textbf{\textit{otherwise}} \end{cases}$
      $f_{reg,ite} = f_{reg,ite-1} + \beta\cdot(z_{reg} - f_{reg,ite-1})$
      $b_{reg} = d(x, v_{reg})\cdot\gamma\cdot(\frac{1}{N_{reg}} - f_{reg,ite})$
    end for
    Find the winner:
      $k = \underset{i=1,2,...,N_{reg}}{\arg\min}\ (d(x, v_i) - b_i)$
    for $reg = 1$ **to** N_{reg} **do**
      $\varphi_{reg} = \begin{cases} 1 & \textbf{\textit{if}}\ \textbf{reg = k} \\ 0 & \textbf{\textit{otherwise}} \end{cases}$
      $v_{reg}(ite+1) = v_{reg}(ite) + \varphi_{reg}\cdot c_{ite}\cdot[x - v_{reg}(ite)]$
    end for
  end for
end for

Firstly, the initial reference points are randomly generated with small values close to 0. Then, an iterative procedure, in which all the patterns are considered, is used to adapt them. Each pattern, x, from the learning set is used to update the values of the reference points. The values of β and γ are parameters which were set with a trial and error procedure as shown in appendix B.6.

The update is done in three stages. In the first stage, the region whose reference point, v_{reg}, is the closest to the pattern, x, is determined as region j. Then, a new value, b, is calculated for all the regions. The b values depend on f and z values. According to the algorithm, z_j is set to 1, whereas z is 0 for the other regions.

In the second stage, the winner region, region k, is determined. This is the region in which the difference between the distance to reference point ($d(x, v_k)$) and b_k has the lowest value among all the regions.

In the last stage, the reference point of the winner region, v_k, is updated. This adaptation makes the winner reference point, v_k, to approach to the pattern, x. The reference points of the other regions are kept unchanged because their φ_{reg} values are always 0. for them.

In this algorithm the parameter c_{ite} also represents a slowly decreasing sequence of learning coefficients as in the original reference [76] and it is calculated with equation 6.60.

$$c_{ite} = 1 - 0.9 \cdot \frac{ite}{N_{ite}} \tag{6.60}$$

Once all the final reference points are calculated, the combination procedure uses them to calculate the specific optimal parameters for each region. It means that the parameters of the combiners will depend on the pattern which is being classified and the region it belongs to.

According to [76], *Feature Based Combination* is only applied to two models previously described:

- *Choquet Integral with Data-Dependent Densities.*
- *Weighted Average with Data-Dependent Weights.*

The description of both methods will be introduced below. They are similar to the original versions, but some parameters will now depend on the region of the pattern which is being classified.

6.2.5.1 Choquet Integral with Data-Dependent Densities

Choquet Integral with Data-Dependent Densities, henceforth *Choquet ddd*, is the feature based version of *Choquet Integral*. In this model, the fuzzy densities g_{reg}^{net} depend on the region and they are given by:

$$g_{reg}^{net} = d_s^{reg} \cdot \frac{p_{reg}^{net}}{\sum_{n=1}^{N_{nets}} p_{reg}^{n}} \tag{6.61}$$

Where, p_{reg}^{net}, is the percentage of correctly classified patterns by network, net. Only the patterns which belong to the region reg from the validation set, V, are used to calculate this percentage. The value of the desired sum, d_s^{reg} is calculated with:

$$d_s^{reg} = 0.95 \cdot \frac{\sum_{net=1}^{k} p_{reg}^{net}}{\max_{net=1}^{k} p_{reg}^{net}} \tag{6.62}$$

The class assigned to the pattern is determined with equation 6.63 where $Cg_c^{reg}(x)$ represents the *Choquet Integral* for the class c when the pattern, x, matches into region reg.

$$Class(x) = \underset{c=1,2,...,N_{classes}}{\arg\max} \; Cg_c^{reg}(x) \tag{6.63}$$

Here, the values for the *Choquet Integral* are calculated as in the original combiner. But, each value of the integral $Cg_c^{reg}(x)$ will depend on the region associated to the pattern, reg, because its final value will be based on g_{reg}^{net}. Then, equation 6.64 is used to calculate the region related to a pattern, reg.

$$reg = \underset{r=1,2,...,N_{regions}}{\arg\min} \; d(x, v_r) \tag{6.64}$$

According to the previous equation, *arg min* assigns to reg the region, r, whose reference point, v_r, has the lowest euclidean distance, $d(x, v_r)$, with respect to the classified pattern, x. This region is selected because its reference point is, among all the regions, the closest one to the pattern.

6.2.5.2 Weighted Average with Data-Dependent Weights

The *Weighted Average with Data-Dependent Weights*, henceforth *W.Ave ddw*, mixes *Weighted Average* and *Choquet Integral with Data-Depend Densitied* in a single combiner. On the one hand, a weighted average scheme is used to generate the final output. On the other hand, the weights are calculated with the densities, g_{reg}^{net}, of the *Choquet Integral*.

The weighted average of equation 6.65 is applied to fuse the outputs. The weight values, g_{reg}^{net}, are calculated with equation 6.61.

$$\widehat{y}(x) = \sum_{net=1}^{N_{nets}} \left(g_{reg}^{net} \cdot y^{net}(x) \right) \tag{6.65}$$

Moreover, the region, reg, associated to a pattern, x, is also determined by equation 6.64 as in *Choquet Integral with Data-Depend Densitied*.

6.2.6 Two-Layered MCS

The main characteristic of this kind of classifiers is that the system is composed by two layers of simple classifiers. In this case, the networks of the first layer are used to solve the problem whereas the networks of the second one are trained for combination or aggregation tasks. Usually, they are complex systems in which the training procedure involves the training of the networks of the first and second layer.

An ensemble of neural networks can be used as the simple classifiers of the first layer, whereas one or several networks can be trained to fuse the outputs of the ensemble. A part of our research has focused on developing new combiners based on this kind of systems.

The two most known models are *Stacked Generalization* [95] and *Mixture of Experts* [94]. Although these two systems are quite different, they can be adapted to be applied as ensemble combiners. In fact, we have proposed in this thesis two new combiners based on *Stacked Generalization* (*Stacked* and *Stacked+*), figure 6.1, and a new combiner based on the *Mixture of Experts*, figure 6.2.

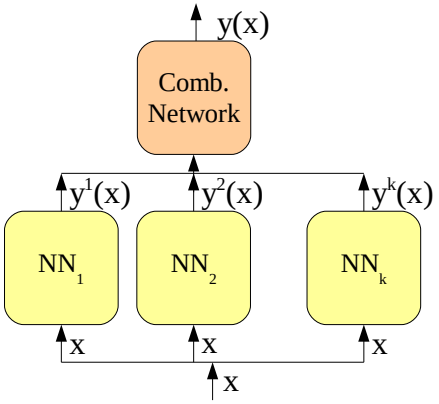


Figure 6.1: Basic Stacked Generalization model

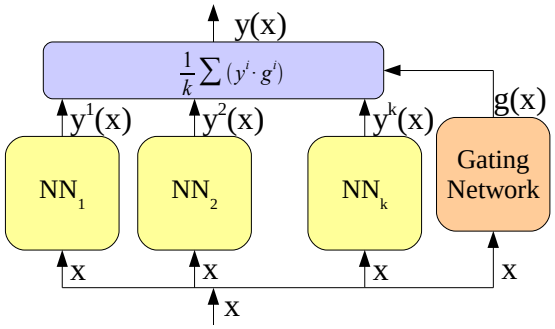


Figure 6.2: Basic Mixture of Experts model

In the previous figures, we can see that the philosophy of both models is different. The combination network of *Stacked Generalization* processes the output of the networks in order to give a final output. The gating network of *Mixture of Experts* uses the original pattern in order to set the weights of a weighted average.

We have considered that the new combiners should not appear in this chapter because they are based on advanced models which will be detailed in other chapters. In this way, we will perform a special analysis of the new combiners and the advanced models in chapters 9 and 10. Moreover, we can do a deeper comparison among the original proposals and the new approaches (combiners) we will introduce.

6.3 Experimental Setup

Prior to our research, Verikas et al. introduced an important combiner analysis [76]. However, we consider that this comparison was not complete enough because only one ensemble model, *CVCv2*, was trained on four databases from the *ELENA* project [110]. Moreover, the experiments were repeated eight times using ensembles of only 5 networks and standard *MF* networks with 10 hidden units. Other ensemble sizes were not used and the number of hidden units was not optimized for the problems.

In the research shown here, we want to expand that research by including other ensemble alternatives, the *RBF* network, more combiners, a high number of databases and we will apply optimal training parameters. Moreover, two general measurements will be employed in order to compare all the ways to fuse the networks. The main characteristics of our research are:

- Nineteen classification problems from the *UCI Repository*.
- Ensembles of *MF* networks generated by:
  - *Simple Ensemble*.
  - *Decorrelated v1*.
  - *Cross Validation Comittee v2*.
  - *Conservative Boosting*.
- Ensembles of *RBF* networks generated by *Simple Ensemble*.
- Four different ensemble sizes: 3, 9, 20 and 40 networks.
- Fourteen combiners applied.
- Optimized parameters for networks, ensembles and combiners.
- The experiments have been repeated ten times with different partitions of training, validation and test sets in order to obtain:
  - Mean value of performance.
  - Error rate by standard error theory.
- Two general measurements applied to the comparison:
  - Mean *Increase of Performance*.
  - Mean *Percentage of Error Reduction*.

The description of the nineteen datasets used in the experiments can be found in appendix A. The ensemble methods applied were previously described in chapters 4 and 5. All the values of the parameters are in appendix B. Concretely, the optimized training parameters of the networks are in sections B.3 and B.4, the parameters of the ensembles are in section B.5 whereas the specific parameters of the combiners are in section B.6. Finally, the general measurements applied to compare the combiners analyzed are the mean *IoP* (equation 3.5) and the mean *PER* (equation 3.6).

6.4 Results and discussion

The current comparison has been split into two subsections in order to differentiate the ensembles of *Multilayer Feedforward* and *Radial Basis Functions* networks. Firstly, the combiners reviewed are applied to the four different ensemble alternatives which were employed to generate ensembles of *MF* networks. Secondly, a similar but different research is accomplished with *Simple Ensembles* of *RBF* networks.

6.4.1 Combining ensembles of *MF* networks

For the case of ensembles of *MF* networks, the combiners analyzed in this chapter are employed with the ensembles previously trained with *Simple Ensemble*, *Conserboost*, *CVCv2* and *Decov1*. Table 6.1 shows the results related to *Simple Ensemble*.

Table 6.1: Combining *SE* of *MF* networks

	Mean *IoP*				**Mean** *PER*			
Combiner	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Average	4.97	5.27	5.34	5.43	21.84	23.65	23.73	24.64
DAN	4.28	4.46	4.36	4.18	16.57	18.56	17.05	16.7
DAN 2	4.25	4.43	4.36	4.2	16.4	18.25	17.04	16.76
Voting	4.77	5.03	5.06	5.17	19.83	22.09	22.39	23.13
Nash	4.85	5.11	5.23	5.29	20.5	21.93	22.81	23.17
Borda	4.77	4.96	4.95	5.09	19.41	21.06	21.21	22.23
WTA	4.83	5.23	5.26	5.18	20.89	22.61	23.14	22.65
W.Ave	4.71	5.55	5.31	4.91	20.03	25.46	24.89	19.87
Bayes	4.89	4.65	4.04	3.65	20.57	18.22	13.11	9.38
Choquet	4.81	5.09	–	–	20.89	22.08	–	–
BADD	4.97	5.27	5.34	5.43	21.84	23.65	23.73	24.64
Zimm	4.96	5.38	4.7	2.02	22.32	24.54	20.27	2.47
Choquet ddd	4.73	5	–	–	20.53	21.46	–	–
W.Ave ddw	4.97	5.33	–	–	21.81	23.83	–	–

In the previous table, we can firstly see that the results related to all the combiners based on *Choquet Integral* (*Choquet*, *Choqued ddd* and *W.Ave ddw*) has been limited to ensembles of 3 and 9 networks because the time required to calculate some parameters was so high for the cases of ensembles of 20 and 40 networks.

Secondly, *Output average* (*Average*), *Weighted Average with Data-Dependent Weights* (*W.Ave ddw*) and *BADD Defuzzyfication Strategy* (*BADD*) provide the best results for all the ensembles. *BADD* is, in this case, equivalent to *Output average* because all the networks are trained on the original training set and the membership degree, μ, is the same in all the networks.

Thirdly, *Weighted Average* (*W.Ave*) and the combiner based on *Zimmermann's Compensatory Operator* (*Zimm*) also provide good results in some cases. But their performance is quite low for ensembles of 40 networks, specially in the case of *Zimmermann's Compensatory Operator*.

Finally, we consider that *Output average* is a good combiner in the case of *Simple Ensemble* because it is simple and it provides good results for any ensemble size. *Weighted Average with Data-Dependent Weights* also provides good results, but it could not be applied to high sized ensembles because it is based on *Choquet Integral*. Other alternatives, such as *Weighted Average* and *Zimmermann's operator* can only be applied to low sized ensembles, 3 networks. As we can see, only *Output average* fits well in all the cases.

In order to compare easily all the alternatives to fuse the outputs with respect to *Output average*, we have calculated the difference in the general measurements provided by *Output average* and the other combiners. A positive value means that the alternative is better than *Output average* and a negative one means the opposite. These results are shown in table 6.2.

Table 6.2: Differences of *Output Average* and the other combiners - *SE* of *MF* networks

	Mean *IoP*				**Mean** *PER*			
Combiner	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
DAN	−0.69	−0.81	−0.98	−1.25	−5.27	−5.09	−6.68	−7.94
DAN 2	−0.72	−0.84	−0.98	−1.23	−5.44	−5.4	−6.69	−7.88
Voting	−0.2	−0.24	−0.28	−0.26	−2.01	−1.56	−1.34	−1.51
Nash	−0.12	−0.16	−0.11	−0.14	−1.34	−1.72	−0.92	−1.47
Borda	−0.2	−0.31	−0.39	−0.34	−2.43	−2.59	−2.52	−2.41
WTA	−0.14	−0.04	−0.08	−0.25	−0.95	−1.04	−0.59	−1.99
W.Ave	−0.26	0.28	−0.03	−0.52	−1.81	1.81	1.16	−4.77
Bayes	−0.08	−0.62	−1.3	−1.78	−1.27	−5.43	−10.62	−15.26
Choquet	−0.16	−0.18	–	–	−0.95	−1.57	–	–
BADD	0	0	0	0	0	0	0	0
Zimm	−0.01	0.11	−0.64	−3.41	0.48	0.89	−3.46	−22.17
Choquet ddd	−0.24	−0.27	–	–	−1.31	−2.19	–	–
W.Ave. ddw	0	0.06	–	–	−0.03	0.18	–	–

With this new table, we can see that the differences with respect to *Output average* are, in general, small and negative. This means that the results provided by *Output average*, in most of the cases, are slightly better than the results provided by the other alternatives.

Table 6.3 shows the results of the current research related to ensembles previously trained with *Decorrelated v1*, the best ensemble model according to the comparison shown in chapter 4.

As can be seen in table 6.3, *Output average* and *Weighted Average with Data-Dependent Weights* (*W.Ave ddw*) provide the best results in ensembles trained with *DECOv1*. The results provided by the other models are, in general, worse than them. Moreover, *BADD* is also equivalent to *Output average*. In this ensemble, *Output average* can also be considered the best combiner.

Table 6.3: Combining *DECOv1* of *MF* networks

	Mean *IoP*				Mean *PER*			
Combiner	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Average	5.48	5.78	5.81	5.83	24.73	26.63	26.84	27.08
DAN	3.91	4.11	3.9	3.68	15.52	17.34	17.17	16.74
DAN 2	3.93	4.08	3.87	3.65	15.53	17.14	16.99	16.38
Voting	5.06	5.49	5.43	5.44	22.66	25.27	25.17	25.52
Nash	5.12	5.38	5.59	5.58	21.6	22.73	24.57	24.52
Borda	5	5.25	5.24	5.21	20.99	22.78	23.13	23.35
WTA	5.46	5.24	5.37	5.33	23.92	22.06	23.84	23.24
W.Ave	5.43	5.36	5.52	4.88	24.68	23.66	24.94	20.77
Bayesian	5.33	4.68	3.99	3.41	23.63	17.99	13.08	8.79
Choquet	5.38	5.18	–	–	23.11	21.54	–	–
BADD	5.48	5.78	5.81	5.83	24.73	26.63	26.84	27.08
Zimm	5.34	5.53	4.76	1.91	23.24	24.83	20.23	3.87
Choquet ddd	5.26	5.14	–	–	22.77	21.45	–	–
W.Ave. ddw	5.53	5.79	–	–	25.03	26.61	–	–

Table 6.4 shows the results related to ensembles generated by *CVCv2*, this is the ensemble method applied in the study done by Verikas et al. in [76].

Table 6.4: Combining *CVCv2* of *MF* networks

	Mean *IoP*				Mean *PER*			
Combiner	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Average	5.47	5.95	5.98	5.85	25.73	27.08	25.34	23.69
DAN	4.22	4.27	4.61	3.57	16.2	16.39	16.86	12.79
DAN 2	4.17	4.27	4.53	3.61	15.9	16.44	16.39	13.05
Voting	5.25	5.82	5.93	5.63	23.98	26.05	25.78	22.52
Nash	5.33	5.63	5.62	5.49	23.38	24.1	22.49	21.37
Borda	5.19	5.62	5.82	5.46	22.98	24.47	24.75	21.03
WTA	5.48	5.39	5.22	4.9	26.03	23.71	20.95	19.82
W.Ave	5.7	5.27	4.75	4.08	24.25	22.81	18.9	14.4
Bayesian	5.43	4.82	3.93	2.61	23.7	16.27	11.06	1.44
Choquet	5.37	5.31	–	–	24.81	23.69	–	–
BADD	5.25	5.81	5.99	5.84	23	25.47	25.61	23.45
Zimm	5.45	5.12	4.14	2.29	23.93	21.58	14.51	−8.37
Choquet ddd	5.31	5.13	–	–	24.58	22.87	–	–
W.Ave. ddw	5.48	5.98	–	–	25.77	27.19	–	–

In *CVCv2*, the best combiner, according to the mean *PER*, is *Winner Takes All* (*WTA*) for ensembles of 3 networks. For ensembles of 9 networks, the best choice is *Weighted Average with Data-Dependent Weights*. *Voting* is the combiner which fits better for ensembles of 20 networks. *Output average* provides the best overall results for ensembles of 40 networks. Moreover, *BADD defuzzyfication* also provides good results in general for any ensemble size.

According to the analysis done by Verikas et al., *Weighted Average with Data-Dependent Weights* provided the best overall results. In this thesis, this combiner also provides the best overall results for *CVCv2*. Moreover, *Choquet Fuzzy Integral with Data-Dependent Densities* (*Choqet ddd*) also provided good results. Furthermore, *BADD* combination scheme was also among the best approaches. Altough the last two combiners provided good results here, they were outperformed by other alternatives such as *Output average*.

Finally, the results provided by *Output average* are close to the best results for all the ensemble sizes. The other *good* alternatives to combine ensembles do not provide good results for any ensemble size. For instance, *BADD defuzzyfication* does not provide good results in ensembles of 3 networks and *Winner Takes All* had low performance in ensembles of 40 networks. Maybe, *Output average* can also be considered the best alternative to fuse the networks in *CVCv2* because it provides high performance for any case.

Table 6.5 shows the results of the combiners on ensembles trained with *Conserboost*, one of the best boosting variants according to the results shown in chapter 5.

Table 6.5: Combining *Conserboost* of *MF* networks

	Mean *IoP*				**Mean** *PER*			
Combiner	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Boosting	4.42	5.31	5.65	6.06	19.72	25.63	26.62	27.84
Average	4.98	5.57	5.75	5.98	23.04	27.08	27.6	26.87
DAN	0.84	−4.89	−5.58	−4.6	−6.95	−39.61	−57.67	−59.21
DAN 2	0.8	−4.9	−5.66	−4.65	−7.4	−39.58	−58.75	−60.15
Voting	4.31	4.93	4.91	5.59	19.11	23.64	23.73	26.36
Nash	4.29	4.94	5.33	5.58	17.3	22.46	25.08	25.16
Borda	3.94	4.68	4.83	5.53	15.97	21.65	23.05	25.93
WTA	4.5	2.83	1.27	−0.91	18.73	8.36	−4.26	−31.45
W.Ave	4.39	5.02	5	4.74	20.43	23.15	22.78	21.26
Bayesian	3.93	3.37	2.23	0.98	16	10.77	2.08	−10.73
Choquet	4.26	2.32	–	–	16.29	2.62	–	–
BADD	4.98	5.57	5.75	5.98	23.04	27.08	27.6	26.87
Zimm	4.48	4.3	−0.2	−3.1	20.39	18.43	−8.24	−64.22
Choquet ddd	4.03	2.29	–	–	14.69	2.32	–	–
W.Ave. ddw	4.89	5.6	–	–	22.41	27.05	–	–

For the case of ensembles of 3, 9 and 20 networks, *Output average* performs better than the specific *Boosting combiner* shown in the first row. Moreover, *Weighted Average with Data-Dependent Weights* (*W.Ave ddw*) also provides better results for ensembles of 3 and 9 networks. In general, *Output average* should be applied to fuse the networks generated by *Conserboost* for any ensemble size. Finally, the specific *Boosting combiner* is the best alternative for ensembles of 40 networks and this concrete configuration provides the best overall results.

The results of the four previous tables show that the best combiner is, in general, *Output average*. It is the only model which provides relative good results according to the general measurements, mean IoP and mean PER, for all the ensembles alternatives and for all the sizes (low, medium and high). However, its results are, sometimes, slightly outperformed by other combination rules such as *Weighted Average with Data-Dependent Weights*. And for the case of *Boosting* variants, *Boosting combiner* should also be seriously considered for high sized ensembles, 40 networks, because it provides the best results.

In general, the results provided by the combiners are quite similar among them but there are some specific cases in which a specific alternative fits specially better on a few datasets. It means that two (or more) combiners can reach a similar general values but one is more suitable for a set of databases and the other one fits better on another subset of classification problems.

This last behavior can not be seen in the general results. However, there is a special case which can be given as example, the results (percentage of correctly classified patterns from the test set) on dataset *ionos* are between 89% and 91% when *Output average* is applied to combine the ensembles previously trained with *Simple Ensemble*, *DECOv1* and *CVCv2*. The complete results related to these ensembles are in the appendix C, tables C.93 to C.104, but we show a resume in table 6.6.

Table 6.6: Performance of dataset *ionos* with *Output Average*

Ensemble	3-**net**	9-**net**	20-**net**	40-**net**
SE	91.1 ± 1.1	90.3 ± 1.1	90.4 ± 1	90.3 ± 1
DECOv1	90.9 ± 0.9	90.7 ± 1.0	91.1 ± 0.9	91.0 ± 1.0
CVCv2	89.7 ± 1.4	90.4 ± 1.3	91.0 ± 0.9	92.0 ± 1.0

In table 6.7, we show the resume of the main results related to the same ensemble but, in this case, *Bayesian Combination* is used to combine the ensembles.

Table 6.7: Performance of dataset *ionos* with *Bayesian Combination*

Ensemble	3-**net**	9-**net**	20-**net**	40-**net**
SE	91.4 ± 1.1	93.1 ± 1.4	93.1 ± 1.4	93.4 ± 1.4
DECOv1	92.3 ± 1	93 ± 0.9	92 ± 1.1	94.1 ± 1
CVCv2	92 ± 1.2	93 ± 0.9	94.1 ± 0.9	93.4 ± 1.1

We can see in the previous table that *Bayesian Combination* can provide better results than *Output average* when they are applied to combine the same ensembles. In fact, the results of *ionos* are between 93% and 94% when *Bayesian Combination* is used to fuse the networks in ensembles of 20 and 40 networks. This values are higher that the ones shown for *Output average*.

To perform easier the comparison between these two combiners in the case of database *ionos*, we have included a new table with the difference in performance between *Bayesian Combination* and *Output average*. A positive value in this table means that *Bayesian Combination* provides a better percentage of correctly classified patterns from the test set than *Output average*.

Table 6.8: Difference in performance of *Bayesian* and *Output Average* in *ionos*

Ensemble	3-net	9-net	20-net	40-net
SE	0.3	2.8	2.7	3.1
DECOv1	1.4	2.3	0.9	3.1
CVCv2	2.3	2.6	3.1	1.4

In the last table, the difference in performance shows that *Bayesian Combination* can reach an increase in performance around 3% with respect to *Output average* for *ionos* when ensembles of 20 or 40 networks are combined. Although *Bayesian Combination* always performs better than *Output average* because the results are always positive, there are two cases in which the difference can be considered low because it is lower than 1%.

As shown in tables 6.1 to 6.5, *Bayesian Combination* performs quite worse than *Output average* for ensembles of 20 and 40 networks. To see it more clearly, we show in table 6.9 the difference in the mean IoP across all databases of *Bayesian Combination* and *Output average*.

Table 6.9: Difference in mean PER of *Bayesian* and *Output Average*

Ensemble	3-net	9-net	20-net	40-net
SE	-0.08	-0.62	-1.3	-1.78
DECOv1	-0.15	-1.1	-1.82	-2.42
CVCv2	-0.04	-1.13	-2.05	-3.24

In table 6.9, the values are always negative because *Output average* always provide a better mean IoP than *Bayesian Combination*. *Bayesian Combination* is quite interesting because it is not recommended to combine ensembles of 40 networks but it provides the best results for *ionos* in ensembles of 40 networks.

As a final conclusion of this part of the experiments, we can say that *Output average* provides good results for any ensemble size and ensemble method but the other combiners can also provide excellent results for specific datasets. In table 6.10, we show for each dataset the performance of the ensembles of 40 networks designed with *Simple Ensemble* and *Output average*. Moreover, the best result considering all the alternatives to fuse the networks, except *Output average*, is shown. Finally, we show the difference between *Output average* and the highest performance.

Table 6.10 shows that the difference between the best model and *Output average* is never negative but, in most of the datasets, it is low. In databases *band*, *bupa*, *cred* and *derma*, *Output average* also provides the highest results because the difference with the best model is 0%. *Output average* should be the first alternative in combining because it provides good results in general, but the 'optimal' combiner depends on the dataset as we can clearly see for *ionos*, *vowel*, *glas*, *mok1* and other datasets.

Table 6.10: Performance of *Output Average* and the best combiner for each dataset

Database	Output Average	Best Combiner		Difference
aritm	73.8 ± 1.1	74.9 ± 1.3	*Zimm*	1.1%
bala	95.9 ± 0.5	97 ± 0.3	*W.Ave*	1.1%
band	73.8 ± 1.3	73.8 ± 1.3	*BADD*	0%
bupa	72.7 ± 1.1	72.7 ± 1.1	*BADD*	0%
cred	86.5 ± 0.7	86.5 ± 0.7	*Nash*	0%
derma	97.6 ± 0.7	97.6 ± 0.7	*Voting*	0%
ecoli	86.9 ± 0.7	87.5 ± 0.6	*WTA*	0.6%
flare	81.6 ± 0.5	81.7 ± 0.5	*Bayesian*	0.1%
glas	94.2 ± 0.6	95.8 ± 0.9	*W.Ave*	1.6%
hear	82.9 ± 1.5	83.1 ± 1.5	*Nash*	0.2%
img	96.8 ± 0.2	97.2 ± 0.3	*W.Ave*	0.4%
ionos	90.3 ± 1	93.4 ± 1.4	*Bayesian*	3.1%
mok1	98.3 ± 0.9	100 ± 0	*Bayesian*	1.7%
mok2	91.1 ± 1.2	91.6 ± 1.2	*Zimm*	0.5%
pima	75.9 ± 1.2	76 ± 1.2	*Borda*	0.1%
survi	74.3 ± 1.3	74.6 ± 0.9	*Zimm*	0.3%
vote	95.6 ± 0.5	95.8 ± 0.6	*Zimm*	0.2%
vowel	92.2 ± 0.7	94.7 ± 0.4	*W.Ave*	2.5%
wdbc	96.9 ± 0.5	97 ± 0.4	*Borda*	0.1%

6.4.2 Combining ensembles of *RBF* networks

Table 6.11 shows the results of the different combiners on ensembles of *RBF* networks previously trained with *Simple Ensemble*.

Table 6.11: Combining *Simple Ensemble* of *RBF* networks (Original ensemble)

	Mean *IoP*				Mean *PER*			
Combiner	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Average	0.02	0.14	0.14	0.23	0.15	3.71	3.06	3.29
DAN	−0.09	−0.12	−0.02	0.00	0.63	−0.51	2.08	−1.89
DAN 2	−0.14	−0.11	−0.07	0.03	0.17	−0.59	1.30	−1.76
Voting	−0.02	0.17	0.14	0.18	1.94	4.40	3.19	3.33
Nash	0.00	0.10	0.09	0.16	0.09	3.50	2.62	0.94
Borda	−0.03	0.14	0.14	0.17	1.87	4.16	3.10	3.23
WTA	0.06	0.10	0.12	0.10	1.27	0.92	0.87	−0.76
W.Ave	−0.11	−0.16	−1.84	−9.50	5.56	5.74	−50.49	−313.54
Bayesian	−0.07	0.06	0.06	0.06	4.02	−0.37	−6.91	−10.56
Choquet	0.10	0.04	–	–	1.31	0.53	–	–
BADD	0.02	0.14	0.14	0.23	0.15	3.71	3.06	3.29
Zimm	−10.57	−9.47	−9.82	−8.19	−274.63	−300.11	−218.13	−205.21
Choquet ddd	0.05	0.04	–	–	1.00	0.63	–	–
W.Ave. ddw	0.02	0.14	–	–	0.21	3.83	–	–

As can be seen in the table, *Output average* is one of the best alternatives to fuse the networks. However, its performance is similar to a single network because the mean IoP is close to 0 and the mean PER is lower than 4%. Moreover, *Voting* and *Borda Count* (*Borda*) also provides good results according to the mean PER.

Both versions of *Dinamically Averaged Networks* (*DANv1* and *DANv2*), *Nash Vote* and *Zimmermann Operator* provide worse results than *Output average*. According to the mean PER, *Weighted Average*, *Bayesian Combination* and *Choquet Integral* only provide good results when the ensemble size is low, 3 networks, but their performance decrease when the ensemble size is higher.

Finally, the IoP and PER values are not correlated in some cases. This means that *Output average* fits specially better for a subset of datasets whereas another subset of datasets is better combined by the other alternatives. This behavior can be clearly seen in *Weighted Average*, *Majority Voting*, *Borda Count* and *Bayesian Combination* for the case of ensembles of 3 networks where the mean IoP is negative but the mean PER is positive and quite high.

The analysis of the results is a little bit more difficult in the case of *RBF* networks because some combiners may not fit well on this network architecture. It is important to mention that the outputs provided by *MF* networks range from 0 to 1 so they can be interpreted as the probability of corresponding to the associated class. A high value on the i-th output neuron means that the probability of the pattern to belong to class i is also high. This behavior can not be expected in *RBF* networks because the outputs are not $[0, \ldots, 1]$ ranged.

Maybe, there are models which can not work well, such as *Zimmermann's Operator*, if the outputs are not in a this range. For this reason three different normalization procedures have been applied in order to solve this problem. The first normalization, *threshold*, is described in equation 6.66. It consists of cutting values lower than 0 and higher than 1 by applying a threshold. *Min-max* is the second procedure and it is described in equation 6.67. This is the typical normalization in which the lowest value of the output vector is set to zero and the highest value is set to one. Finally, the last alternative we have used, *sum*, is calculated with equation 6.68. In this last case, the summatory of all the elements of the output vector must be one and there can not be negative values.

$$\widehat{y}_{class}(x) = \begin{cases} 0 & \text{if } y(x) < 0 \\ 1 & \text{if } y(x) > 1 \\ y_{class}(x) & \text{otherwise} \end{cases} \tag{6.66}$$

$$\widehat{y}_{class}(x) = \frac{y_{class}(x) - min(y)}{max(y) - min(y)} \tag{6.67}$$

$$\widehat{y}_{class}(x) = \frac{y_{class}(x) - min(y)}{sum(y - min(y))} \tag{6.68}$$

Table 6.12 shows the results of the combiners on ensembles of *RBF* networks whose outputs have been normalized with the *threshold* equation.

Table 6.12: Combining *Simple Ensemble* of *RBF* networks (*Threshold* norm.)

	Mean *IoP*				**Mean** *PER*			
Combiner	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Average	0.01	0.14	0.16	0.24	0.09	3.78	3.27	3.36
DAN	−0.09	−0.12	−0.03	−0.01	0.63	−0.47	2.16	−1.87
DAN 2	−0.14	−0.10	−0.07	0.03	0.17	−0.25	1.30	−1.76
Voting	−0.02	0.17	0.14	0.18	1.94	4.40	3.19	3.33
Nash	0.00	0.08	0.05	0.08	0.09	3.34	2.21	0.18
Borda	−0.03	0.14	0.14	0.17	1.87	4.16	3.10	3.23
WTA	0.06	0.08	0.11	0.09	1.27	0.78	0.72	−0.85
W.Ave	−0.20	−0.26	−1.90	−9.74	4.51	3.73	−48.03	−316.57
Bayesian	−0.07	0.06	0.06	0.06	4.02	−0.37	−6.91	−10.56
Choquet	0.12	0.06	–	–	1.89	1.14	–	–
BADD	0.01	0.14	0.16	0.24	0.09	3.78	3.27	3.36
Zimm	0.13	−0.20	−0.87	−3.20	7.15	0.85	−11.10	−69.37
Choquet ddd	0.07	0.06	–	–	1.51	1.24	–	–
W.Ave. ddw	0.02	0.14	–	–	0.15	3.83	–	–

The results provided are quite similar to the results shown in table 6.11. However, this normalization highly improved the results of *Zimmerman Operator*, specially for the case of 3 networks in the ensemble. It also slightly improved, *Choquet Integral with data depend densities*. However, *Weighted Average* is slightly worse than the original *RBF* output.

Table 6.13 shows the results related to the ensembles of *RBF* networks where the *min-max* equation has been used to normalize the outputs.

Table 6.13: Combining *Simple Ensemble* of *RBF* networks (*min-max* normalization)

	Mean *IoP*				Mean *PER*			
Combiner	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Average	−0.02	0.16	0.15	0.18	1.69	4.23	3.41	3.29
DAN	−0.17	−0.14	−0.12	−0.10	−0.09	−1.03	−1.57	−3.88
DAN 2	−0.17	−0.14	−0.11	0.00	−0.08	−0.81	−1.08	−2.33
Voting	−0.02	0.17	0.14	0.18	1.94	4.40	3.19	3.33
Nash	0.01	0.00	−0.08	−0.29	1.68	−1.91	−7.98	−13.08
Borda	−0.03	0.14	0.14	0.17	1.87	4.16	3.10	3.23
WTA	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
W.Ave	−6.67	−12.35	−17.54	−28.53	−816.49	−901.34	−986.77	−1273.33
Bayesian	−0.07	0.06	0.06	0.06	4.02	−0.37	−6.91	−10.56
Choquet	−0.42	−0.81	–	–	−7.46	−17.15	–	–
BADD	−0.02	0.16	0.15	0.18	1.69	4.23	3.41	3.29
Zimm	0.03	0.02	−0.08	−0.28	2.21	−1.48	−8.01	−13.32
Choquet ddd	−0.46	−0.91	–	–	−7.76	−18.23	–	–
W.Ave. ddw	−0.02	0.15	–	–	1.79	4.12	–	–

The results provided by *min-max* are also similar to the results shown in table 6.11. This normalization only improved *Zimmerman* and *Weighted Average with data depend weights*. However, combiners such as *Choquet Integral*, *Choquet Integral with data depend densities* and *Weighted Average* now perform quite worse than a single *RBF* network and the *RBF* ensemble without normalization. Concretely, in *Weighted Average*, the mean *PER* value is close to −1000%.

Finally, table 6.14 shows the results of all the alternatives to fuse *RBF* networks which have been normalized with the *sum* equation.

Table 6.14: Combining *Simple Ensemble* of *RBF* networks (*sum* normalization)

	Mean *IoP*				Mean *PER*			
Combiner	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Average	0.01	0.13	0.14	0.20	0.28	3.88	3.05	3.02
DAN	−0.15	−0.11	−0.10	0.03	0.25	−0.17	−0.61	−1.27
DAN 2	−0.16	−0.15	−0.10	0.02	0.08	−0.88	−0.74	−1.64
Voting	−0.02	0.17	0.14	0.18	1.94	4.40	3.19	3.33
Nash	0.00	0.10	0.08	0.12	0.09	3.48	2.48	0.60
Borda	−0.03	0.14	0.14	0.17	1.87	4.16	3.10	3.23
WTA	0.02	0.13	0.12	0.12	2.17	1.50	1.64	−0.22
W.Ave	−0.25	−0.43	−2.43	−9.93	−0.90	−0.99	−60.35	−319.62
Bayesian	−0.07	0.06	0.06	0.06	4.02	−0.37	−6.91	−10.56
Choquet	0.04	0.11	–	–	4.19	1.31	–	–
BADD	0.01	0.13	0.14	0.20	0.28	3.88	3.05	3.02
Zimm	0.15	−0.41	−3.49	−5.97	6.07	−3.34	−84.39	−149.41
Choquet ddd	−0.01	0.11	–	–	3.94	1.54	–	–
W.Ave. ddw	0.01	0.12	–	–	0.26	3.83	–	–

The performance and analysis of *min-max* and *sum* are similar. This similarity was expected because both procedures are nearly the same. However, the *sum* normalization performs better than *mix-max* for the following combiners: *Weighted Average*, the two *Choquet Integral* versions (*Choquet* and *Choquet ddd*) and, in some cases, *Zimmerman Operator*.

In general, the results provided by most of the combiners tend to be similar. However, their accuracy highly depend on the normalization applied to the outputs of the *RBF* networks, according to the complete results and the results shown in tables 6.11 to 6.14. There are three combiners, *Majority Voting*, *Borda Count* and *Bayesian Combination*, which are not affected by any of the normalization procedures used in this research. These combiners are based on relative measurements which are not altered by the normalization equations. Furthermore, *Majority Voting* and *Borda Count* have provided good general results and they are, maybe, the most appropriated combiners for *RBF* networks.

Although *Output average* provides good results in general and for any normalization procedure, they can be improved if the most appropriate model and normalization equation are chosen. Concretely, the best general performance is obtained by the ensembles of three *RBF* networks combined by *Zimmermann's operator* when *sum* and *threshold* are used to obtain *RBF* networks with $[0, \ldots, 1]$ ranged outputs. In those cases, the mean PER was higher than 6% and the highest overall value, 7.15%, was also obtained with *Zimmermann's operator*.

Furthermore, the general results also shown that the combiners are dataset dependent. Two different alternatives with similar values of the mean IoP do not have similar values of the mean PER and vice versa. This means that the first model fits better for some datasets and the other one fits better for other classification problems. Moreover, there are some cases in which the mean IoP is negative whereas the mean PER is positive. In this case, we can realize easily that both measurements are important and they do not have to completely agree. Maybe, this behavior means that the alternative used to fuse the networks is valid for some classification problems but it is not a good choice for the others.

To conclude the experiments related to *RBF* networks, we also can say that *Output average* provides good results in all the cases considered. As in the research done in ensembles of *MF* networks, *Output average* should be one of the first options when ensembles of *RBF* networks are being combined. Moreover, the voting schemes *Majority Voting* and *Borda Count* also provide good general results according to the mean PER and they do not require any normalization procedure. The other alternatives may be less important because their results depend on the classification problem, ensemble size and normalization procedure used. However, they can provide high results in specific cases.

6.5 Conclusions

Some interesting papers were studied and deeply analyzed in order to do this comparison. Fourteen combiners (fifteen in boosting ensembles) were selected to be employed in ensembles of *MF* and *RBF* networks. The experimental setup depends on the network architecture chosen. In the case of *MF* networks, we have applied them to four ensemble methods. In the case of *RBF* networks, only *Simple Ensemble* was considered but three equations were introduced to normalize the output of the networks.

Firstly, the different alternatives were applied to fuse the networks in ensembles of *MF* networks. A representative set of ensemble models were used in the experiments: *Simple Ensemble*, *Decorrelated version 1*, *Cross Validated Committee version 2* and *Conservative Boosting*. The results are quite interesting because *Output average* provides good results even for the ensembles generated by *Conservative Boosting*. Moreover, *Weighted Average with Data Depend Weights* should be seriously considered for the case of a low number of networks in the ensemble (3 and 9 networks in the ensemble). Furthermore, *Boosting combiner* should also be applied to combine ensembles of a large number of networks in the ensemble (40 networks) generated by the boosting variants.

Secondly, the combiners were applied to ensembles of *RBF* networks previously generated by *Simple Ensemble*. Unfortunately, the outputs provided by the *RBF* networks used in the experiments did not range from zero to one so a normalization procedure had to be applied. For this reason, three different normalization equations were used to obtain the final output of the *RBF* networks with information inside $[0, 1]$. Finally, the experiments involved the original and the modified ensembles.

The results related to ensembles of *RBF* networks show that the *Output average* also provides good results for any case and normalization procedure applied. Moreover, *Majority Voting* and *Borda Count* also provided good general results. However, the performance of most of the combiners, expect the voting ones, depend on the normalization procedure applied. The extreme cases are provided by ensembles of 3 networks combined by *Zimmermann's Operator*. On the one hand, it performs quite worse than a single network when the original outputs, without any normalization, are combined, in this case, the *mean* PER is lower than -10%. On the other hand, it provides the best overall *mean* PER, its value is close to 7%, on ensembles of *RBF* networks when the *threshold* and *sum* normalizations are used.

Thirdly, the combination methods reviewed in this chapter are dataset dependent. In the case of *MF* networks, some of them, such as *Bayesian Combination*, highly improved some datasets, *ionos* for instance. However, their general results are not the best and sometimes they are worse than *Output average*. In the case of *RBF* networks, we can observe more clearly that there is not any direct relation between *mean* IoP and *mean* PER. Sometimes, the results show that an alternative performs better than another according to a general measurement but, in the same case, it is

worse according to the other measurement. This means that both can have similar general performance but the first way to fuse the networks fits better for some datasets and the second one fits specially better in other datasets. In the case of *RBF* networks, the best combiner for an ensemble also depended on the classification problem and the procedure applied to normalize the outputs has also a direct relation with the final performance.

In the ensembles of *MF* or *RBF* networks, the 'optimal' alternative to combine the networks will depend on the dataset and on the ensemble generated. Any combiner, even a bad one according to the general results, can provide good results for a specific dataset.

Its important to conclude by remarking that *Output average* is the first alternative that should be seriously considered if accuracy and resources required are balanced. It provides good results for any dataset when it is applied to combine ensembles of *MF* and *RBF* networks. Moreover, it is the simplest model because it is based on an unweighted average procedure. Furthermore, *Majority Voting* and *Borda Count* provide good results for ensembles of *RBF* networks and they do not depend on the normalization procedure. However, a deeper study considering all the alternatives should be applied when the experiments are focused on a specific dataset. For a specific classification problem, a punctual high improvement may be obtained by choosing the appropriate combiner and ensemble configuration

The results shown in this chapter have been published in references [MFC4, MFC5, MFC6, MFBI4, RBF3, RBF4] which are detailed in the conclusions chapter of this thesis.

Chapter 7

Adding diversity by reordering the training set

7.1 Introduction

In the researches previously done, the original *Backpropagation* algorithm has been applied as described in the original references. Although it has been modified in some ensemble methods, there has not been introduced any constraint related to the patterns. In particular, the order in which the patterns are presented to the network during training is fixed for every epoch of the learning procedure.

However, this order can be randomly changed. This means that the sequence of patterns from the training set can be exclusively set for every network because the on-line version of the *Backpropagation* algorithm is used. Change the sequence from one network to another can add diversity to the ensemble. Moreover, the sequence can be altered for every iteration or epoch of the *Backpropagation* algorithm. Although there are some ensemble alternatives in which the training set depends on the network, such as *Boosting*, we propose the application of two reordering methods to all the traditional ensembles when possible.

This chapter is organized as follows. Firstly, the two reordering alternatives will be described in section 7.2. Secondly, the experimental setup along with the databases used are introduced in section 7.3. Furthermore, the results and their discussion are shown in section 7.4. Finally, this research is finished with the conclusions.

7.2 Ensemble methods and reordering procedures

The reordering procedures that modify the sequence of patterns from the training set are described in this section. Three algorithms to reorder the training set are introduced in order to test how important reordering is on the ensemble model.

The first alternative consists in using a fix sequence of patterns for all the networks in the ensemble or a fix sequence in each network of the ensemble in the case of *Boosting* or similar methods, no change in the sequence of patterns is applied. The second procedure consists in randomly reordering the original training set at the beginning of the learning process and after to keep the initial order fixed. Finally, the last one consists in randomly reordering the training set at the beginning of each epoch of the *Backpropagation* algorithm. In the two last methods, randomly sampling without replacement is applied to alter the sequence of patterns from the training set.

As mentioned in the third chapter, all the networks of an ensemble converge into some different configurations. As shown in figure 7.1, there can be some networks which can fall in the same, or very similar, final configuration if the diversity in the training conditions is low.

In figure 7.1, some classifiers are shown in the possible classifiers space. Each dot "·" represents the network configuration at the end of an epoch. It can be seen that classifiers C_1, C_3, C_4 and C_6 have a similar final configuration. Concretely, classifiers C_1 and C_3 will have the same final configuration since they had the same configuration at the end of epoch A and the configuration will continue being the

same after epoch A if they have the same order of patterns in the training set. Moreover, classifier C_6 has a similar performance if compared to classifiers C_1 and C_3 since it has a similar "intermediate" configuration if compared to C_3. However, C_4 achieves the final configuration without having any similarity with the other three classifiers previously mentioned.

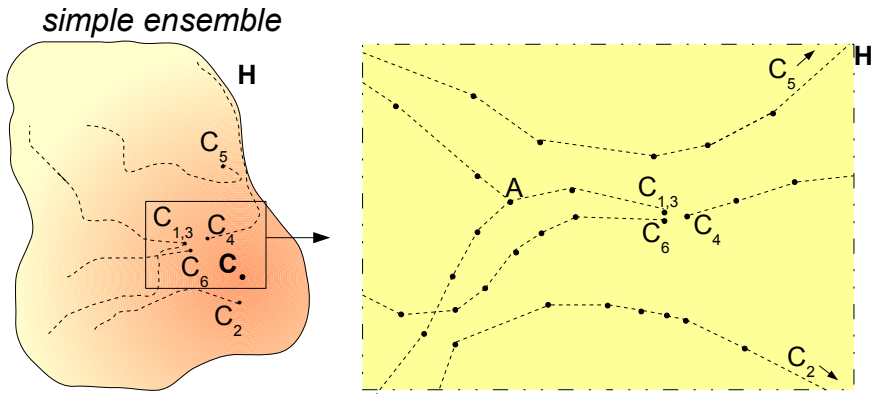


Figure 7.1: *Simple Ensemble* space and zoom

When the sequence of patterns in the training set is the same for two classifiers, they will achieve the same final network configuration if they have a common starting point, which is given by the weight initialization, or a common "intermediate" configuration. For this reason, it is important to apply a reordering algorithm in order to avoid this behavior.

Moreover, network learning is also viced by the sequence of patterns if it is kept unchanged during the whole network training. The sequence of patters should also be reordered during training because the direction to the final configuration depends on the sequence of patterns applied.

This effect is important in the case of an on-line learning algorithm, where the weights are adapted after every pattern presentation to the network and will not be interesting in a batch training algorithm where the weights are only changed after the presentation of the whole training set. In theory, on-line training and batch training should be equal if the learning step is small enough, but in practical situations this is not true and the order of the patterns in the training set has a significant effect.

7.2.1 Original training set order

This first procedure was described in chapter 2, and it is reproduced here in algorithm 7.1. With this alternative, each network is trained according to the original *Backpropagation* algorithm. Furthermore, the training set is not altered in most of the ensemble methods so all the networks are trained using the same sequence of patterns.

In this algorithm, the *Mean Squared Error* (equation 2.10) is minimized in order to adapt the parameters of the networks, usually their weights.

Algorithm 7.1 Original Network Training$\{T$,V, $net\}$

```
Set initial weights randomly
for e = 1 to epochs do
  for i = 1 to N_patterns do
    Select pattern x_i from T
    Adjust the trainable parameters
  end for
  Calculate MSE over validation set V and save weights
end for
Select epoch with lowest validation MSE
Assign best epoch configuration to the network and save network configuration
```

This is the algorithm carried out in the experiments done in the previous chapters. Although the sequence of patterns is not altered by the algorithm applied to train the network, some ensembles perform an initial *reordering* task such as *Bagging* and *Boosting*.

7.2.2 Static reordering

The *static* reordering algorithm can be applied to those ensembles in which the networks are trained independently. It is called *static* because the sequence of patterns for each individual network is kept unchanged during the whole training process.

In this case, the sequence of patterns is reordered at the begining of the training procedure of every given network. The new training set TR is drawn at random without replacement from the original training set T associated to the network and with the same number of patterns included in T.

Algorithm 7.2 Static Network Training $\{T$,V, $net\}$

```
Generate TR^net by sampling T without replacement
Set initial weights randomly
for e = 1 to epochs do
  for i = 1 to N_patterns do
    Select pattern x_i from TR^net
    Adjust the trainable parameters
  end for
  Calculate MSE over validation set V and save weights
end for
Select epoch with lowest validation MSE
Assign best epoch configuration to the network and save network configuration
```

This reordering method can not be applied to the ensembles in which all the networks are trained simultaneously, such as *CELS* or *EENCL*. According to their original references [61, 68], a pattern is presented to all the networks in each iteration. Different patterns can not be presented to different networks in a given iteration.

Moreover, it is senseless to apply them to *Boosting* and *Bagging* because in those ensembles each network has a specific training set which is not shared with the other networks. Perhaps, the reordering intrinsically applied by them may be part of their increase of performance with respect to *Simple Ensemble*.

Furthermore, *Static reordering* will be applied to the ensembles based on *Cross Validation Committee* because the training sets and the sequences of patterns differ only slightly from one network to another.

7.2.3 Dynamic reordering

The *dynamic* reordering algorithm can be applied to any ensemble. It is called *dynamic* because the sequence of patterns is altered sometimes during training, typically in every epoch. In this case, the new training set TR_e^{net} is also drawn at random without replacement from the original training set T associated to the network or ensemble and with the same number of patterns of T.

As shown in algorithm 7.3, the network will not be viced by applying the same sequence of patterns epoch after epoch because it is randomly reordered at the beginning of each epoch of *Backpropagation*.

Algorithm 7.3 Dynamic Network Training$\{T$, V , $net\}$

```
Set initial weights randomly
for e = 1 to epochs do
  Generate TR_e^net by sampling T without replacement
  for i = 1 to N_patterns do
    Select pattern x_i from TR_e^net
    Adjust the trainable parameters
  end for
  Calculate MSE over validation set V
  Save epoch weights and calculated MSE
end for
Select epoch with lowest validation MSE
Assign best epoch configuration to the network
Save network configuration
```

7.2.4 Simple Ensemble*

In the third chapter *Simple Ensemble* (*SE*) was fully described. The results of that chapter showed that it provides reasonable good performance despite its simplicity. To perform the first experiments on reordering a slight modification was introduced to *Simple Ensemble*.

This modification will be named *Simple Ensemble** or *SE**. Its algorithmic description is shown in algorithm 7.4 and its graphical description can be found in figure 7.2.

Algorithm 7.4 Simple Ensemble* $\{T\ ,\ V\ ,\ N_{networks}\ \}$

Generate a single seed value for weight initialization: *seed*
for $net = 1$ to $N_{networks}$ **do**
  Random Generator Seed = *seed*
  Network Training $\{T$,V, $net\}$
end for
Save Ensemble Configuration

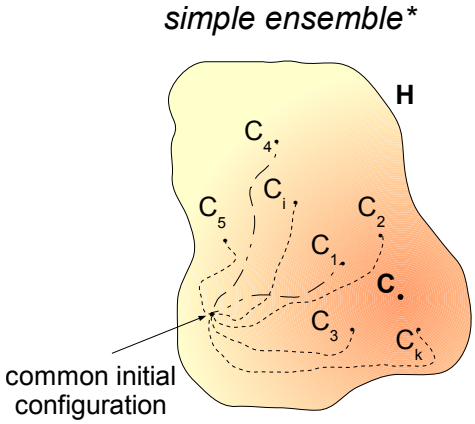


Figure 7.2: *Simple Ensemble** output space

As can be seen in the previous algorithm, the modification consists of training a set of different networks with the same initial weight initialization. Any reordering procedure described in this chapter can be applied as the *Network Training* procedure.

In the case of *Static reordering*, the networks will be trained with the same initial configuration, training parameters and learning set. However, the sequence of patterns in the training set will be different for each network despite having the same patterns.

In the case of *Dynamic reordering*, the sequence of patterns will be different for each network and for each epoch of *Backpropagation*. But the networks will also be trained with the same parameters and initial weight configuration.

The *original network training* without reordering could be also applied to *Simple Ensemble**. However, its application is totally senseless because all the networks will fall into the same final network configuration, they will be identical. For instance, if we set all the weights to the same initial values for the different networks and we do not apply reordering, it is clear that all the networks in the ensemble will have the same final configuration when we use training algorithm 7.1. The ensemble will not have any diversity and this ensemble will not improve the single network. So with ensemble *SE** we can test the effectiveness of the reordering procedures.

7.3 Experimental Setup

In this chapter two reordering algorithms, *Static Reordering Algorithm* and *Dynamic Reordering Algorithm*, have been proposed. Moreover, a new ensemble based on *Simple Ensemble* has been introduced to perform the first experiments and see the effectiveness of reordering. The main characteristics of the experiments done to test the performance of the new methods are:

- Nineteen datasets from the *UCI Repository*.
- Three reordering algorithms:
  - *Original Training Algorithm*.
  - *Static Reordering Training Algorithm*.
  - *Dynamic Reordering Training Algorithm*.
- Two basic ensembles to perform a first analysis:
  - *Simple Ensemble*.
  - *Simple Ensemble**.
- Seventeen classic ensembles to test reordering:
  - All the versions of *ATA*, *CELS*, *Decorrelated* and *EENCL*.
  - *Bagging*, *Boosting* and *CVC* and variants.
- *Output Average* as ensemble combiner.
- *MF* network as network architecture with optimized training parameters.
- Four different ensemble sizes: 3, 9, 20 and 40 networks.
- Experiments repeated ten times with different partitions of the training, validation and test sets to obtain:
  - Mean value of performance.
  - Error rate by standard error theory.
  - Paired Student's t-test.
- Three general measurements applied to the comparison:
  - Mean *Increase of Performance* (IoP).
  - Mean *Percentage of Error Reduction* (PER).
  - *Student's Paired t-test*.

The description of the nineteen datasets used in the experiments can be found in appendix A. The optimized training parameters of the networks are in appendix B.3 whereas the specific parameters of the ensemble methods are in appendix B.5. Finally, the mean IoP (equation 3.5) and the PER (equation 3.6) are used to compare the ensembles and the t-test is employed to perform the statistical comparisons.

7.4 Results and discussion

The main results related to the research done on reordering are shown in this section. As previously mentioned, a first research was performed on *Simple Ensemble* and *Simple Ensemble** whereas a second research has been focused on applying the reordering algorithms to the classic ensemble methods.

For this reason, this section has been split into three subsections. The first and second ones will show the discussion of the two versions of *Simple Ensemble*, whereas the main results of the classic ensemble methods are in the third subsection.

7.4.1 Reordering on Simple Ensemble

Firstly, the reordering algorithms were applied to *Simple Ensemble* and *Simple Ensemble** in order to test if the accuracy of the ensembles could be improved. The general results, mean IoP and mean PER across all databases with respect to a single MF network, are shown in table 7.1. The ensembles have been tested for ensembles of 3, 9, 20 and 40 networks.

Table 7.1: Reordering on Simple Ensemble

	Mean IoP				Mean PER			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
SE	4.97	5.27	5.34	5.43	21.84	23.65	23.73	24.64
SE - Static	5.32	5.67	5.78	5.83	23.79	26.41	27	27.61
SE - Dynamic	5.53	5.65	5.8	5.8	25.72	26.16	27.43	27.17
SE* - Static	5.23	5.55	5.58	5.59	23.46	25.4	25.91	26.42
SE* - Dynamic	5.45	5.81	5.91	5.74	24.44	26.65	27.08	26.37

At first sight, it can be seen that the accuracy of *Simple Ensemble* is improved when the two reordering algorithms, *static* and *dynamic*, are applied. The mean IoP is increased in a 0.5% whereas the mean PER is increased around 3%.

Moreover, the results provided by applying the reordering algorithms to *Simple Ensemble** are also better than the original *Simple Ensemble*. According to these results, changing the sequence of patterns may be considered a source of diversity stronger than selecting different initial network configurations.

Perhaps, this behavior may be given by the procedure used to set the initial weight values. The interval for the initial weight values in MF networks is small, $[-0.05, \ldots, +0.05]$ in all our experiments, according to [111]. So the differences among all the different initial networks and the diversity of using different *starting networks* or *starting points* are also small. This small interval is used in order to avoid *premature saturation*. The premature saturation occurs when the output value of a neuron is a wrong extreme value. The extreme values are given by the lowest and highest possible values, 0 and 1 in our case. Furthermore, it was shown in [112] that the saturation is likely to occur at the first epochs of training. This saturation

depends on the number of nodes, the maximum slope the transfer function and, above all, the amplitude of the initial weight values.

Figure 7.3 shows graphically how the classifiers of the ensemble are generated when the original *Simple Ensemble* and *Simple Ensemble** with reordering are applied. Obviously, the best results are obtained if reordering and selecting different initial network configurations are both applied. When *Static reordering* is applied the different networks are not trained with the same sequence of patterns so the probability of reaching a similar final configuration is lower. Furthermore, this sequence of patterns is also altered in each epoch of training when *Dynamic reordering* is used so any network from the ensemble is not viced with the same sequence during its training.

In the case of *SE** it is clear that the *Dynamic reordering* provides slightly better results than the *Static reordering* so it may be interesting to apply it. However, in the case of *SE* it is only better for the case of ensembles of low number of networks (3 networks). For these reasons, we consider that both reordering alternatives should be applied to other classic ensemble methods. Reordering provides an interesting extra source of diversity when an ensemble is generated.

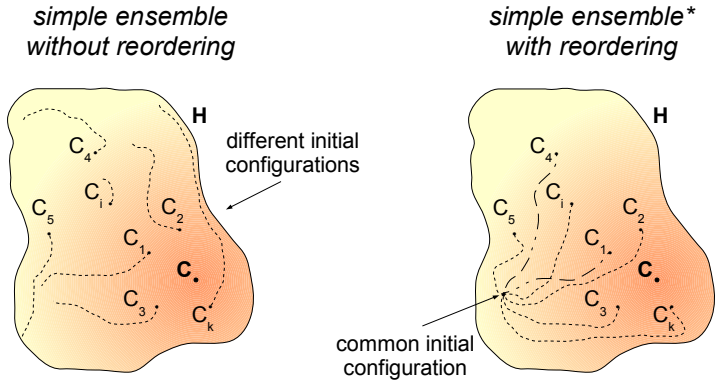


Figure 7.3: Simple ensemble and Simple Ensemble*

Furthermore, reordering on the original *Simple Ensemble* (*SE*) outperforms some classic ensemble methods such as *Cooperative Ensemble Learning System* (*CELS*) and *Decorrelated* as shown in table 7.2. There is only one case, *DECOv1* and ensembles of 9 networks, in which any reordering method does not provide the best results. In this case, the mean PER of *DECOv1* is 26.6% but the highest PER provided by *Simple Ensemble* is 26.41%.

Moreover, reordering is better than *Bagging* and the best *Boosting* variants in some cases. In the case of *Bagging*, reordering on *SE* is better when the number of networks in the ensemble is low (3 networks). Whereas, it is clearly better than *Adaboost* in all the cases. Finally, both reordering alternatives on *SE* provide better general results that the best boosting variants, *Aveboost* and *Conserboost*, for the case of low and medium number of networks in the ensemble (3 and 9 networks).

Table 7.2: Comparing reordering algorithms to classic ensemble methods

	Mean IoP				Mean PER			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
SE - Static	5.32	5.67	5.78	5.83	23.79	26.41	27	27.61
SE - Dynamic	5.53	5.65	5.8	5.8	25.72	26.16	27.43	27.17
SE* - Static	5.23	5.55	5.58	5.59	23.46	25.4	25.91	26.42
SE* - Dynamic	5.45	5.81	5.91	5.74	24.44	26.65	27.08	26.37
CELS	4.72	5.42	5.46	5.53	21.51	23.73	25.75	26.35
DECOv1	5.48	5.78	5.85	5.83	24.76	26.6	26.99	27.03
DECOv2	5.46	5.53	5.58	5.78	24.87	25.71	25.92	26.4
Bagging	5.09	5.78	5.95	5.76	22.93	27.55	28.25	27.69
Adaboost	3.69	4.55	4.93	4.98	15.4	19.5	22.96	24.54
Aveboost	4.33	5.57	5.95	6.04	18.26	26.11	27.12	26.53
Conserboost	4.42	5.31	5.65	6.06	19.72	25.63	26.62	27.84

The results shown in the previous table are important because ensembles with good performance can be generated with a simpler procedure when they are compared to other classic ensembles. Moreover, there are a few cases, specially for the case of boosting variants, in which reordering on *SE* is better according to the mean PER but a boosting alternative is better according to the mean IoP. For instance, in the case of 20 networks in the ensemble the mean IoP for *Dynamic reordering* on *Simple Ensemble* is 5.8% and this value for *Aveboost* is 5.95%, whereas the the mean PER values are 27.43% and 27.12% respectively. Although both methods provide general good results, *Aveboost* specially improves the results of some datasets whereas reordering specially improves the results of other datasets. For this reason, *Dynamic reordering* should be applied to other ensemble methods, such as *Boosting*, because the sources of diversity are different.

Although all the results shown improve the original *Simple Ensemble*, the general measurements applied do not provide any statistical information so the *t-test* was employed to compare the results obtained with the original *Simple Ensemble*, the statistical results are shown in table 7.3.

Table 7.3: Statistical results - Reordering on *Simple Ensemble*

Methods	**measure**	**3-net**	**9-net**	**20-net**	**40-net**
SE traditional vs SE static	t-**value**	-1.66	-2.43	-3.32	-2.79
	α	0.1	0.016	0.0011	0.0059
SE traditional vs SE dynamic	t-**value**	-2.42	-2.04	-2.97	-2.4
	α	0.017	0.043	0.0034	0.017
SE traditional vs SE* static	t-**value**	-1.25	-1.65	-1.64	-1.04
	α	0.21	0.1	0.1	0.3
SE traditional vs SE* dynamic	t-**value**	-2.3	-3.22	-3.81	-2.05
	α	0.022	0.0015	0.00019	0.042
SE* static vs SE* dynamic	t-**value**	-1.27	-2.0	-2.74	-1.34
	α	0.21	0.047	0.0067	0.18

According to the results shown in table 7.3, applying *Dynamic reordering* to *Simple Ensemble* and *Simple Ensemble** improve the original *Simple Ensemble* and the differences are statistically significant because the t-values are negative and α is lower than 0.05. Moreover, *Static reordering* applied to *Simple Ensemble* also statistically improves the original ensemble in most of the cases. However, *Static reordering* applied to *Simple Ensemble** slightly improves the original *Simple Ensemble* but the differences are not statistically significant.

Moreover, *Static reordering* is statistically compared to *Dynamic reordering* for the ensembles generated by *Simple Ensemble** in the last row. The statistical results show that *Dynamic reordering* is better than *Static reordering* because the t-values are negative. However, their differences are only statistically significant (α lower than 0.05) for the cases of ensembles of 9 and 20 networks.

As a final conclusion, *Dynamic reordering* should be applied to any ensemble whereas *Static reordering* should be used with those ensembles in which the networks are not trained simultaneously or reordering is not implicit. In both cases, the use of different initial network configurations must be also applied. Both, reordering and different weight initializations, should be used simultaneously.

7.4.2 Reordering and weight initialization strategy

As it has been previously shown, the interval for initial weight values in *MF* networks is $[-0.05, \ldots, +0.05]$ according to [111]. To test if changing the sequence of patterns can be considered a better source of diversity than selecting different initial network configurations, we have repeated the experiments using different intervals for initial weight configuration. The results of these experiments are in tables 7.4 to 7.6

Table 7.4: Reordering on Simple Ensemble with initial interval $[-0.25, \ldots, +0.25]$

	Mean *IoP*				Mean *PER*			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
SE	5.36	5.68	5.66	5.72	24.31	25.32	25.55	25.84
SE - Static	5.56	5.89	6	5.97	25.63	27.49	28.38	28.3
SE - Dynamic	5.67	5.96	5.97	5.9	25.84	27.71	28.17	27.72
SE* - Static	5.33	5.66	5.68	5.76	24.03	26.43	26.84	27.26
SE* - Dynamic	5.42	5.52	5.37	5.43	24.5	25.07	24.28	25.03

Table 7.5: Reordering on Simple Ensemble with initial interval $[-0.5, \ldots, +0.5]$

	Mean *IoP*				Mean *PER*			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
SE	5.56	5.82	5.8	5.78	24.99	26.33	25.97	26.02
SE - Static	5.55	5.93	5.99	5.99	25.04	28.01	28.42	28.5
SE - Dynamic	5.75	5.87	6.02	6.02	26.82	27.39	27.97	28.33
SE* - Static	5.75	5.87	6.02	6.02	26.82	27.39	27.97	28.33
SE* - Dynamic	5.15	5.14	5.26	5.23	23.63	23.53	24.54	24.39

Table 7.6: Reordering on Simple Ensemble with initial interval $[-1.5, \ldots, +1.5]$

	Mean *IoP*				Mean *PER*			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
SE	5.61	5.93	6.07	6.08	25.17	27.64	28.14	28.1
SE - Static	5.52	6.13	6.18	6.24	25.42	29.38	29.48	29.95
SE - Dynamic	5.82	6.16	6.23	6.22	27.32	29.29	29.64	29.46
SE* - Static	5.1	5.33	5.41	5.55	22.34	24.35	24.88	25.7
SE* - Dynamic	5.03	5.16	5.19	5.21	22.99	24.18	24.33	24.22

Firstly, according to the three previous tables, the performance of *Simple Ensemble* with or without reordering seems to increase as the interval for initial weight values becomes higher. It means that there can be more initial different networks and diversity may be increased when the interval becomes higher. In fact, the highest results of *Simple Ensemble* (mean IoP around 6.2% and mean PER close to 30%) are obtained by using reordering with high sized ensembles (40 networks) and the highest interval ($[-1.5, \cdots, +1.5]$). On the other hand, the effect of premature saturation described above seems to be less important in an ensemble than in a single network.

Secondly, *Simple Ensemble** with *Dynamic reordering* tends to perform worse as the interval increases. *Simple Ensemble** with *Static reordering* performs better than *Simple Ensemble* for the first interval, $[-0.25, \ldots, 0.25]$ (except for 3 networks where the results are similar), and the second interval, $[-0.5 \ldots 0.5]$ but it performs worse for the last interval $[-1.5, \ldots, +1.5]$. In general, *SE** fits better when the interval used to set the initial weight values is small. If this interval is high, such as $[-1.5, \ldots, +1.5]$, the use of different initial networks is better source of diversity than reordering because there are more possible *starting points* and the initial networks of the ensemble are more different.

Finally, the best results are obtained if both sources of diversity, different initial configuration for weights and reordering the sequence of patterns of the training set, are applied together to generate ensembles.

7.4.3 Reordering on classic ensemble methods

The results have shown that reordering has a positive effect on *SE* so both reordering algorithms, *static* and *dynamic*, have been applied to the classic methods.

Initially, all the ensemble models described in chapters 4 and 5 have been considered to be used along with the reordering algorithms. But the worst ones, such as *EVOL* and *OLA*, are not included and only the most important *Boosting methods* are considered. The results related to the original network training of the chosen ensembles correspond to the original results shown in those chapters. These results are reproduced in tables 7.7 and 7.8 to perform the comparison easier. For each ensemble alternative, we compare the results of the original version with respect to the results when the reordering algorithms are employed.

Firstly, *Static reordering* has been applied to the classic ensemble models in which the networks are trained independently and the sequence of patterns has not been altered by the ensemble. *Bagging* and *Boosting* generate a specific training set for each network by sampling so *Static reordering* is implicit. Moreover, *Static reordering* can not be applied to *CELS* and *EENCL* because all the networks are trained at the same time and the sequence of patterns must be the same for all the networks according to the original references.

The general results, mean IoP and PER, of the classic ensembles along with the results related to applying *Static reordering* to them are shown in table 7.7.

Table 7.7: Static reordering on classical ensemble methods

	Mean IoP				Mean PER			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
CVCv1	4.38	5.53	5.35	5.53	20.66	24.3	24.23	24.21
CVCv1 static	4.91	5.46	5.56	5.51	21.52	23.62	23.16	23.90
CVCv2	5.48	5.95	5.98	5.85	25.73	27.09	25.34	23.7
CVCv2 static	5.48	6.06	6.02	5.96	25.28	27.69	25.89	24.96
CVCv2.5	5.48	6.17	5.94	5.94	24.25	27.85	26.65	27.55
CVCv2.5 static	5.69	6.13	6.23	6.22	25.65	28.00	28.87	28.95
CVCv3	5.56	5.96	6.17	6.16	24.06	26.59	28.35	28.68
CVCv3 static	5.87	6.19	6.24	6.23	26.58	27.52	28.83	28.90
DECOv1	5.48	5.78	5.85	5.83	24.76	26.6	26.99	27.03
DECOv1 static	5.30	5.67	6.02	5.96	24.46	27.21	28.54	28.24
DECOv2	5.46	5.53	5.58	5.78	24.87	25.71	25.92	26.4
DECOv2 static	5.36	5.67	5.86	5.72	24.69	26.17	27.22	26.44
ATA-LE	4.58	4.91	5.01	4.77	19.44	22.05	22.18	20.22
ATA-LE static	4.46	5.05	5.02	4.66	18.14	22.36	21.36	19.99
ATA-BE	4.42	5.24	5	5.02	19.97	23.4	22.58	21.59
ATA-BE static	4.28	5.21	5.03	5.06	17.61	22.95	21.10	21.88

The results of the previous table show that the ensembles based on *Cross-Validation Committee* can be improved, in general, if *Static reordering* is applied. However, *CVCv1* is the unique exception because *Static reordering* provides similar or worse results with respect to the original version. In fact, *Static reordering* provides the best overall results when it is applied to *CVCv2.5* and *CVCv3*, the mean IoP is nearly 6.25% and the mean PER is close to 29% for ensembles of 40 networks.

In the case of the alternatives based on *Decorrelated*, *Static reordering* slightly improves them for ensembles of 9, 20 and 40 networks according to the mean PER. However, there are two cases, ensembles of 9 networks trained with *DECOv1* and ensembles of 40 networks trained with *DECOv2*, in which the original training provides better IoP and worse PER. In those cases, both have similar general performance.

For the methods based on *Adaptive Training Algorithm*, the *Static reordering* algorithm can not improve the original ensembles (*ATA-BE* and *ATA-LE*), in general, and their general results are similar.

Table 7.8 shows the results of the original ensembles along with the results related to applying *Dynamic reordering* to the classic ensembles. In this case, there are more alternatives because *Dynamic reordering* can be applied to any ensemble.

Table 7.8: Dynamic reordering on classical ensemble methods

	Mean *IoP*				Mean *PER*			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Bagging	5.09	5.78	5.95	5.76	22.93	27.55	28.25	27.69
Bagging dyn	5.11	5.67	5.80	5.84	22.71	25.99	27.21	27.88
Bagnoise	3.95	4.03	3.98	4.18	13.32	12.44	13.12	13.66
Bagnoise dyn	3.88	4.08	4.01	3.97	12.88	13.64	13.47	13.18
CVCv1	4.38	5.53	5.35	5.53	20.66	24.3	24.23	24.21
CVCv1 dyn	5.13	5.72	5.51	5.33	24.16	25.06	22.64	21.65
CVCv2	5.48	5.95	5.98	5.85	25.73	27.09	25.34	23.7
CVCv2 dyn	5.56	6.41	6.04	5.86	26.73	30.10	25.90	24.86
CVCv2.5	5.48	6.17	5.94	5.94	24.25	27.85	26.65	27.55
CVCv2.5 dyn	5.84	6.10	6.04	6.15	26.45	27.59	28.05	28.38
CVCv3	5.56	5.96	6.17	6.16	24.06	26.59	28.35	28.68
CVCv3 dyn	5.66	6.07	6.29	6.13	25.73	28.03	28.77	28.05
CELS	4.72	5.42	5.46	5.53	21.51	23.73	25.75	26.35
CELS dyn	5.44	5.93	6.15	5.91	25.68	27.62	29.31	29.03
DECOv1	5.48	5.78	5.85	5.83	24.76	26.6	26.99	27.03
DECOv1 dyn	5.51	5.81	5.87	5.98	25.64	26.86	27.53	28.23
DECOv2	5.46	5.53	5.58	5.78	24.87	25.71	25.92	26.4
DECOv2 dyn	5.35	5.67	5.69	5.77	23.62	24.73	24.93	25.24
Adaboost	3.69	4.55	4.93	4.98	15.4	19.5	22.96	24.54
Adaboost dyn	4.00	4.88	4.92	4.93	16.05	23.03	22.86	23.05
Aveboost	4.33	5.57	5.95	6.04	18.26	26.11	27.12	26.53
Aveboost dyn	4.61	5.73	6.02	5.93	18.76	25.35	27.91	26.72
Aggreboost	3.56	4.79	5.53	5.83	14.82	20.26	25.27	26.56
Aggreboost dyn	3.54	4.70	5.56	5.95	13.58	20.43	25.86	27.50
Conserboost	4.42	5.31	5.65	6.06	19.72	25.63	26.62	27.84
Conserboost dyn	4.37	5.43	6.00	6.22	19.70	25.05	28.49	29.80
ATA-LE	4.58	4.91	5.01	4.77	19.44	22.05	22.18	20.22
ATA-LE dyn	4.23	5.03	4.82	4.62	17.12	22.20	20.55	19.59
ATA-BE	4.42	5.24	5	5.02	19.97	23.4	22.58	21.59
ATA-BE dyn	4.47	5.23	5.36	5.27	19.04	24.50	24.99	24.20
EENCL-LG	4.41	4.06	4.52	4.2	16.87	14.93	15.99	17.71
EENCL-LG dyn	4.29	4.49	4.44	4.61	17.14	16.40	18.92	18.54
EENCL-BG	4.89	4.82	4.67	5.07	20.42	19.6	19.02	22.36
EENCL-BG dyn	5.15	5.05	5.12	4.77	22.86	19.92	21.83	20.89

According to table 7.8, *Dynamic reordering* does not have a high positive impact on the ensembles based on *Bagging* but the ensembles based on *Cross-Validation Committee* can be improved if *Dynamic reordering* is applied. *CVCv1* is the unique *CVC* ensemble in which *Dynamic reordering* provides considerable low performance for the case of a high number of networks in the ensemble (20 and 40 networks).

Furthermore, *Dynamic reordering* provides better results than *Static reordering* in most of the cases. However, *Static reordering* performs better on *CVC* when ensembles of a high number of networks are generated (typically 40 networks and sometimes 20).

In the case of the ensembles generated by *CELS*, *Dynamic reordering* provides a high improvement in the performance with respect to the original ensemble. The best results for *CELS* are provided by ensembles of 20 networks using *Dynamic reordering* (*mean IOP* equal to 6.15% and *mean PER* equal to 29.31%).

For the methods based on *Decorrelated*, *Dynamic reordering* slightly improves *DECOv1* whereas the results related to *DECOv2* are slightly worse. For them, *Static reordering* has been better except for the case of 3 networks in the ensemble.

Dynamic reordering works well on boosting. It improves the results provided by the best boosting alternatives for medium and high sized ensembles (typically 20 and 40 networks in the ensemble) and it provides the best overall results of these methods. However, it only improves the results of *Adaboost* for the cases of 3 and 9 networks.

The results related to *Adaptive Training Algorithm* show that the *Dynamic reordering* algorithm can not improve the original method *ATA-LE* except for ensembles of 9 networks. However, the version we proposed of *ATA*, *ATA-BE*, performs better when *Dynamic reordering* is applied, especially for ensembles of 20 and 40 networks.

Applying *Dynamic reordering* to *EENCL* provides, in general, better results that the traditional versions according to the PER values. Although the traditional training is the best choice for *EENCL-BG* and 40 networks (highest PER for the original ensembles), its results are overcome by *Dynamic reordering* with only 3 networks.

Furthermore, we have applied the *Student's t-test* in order to statistically compare the original training to reordering in tables 7.9 to 7.14. The first table introduces the statistical results for *Bagging* and *Bagnoise*.

Table 7.9: Statistical results - Reordering on classic ensemble methods (I)

Methods	**measure**	3-**net**	9-**net**	20-**net**	40-**net**
Bagging Vs. Bagging dyn	t-**value**	−0.07	0.73	1.11	−0.62
	α	0.93	0.46	0.26	0.53
Bagnoise Vs. Bagnoise dyn	t-**value**	0.30	−0.07	−0.14	1.23
	α	0.75	0.94	0.88	0.21

The statistical results of table 7.9 show that the differences between the original training and the reordered version are not statistically significant for *Bagging* and *Bagnoise*. There are two cases in *Bagging*, 9 and 20 networks in the ensemble, in which the original training procedure is slightly better than *Dynamic reordering*. In *Bagnoise* there are also two cases, 3 and 40 networks in the ensemble, in which *Dynamic reordering* is slightly worse than the original training procedure because the t-values are positive. This behavior was expected because the original *Bagging* and *Bagnoise* implicitly performed *Static reordeing*.

The statistical results for the ensembles based on *CVC* are shown in table 7.10.

Table 7.10: Statistical results - Reordering on classic ensemble methods (II)

Methods	measure	3-net	9-net	20-net	40-net
CVCv1 Vs. CVCv1 stat	t-**value**	−2.34	0.37	−1.05	0.11
	α	0.02	0.71	0.29	0.92
CVCv1 Vs. CVCv1 dyn	t-**value**	−2.79	−0.78	−0.81	0.78
	α	0.01	0.44	0.42	0.43
CVCv2 Vs. CVCv2 stat	t-**value**	−0.02	−0.78	−0.22	−0.66
	α	0.98	0.43	0.83	0.51
CVCv2 Vs. CVCv2 dyn	t-**value**	−0.45	−2.96	−0.32	−0.06
	α	0.66	0.0034	0.75	0.95
CVCv2.5 Vs. CVCv2.5 stat	t-**value**	−1.07	0.28	−2.18	−1.74
	α	0.29	0.78	0.03	0.083
CVCv2.5 Vs. CVCv2.5 dyn	t-**value**	−1.77	0.44	−0.72	−1.49
	α	0.08	0.66	0.48	0.14
CVCv3 Vs. CVCv3 stat	t-**value**	−1.42	−1.31	−0.45	−0.64
	α	0.16	0.19	0.65	0.53
CVCv3 Vs. CVCv3 dyn	t-**value**	−0.44	−0.58	−0.88	0.2
	α	0.66	0.56	0.38	0.84

According to table 7.10, the differences between the original *CVC* and the reordered versions are not, in general, statistically significant. However, there are a few cases in *CVC* in which the value of α is lower than 0.05 and the results are statistically significant. Both new reordering algorithms are statistically better than the original ensemble for *CVCv1* and 3 networks in the ensemble. There are other two cases, *CVCv2* with 9 networks and *CVCv2.5* with 20 networks, in which the differences between one of the two proposed alternatives with respect to the original ensemble are also statistically significant.

In table 7.11, the statistical results for *CELS* are introduced. In this table, it can be observed that applying *Dynamic reordering* to *CELS* improves the original ensemble and their differences are statistically significant in all the cases (negative t-values and α lower than 5%). It is important to mention, that *CELS* is the only traditional ensemble (without considering *Simple Ensemble*) which has been outperformed by a reordering algorithm (*Dynamic reordering*) for all the ensemble sizes and the differences are statistically significant also for all the ensemble sizes.

Table 7.11: Statistical results - Reordering on classic ensemble methods (III)

Methods	measure	3-net	9-net	20-net	40-net
CELS Vs. CELS dyn	t-**value**	−3.27	−3.10	−4.02	−2.21
	α	0.001	0.002	0.00008	0.028

The statistical results of the ensembles generated with *DECOv1* and *DECOv2* are shown in table 7.12.

Table 7.12: Statistical results - Reordering on classic ensemble methods (IV)

Methods	measure	3-net	9-net	20-net	40-net
DECOv1 Vs. DECOv1 stat	*t*-value	0.84	0.79	−1.68	−1.07
	α	0.4	0.43	0.094	0.29
DECOv1 Vs. DECOv1 dyn	*t*-value	−0.14	−0.16	−0.36	−1.07
	α	0.89	0.88	0.72	0.29
DECOv2 Vs. DECOv2 stat	*t*-value	0.58	−0.94	−1.78	0.37
	α	0.57	0.35	0.076	0.72
DECOv2 Vs. DECOv2 dyn	*t*-value	0.64	−0.76	−0.53	0.06
	α	0.52	0.45	0.6	0.95

It can be seen, in table 7.12, that the differences between the original versions of *DECO* and their reordered versions are not statistically significant. The reordered versions of *DECO* are slightly better because the *t*-values are negative and α is higher than 0.05 in most of the cases.

Table 7.13 shows the statistical results for ensembles based on *ATA* and *EENCL*. According to this table, the difference between the performance of these methods with the traditional training and reordering algorithms is not statistically significant because α is higher than the threshold value (0.05).

Table 7.13: Statistical results - Reordering on classic ensemble methods (V)

Methods	measure	3-net	9-net	20-net	40-net
ATA-LE Vs. ATA-LE stat	*t*-value	1.23	−0.4	−0.012	0.736
	α	0.22	0.68	0.99	0.46
ATA-LE Vs. ATA-LE dyn	*t*-value	1.74	−0.3	0.83	0.62
	α	0.08	0.76	0.4	0.53
ATA-BE Vs. ATA-BE stat	*t*-value	0.9	0.56	0.17	−0.31
	α	0.37	0.58	0.87	0.76
ATA-BE Vs. ATA-BE dyn	*t*-value	0.29	0.36	−1.49	−1.2
	α	0.77	0.72	0.14	0.23
EENCL-LG Vs. EENCL-LG dyn	*t*-value	0.53	−1.82	0.35	−1.58
	α	0.6	0.07	0.73	0.12
EENCL-BG Vs. EENCL-BG dyn	*t*-value	−1.2	−1.13	−1.95	1.54
	α	0.23	0.26	0.052	0.12

The *t*-values related to both versions of *ATA* show that in more than half of cases (62.5%), the orginal ensemble is slightly better than the proposed alternatives with reordering because the *t*-value is positive.

For *EENCL*, the *t*-values show that the proposed versions are better than the original ensemble in 5 of 8 cases (62.5%). Moreover, the α value for the case of *EENCL* is low for 9 networks (*EENCL-LG*) and 20 networks (*EENCL-BG*). In these two cases, *Dynamic reordering* is better than the original ensemble and their differences are almost significant.

Finally, statistical results related to *Adaptive Boosting* and the best variants (*Aveboost*, *Aggreboost* and *Conserboost*) are in table 7.14. According to the table, the difference between their performance with the traditional training and the reordering algorithms is not statistically significant because α is greater than 0.05. However, there is only one case in which the differences are statistically significant (*Conserboost* and 20 networks) and there are some cases (mainly in *Adaboost*) in which α is quite close to this percentage (5%) and the differences are almost significant.

Table 7.14: Statistical results - Reordering on classic ensemble methods (VI)

Methods	measure	3-net	9-net	20-net	40-net
Adaboost Vs. Adaboost dyn	t-**value**	−1.58	−1.57	−0.84	−1.78
	α	0.12	0.12	0.4	0.08
Aveboost Vs. Aveboost dyn	t-**value**	−1.11	−0.95	−0.43	0.68
	α	0.27	0.34	0.67	0.5
Aggreboost Vs. Aggreboost dyn	t-**value**	−0.09	0.54	−0.06	−0.75
	α	0.93	0.59	0.96	0.45
Conserboost Vs. Conserboost dyn	t-**value**	0.23	−0.78	−2.09	−1
	α	0.81	0.44	0.038	0.32

In general, the results of reordering on classic ensemble methods slightly improve the results provided by the original methods without reordering. Moreover, the difference in performance, in general, is not statistically significant because in a 91% of the cases shown the value of α is greater than 0.05, only in 9 cases α denotes a significant difference.

However, there are some cases in which α is close to this "limit" value. We think that we may obtain significant differences if we increase the number of results used to do the statistical test, i.e., the number of databases and the number of partitions of the original learning set in training, validation and test sets.

For instance, the experiments are repeated n times with different partitions on the training, validation and test sets. In all our experiments performed, n has been set to 10. If n is increased to 20, we may have more precise general results and we will have more data in order to perform the statistical tests. In the following table, we show the statistical test applied to compare the traditional *CVCv3* with its reordered versions. To perform this test, in this case, we have repeated the experiments 20 times as described above.

Table 7.15: Statistical results - Reordering on *CVCv3* after repeating the experiments 20 times

Methods	measure	3-net	9-net	20-net	40-net
CVCv3 Vs. CVCv3 stat	t-**value**	−2.935	−2.585	−1.468	−2.558
	α	0.004	0.01	0.143	0.011
CVCv3 Vs. CVCv3 dyn	t-**value**	−2.332	−1.694	−1.308	−1.563
	α	0.02	0.091	0.192	0.119

As we can see, the value of α in the previous table is lower than for table 7.10. Moreover, α is lower than 0.05 in half of cases so the new alternatives are statistically significant in these cases. Repeating the experiments 20 times and increasing the number of databases will provide more data to the statistical test and we can realize that the difference of reordering with respect to traditional training tends to be statistically significant.

7.5 Conclusions

In this chapter an exhaustive comparison about reordering the training set on the ensemble methodology has been performed. Two reordering procedures are successfully proposed. Important conclusions can be derived from this research.

Firstly, the results showed that applying reordering to *Simple Ensemble* provides good results. *Dynamic reordering* fitted better in the cases of low number of networks in the ensemble (3 and 9 networks) and *Static reordering* fitted better on medium-high sized ensembles (20 and 40 networks).

Secondly, reordering the sequence of patterns is an important source of diversity as reported with the results of *Simple Ensemble**. Training a set of networks with the same initial configuration and with a reordering algorithm provides statistically better results than the original *Simple Ensemble* algorithm when a small interval for weight initialization is used to generate the initial networks.

However, the use of different initial configuration for the networks is also important, specially when the interval for weight initialization is high. Although *Simple Ensemble* has been improved by using a higher interval, we have employed the small interval suggested in [111], $[-0.05, \ldots, +0.05]$, in all the experiments performed in this thesis in order to avoid premature saturation.

Thirdly, the best results with *Simple Ensemble* are obtained when reordering is applied to networks with different initial weight configurations. In this case, two different sources of diversity, the use of different initial networks and reordering, are properly mixed.

The results of these firsts experiments were a vital aim to apply reordering to the classic ensemble methods described in the previous chapters. Some ensemble methods were omitted since their results with the original training were not good enough.

Fourthly, applying *Static reordering* and *Dynamic reordering* slightly improve the classic ensemble methods. The *Dynamic reordering* algorithm is a better alternative since it provides the best results in most of the cases and it can be applied to any ensemble method. But *Static reordering* provides better results in *CVC* methods for the case of ensembles of 40 networks. Although the results are only slightly improved, reordering can be considered an important step in order to design better ensemble methods. In fact, reordering on *Simple Ensemble* can improve the results of good classic ensemble methods.

Finally, the results related to the best classic ensemble methods, which were reported in chapters 4-5, were mean IoP lower than 6.2% and the mean PER between 28% and 29%. The best overall results of this chapter are provided by applying *Dynamic reordering* to *CVCv2*, where the mean IoP is 6.41% and the mean PER is close to 30.1%. Although the results of the classic ensembles are quite good, they may be improved by refining the ensemble methods step by step.

The results related to reordering have been published in reference [MFR1] which is detailed in the conclusions chapter.

Chapter 8

Improving Boosting methods

8.1 Introduction

In chapters 4 and 5, it has been shown that *Boosting variants* and different versions of *Cross-Validation* are successfully applied to train ensembles of neural networks.

Some authors like Breiman [81], Kuncheva [82] or Oza [83] have deeply studied *Adaptive Boosting* (*Adaboost*) and successfully improved it by modifying the equation used to update the sampling distribution or by adding new constraints to the original algorithm. Although these variants of boosting perform better than *Adaboost* in general, there are some important cases in which the variants perform worse because the results depend on the database and on the ensemble size.

In this chapter the following two new boosting models and a new methodology to improve boosting are successfully proposed:

- *Averaged-Conservative Boosting*.
- *Weighted-Conservative Boosting*.
- *Cross-Validated Boosting* methodology.

This chapter is organized as follows. Firstly, the new ensemble methods will be described in sections 8.2 to 8.4. Secondly, the experimental setup is introduced in section 8.5. Thirdly, the results and their discussion is shown in section 8.6.

8.2 Averaged-Conservative Boosting

Averaged-Conservative Boosting, henceforth *ACB*, is a new boosting variant, proposed by us, in which *Averaged Boosting* and *Conservative Boosting* are mixed in order to design a more robust boosting alternative. This new ensemble, described in algorithm 8.1 at the end of this section, is based on the principles of boosting described in chapter 5 (subsection 5.2.2).

As we have previously mentioned, *Adaptive Boosting*, *Adaboost*, is a well known ensemble. Its main problem is that after training some networks the training sets tend to be overfitted with hard to learn patterns by a sampling distribution, *Dist*. Figure 8.1 graphically describes the strength of the equation applied in *Adaboost*.

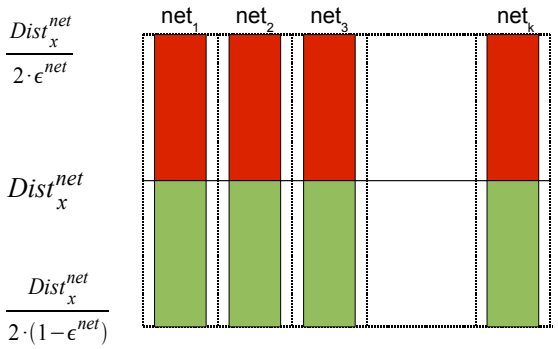
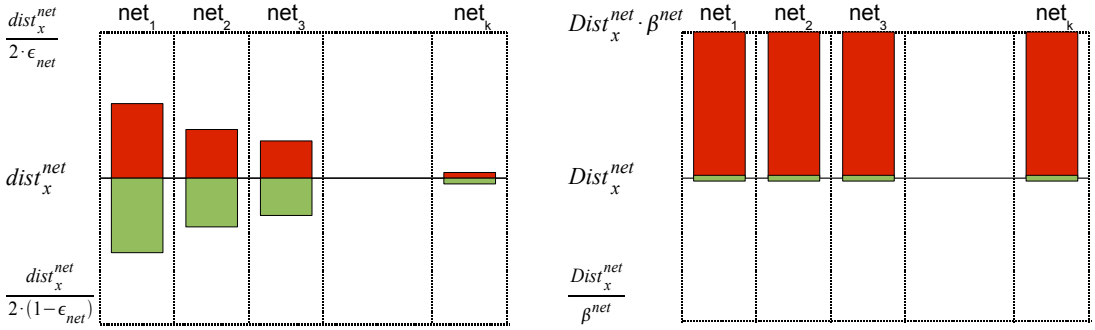


Figure 8.1: Updating the sampling distribution in *Adaboost*

The error, ϵ, and other variables (such as β and the missclassification vector $miss$) correspond to the original ensembles and they are reproduced here in algorithm 8.1.

In the previous figure, the green and red bars denote the new value of the distribution for sampling the patterns and building the new training set for correctly and incorrectly classified patterns respectively. Analyzing the figure, it can be noticed that the strength keeps unchanged network after network. So the values of the distribution, $Dist$, can shoot up or down quickly. Some authors proposed in [82, 83, 84] softer versions of *Adaboost* in order to slow the process and achieve better results. *Aveboost* and *Conserboost* are the best alternatives. Figure 8.2 graphically describes the strength of the equations used in *Aveboost* (left part) and *Conserboost* (right part).



(a) (b)

Figure 8.2: Updating the sampling distribution in *Aveboost* (a) and *Conserboost* (b)

In *Aveboost*, the strength of the equation applied becomes softer and softer as new networks are added to the ensemble. After training some networks, the strength is marginal and the distributions of two consecutive networks are similar. In this method, the equation to update the sampling distribution is given by:

$$Dist_x^{net+1} = \frac{net \cdot Dist_x^{net} + C_x^{net}}{net + 1} \tag{8.1}$$

where:

$$C_x^{net} = Dist_x^{net} \cdot \begin{cases} \frac{1}{2\cdot\epsilon^{net}} & \text{if } x \text{ is incorrectly classified by } net \\ \frac{1}{2\cdot(1-\epsilon^{net})} & otherwise \end{cases} \tag{8.2}$$

In *Conserboost* the value of the distribution of the incorrectly classified patterns is increased whereas the value of the correctly classified is kept unchanged. It is considered more *conservative* or softer than the equation used in *Adaboost* because the probability of selecting the correctly classified patterns is not changed. The equation used to update the sampling distribution, $Dist_x^{net+1}$, is given by equation 8.3.

$$Dist_x^{net+1} = Dist_x^{net} \cdot \left(\beta^{net}\right)^{miss_x^{net}} \tag{8.3}$$

Moreover, the summatory of all the elements of the distribution must be always 1. In the new distribution, $Dist^{net+1}$, the sum of all the values will be higher that 1 because the value of the incorrectly classified patterns is increased with respect to $Dist^{net}$ and the other values are kept unchanged. A normalization procedure is applied in order to satisfy that the summatory of all the elements of $Dist^{net+1}$ will be also 1. This procedure is given by the following equation:

$$Dist_x^{net+1} = \frac{Dist_x^{net+1}}{\sum_{i=1}^{N_{patterns}} Dist_i^{net+1}} \tag{8.4}$$

In this normalization, all the values of the distribution are decreased by a factor which corresponds to the summatory of all the elements of $Dist^{net+1}$ before the normalization and this summatory will never be lower than 1. For this reason, the strength of the equation is softer than in *Adaboost* after applying the normalization of the distribution because all the values of the distribution are decreased during the normalization after updating its values. The strength of *Conserboost* after normalization is graphically shown in figure 8.3.

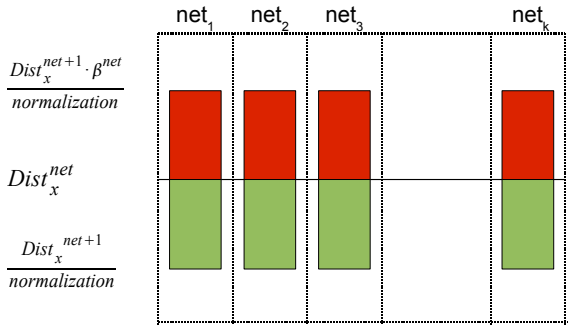


Figure 8.3: Updating the sampling distribution in *Conserboost* after normalization

The previous methods, *Aveboost* and *Conserboost*, improve the results provided by *Adaboost* separately. The origins of both methods are quite different so we thought that a new approach based on both can provide good performance and robustness.

In the method proposed, *ACB*, the sampling distribution is updated by mixing the averaged equation of *Aveboost*, equation 8.1, and the relaxed equation of *Conserboost*, equation 8.3. The main idea of this method is to apply a equation which weights the values of the former distribution as in *Aveboost*, but the new values are calculated as in *Conserboost* giving priority to the former values. The values of the new distribution are given by the weighted equation 8.5.

$$Dist_x^{net+1} = \frac{net \cdot Dist_x^{net} + C_x^{net}}{net + 1} \tag{8.5}$$

Where the value of C_x is now given by:

$$C_x^{net} = Dist_x^{net} \cdot \left(\beta^{net}\right)^{miss_x^{net}} \tag{8.6}$$

Figure 8.4 shows the graphical representation of the strength related to this new ensemble method.

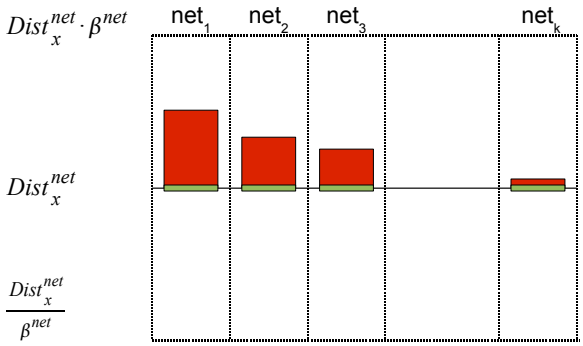


Figure 8.4: Updating the sampling distribution in *ACB*

In the ensemble comparison performed in a previous chapter, it can be seen that *Conserboost* performs better than *Aveboost* for the case of ensembles of 3 networks, whereas *Aveboost* performs better for the cases of ensembles of 9, 20. In the case of 40 networks, both have similar performance but *Conserboost* is slightly better. These results are reproduced in table 8.1.

Table 8.1: General Results of *Aveboost* and *Conserboost*

	Mean *IoP*				**Mean** *PER*			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Aveboost	4.33	5.57	5.95	6.04	18.26	26.11	27.12	26.53
Conserboost	4.42	5.31	5.65	6.06	19.72	25.63	26.62	27.84

As can be seen in the previous figure and table, the strength of the new equation initially corresponds to *Conserboost*, which performs better for small ensembles, and it becomes similar to *Aveboost*, which performs better for high ensembles. In this way, the behavior of two important methods is mixed into a single one. The first networks, the basis of the ensemble of *ACB*, are trained as in *Conserboost* (only the probability for incorrectly classified patterns is altered) and then this update of the sampling distribution tends to be softer as the number of networks increases.

Finally, the reinitialization of the distribution is also allowed as in *Conserboost*. It has been successfully applied in the methods proposed in [82] in which *Conserboost* was introduced. So, we think it can be useful in *ACB* too. The description of this new boosting method is shown in algorithm 8.1.

Algorithm 8.1 ACB $\{T, V, N_{networks}\}$

Initialize Sampling Distribution $Dist$:
$Dist_x^1 = 1/N_{patterns}\ \forall x \in T$
for $net = 1$ to $N_{networks}$ **do**
Create $T^{'}$ sampling from T using $Dist^{net}$
Network Training $\{T^{'}, V\}$
Calculate misclassified vector:
$$miss_x^{net} = \begin{cases} 1 & \text{if } h^{net}(x) \neq d(x) \quad x \text{ is incorrectly classified} \\ 0 & \text{otherwise} \quad x \text{ is correctly classified} \end{cases}$$
$h^{net}(x)$ is the class associated to the pattern x according to the output of net
$d(x)$ is the target class associated to the pattern x
Calculate error and other parameters:
$\epsilon^{net} = \sum_{x=1}^{m} Dist_x^{net} \cdot miss_x^{net}$
$\beta^{net} = \sqrt{\frac{1-\epsilon^{net}}{\epsilon^{net}}}, \epsilon^{net} \in (0, 0.5)$
if $\epsilon^{net} \notin (0, 0.5)$ **then**
Reinitialize Sampling Distribution $Dist$: $Dist_x^{net} = 1/N_{patterns}\ \forall x \in T$
end if
Update sampling distribution:
$C_x^{net} = Dist_x^{net} \cdot (\beta^{net})^{miss_x^{net}}$
$Dist_x^{net+1} = (net \cdot Dist_x^{net} + C_x^{net}) \cdot \frac{1}{net+1}$
Normalize($Dist^{net+1}$)
end for

8.3 Weighted-Conservative Boosting

Kuncheva proposed a relaxed version of *Adaboost* called *Conservative Boosting* [82]. According to the results shown in chapter 5, *Conserboost* can be considered one of the best ensemble alternatives. However, the results depend on the database and on the ensemble size and, moreover, there are some specific cases in which *Conserboost* performs worse than *Adaboost*.

Table 8.2 shows an extract of the complete results of *Adaboost* and *Conserboost* with representative results. Moreover, the difference of performance between them is also shown. With these results we show how the *difference* between *Conserboost* and *Adaboost* is not constant and in some cases is negative (i.e. database *Bupa* with 9 networks), which means that *Adaboost* performs better.

Weighted-Conservative Boosting, henceforth *WCB*, is a new boosting method in which *Conservative Boosting* is slightly modified in order to obtain a parameterized equation to update the sampling distribution. We think that *Conserboost* may be improved if a new parameter which should depend on the problem, is added in order to control the update of the sampling distribution. Our main purpose is to obtain a method whose performance will be better than the performance of *Adaboost* for any case in the general measurements.

Table 8.2: Comparison of the performance of *Adaboost* and *Conserboost*

database	networks	Conserboost	Adaboost	difference
band	3	71.82%	70.18%	1.64
	9	74%	72.91%	1.09%
	20	74.55%	73.45%	1.1%
	40	75.82%	71.09%	4.73%
bupa	3	72%	70.43%	1.57%
	9	71.29%	73.29%	−2%
	20	72.86%	73%	−0.14%
	40	73.43%	72.57%	0.86%

In *WCB*, the sampling distribution is updated by a modified version of the relaxed equation of *Conserboost* in which a problem-dependent parameter is added. The main idea of *WCB* is to apply a good equation whose strength could be set depending on the database and ensemble size, equation 8.7. This parameter, α, allows the algorithm to successfully update the sampling distribution for each case and it should be set by trial and error. The values of the new distribution are given by:

$$Dist_x^{net+1} = Dist_x^{net} \cdot \alpha \cdot \left(\beta^{net}\right)^{miss_x^{net}} \tag{8.7}$$

Finally, the proposed ensemble method is fully described in algorithm 8.2.

Algorithm 8.2 WCB $\{T, V, N_{networks}\}$

Initialize Sampling Distribution $Dist$: $Dist_x^1 = 1/N_{patterns}\ \forall x \in T$
for $net = 1$ to $N_{networks}$ **do**
  Create T' sampling from T using $Dist^{net}$
  Network Training $\{T'\ , V\}$
  Calculate misclassified vector:

$$miss_x^{net} = \begin{cases} 1 & \text{if } h^{net}(x) \neq d(x) \quad x \text{ is incorrectly classified} \\ 0 & \text{otherwise} \quad\quad\quad\quad\ x \text{ is correctly classified} \end{cases}$$

  $h^{net}(x)$ is the class associated to the pattern x according to the output of net
  $d(x)$ is the target class associated to the pattern x
  Calculate error and other parameters:
    $\epsilon^{net} = \sum_{x=1}^{m} Dist_x^{net} \cdot miss_x^{net}$
    $\beta^{net} = \sqrt{\frac{1-\epsilon^{net}}{\epsilon^{net}}}, \epsilon^{net} \in (0, 0.5)$
  if $\epsilon^{net} \notin (0, 0.5)$ **then**
    Reinitialize Sampling Distribution $Dist$: $Dist_x^{net} = 1/N_{patterns}\ \forall x \in T$
  end if
  Update sampling distribution:
    $Dist_x^{net+1} = Dist_x^{net} \cdot \alpha \cdot (\beta^{net})^{miss_x^{net}}$
    Normalize($Dist^{net+1}$)
end for

8.4 Cross-Validated Boosting

Cross-Validated Boosting is a completely new boosting methodology in which *Cross-Validation Committee* and *Boosting* approaches are mixed into a single methodology. According to the results published in chapter 5, some methods based on *Boosting* and *CVC* provided the best results.

However, both approaches are quite different. On the one hand, *Cross-Validation Committee* are methods in which the original learning set is split into subsets which are used to generate the specific training and validation sets. On the other hand, *Boosting* is a methodology that constructs a sequence of networks in which the networks are overfitted with hard to learn patterns.

Although the general results show that *CVC* approach is slightly better, in general, than *Boosting*, this does not mean that they classify exactly equal the same patterns because *Boosting* focuses its training on hard to learn patterns, to improve the performance of the ensemble in those patterns which are close to the class boundaries, and *CVC* focuses on generating slightly different versions of the training and validation sets, to increase the diversity of the ensemble.

In order to test that both approaches provide different solutions with similar performance we have compared the performance of 40-net ensembles previously trained with *CVCv3*, *Conserboost* and a special classifier *Any*, described below, in the following table. In this table, "*performance*" is the percentage of correctly classified patterns on the test set.

Table 8.3: Performance of *CVCv3* and *Conserboost*

Database	CVCv3	Conserboost	Any	Difference
aritm	76.0 ± 1.3	75.1 ± 1.0	81.5 ± 1	5.5%
bala	95.5 ± 0.6	96.2 ± 0.7	97.1 ± 0.4	0.9%
band	74.9 ± 1.1	75.8 ± 1.5	79.6 ± 1.3	3.8%
bupa	73.3 ± 1.2	73.4 ± 1.1	76.6 ± 1.2	3.2%
cred	86.9 ± 0.8	86.0 ± 0.7	89.6 ± 0.6	2.7%
derma	97.6 ± 0.5	97.6 ± 0.7	98.3 ± 0.5	0.7%
ecoli	86.2 ± 1.0	87.8 ± 1.1	89 ± 0.9	1.2%
flare	82.1 ± 0.6	82.4 ± 0.6	84.8 ± 0.4	2.4%
glas	94.6 ± 0.9	97.0 ± 0.7	97.6 ± 0.6	0.6%
hear	84.4 ± 1.4	83.9 ± 0.9	86.6 ± 1.2	2.2%
img	97.0 ± 0.3	97.2 ± 0.2	98 ± 0.2	0.8%
ionos	91.1 ± 0.9	91.9 ± 0.8	94.9 ± 0.6	3%
mok1	99.4 ± 0.6	100 ± 0	100 ± 0	0%
mok2	95.3 ± 1.3	86.6 ± 1.4	96.5 ± 0.9	1.2%
pima	76.6 ± 1.1	76.2 ± 1.2	79.5 ± 1.1	2.9%
survi	73.9 ± 1.2	73.3 ± 1.5	75.9 ± 1.4	2%
vote	96.1 ± 0.6	95.5 ± 0.8	97 ± 0.6	0.9%
vowel	93.2 ± 0.6	97.3 ± 0.6	98.4 ± 0.3	1.1%
wdbc	97.1 ± 0.3	96.3 ± 0.5	97.7 ± 0.5	0.6%

In the previous table, the performance of ensembles of 40 networks generated with *CVCv3* and *Conserboost* (two of the best ensemble alternatives according to the comparison performed in chapter 5) is shown. Moreover, we have also included the performance of a special classifier, *Any*, which denotes the percentage of patterns correctly classified by, at least, one of those two ensembles (*CVCv3* and *Conserboost*). Furthermore, we have also included the difference in performance of *Any* with respect to the maximal performance provided by *CVCv3* and *Conserboost*.

In this case, ensembles of 40 networks, the results provided by *Any* are better than the results provided by *CVCv3* and *Conserboost*. Although both ensemble methods have similar performance, they do not correctly classify the same patterns as can be obviously seen in database *aritm*. In this case, the *difference* of *Any* with respect to the other ensemble methods is higher than 5%.

As it has been mentioned before, both approaches are quite different. *CVC* focuses on using slightly different training and validation sets whereas *Boosting* focuses on specializing the networks on patterns close to the boundaries. These sources of diversity are also completely different so we consider that a new methodology based on both approaches should be seriously considered in order to build ensembles of neural networks efficiently. The methods based in the new methodology will inherit the advantages provided by both approaches.

This new methodology is detailed in algorithm 8.3.

Algorithm 8.3 Cross-Validated Boosting $\{T, V, N_{networks}\}$

Initialize Sampling Distribution:
  $Dist_x^1 = 1/N_{patterns}\ \forall x \in L$
Generate L from the union of T and V
Apply *Cross-Validation* to Learning set L to generate subsets
for $net = 1$ to $N_{networks}$ **do**
  Create T^{net} from subsets of L
  Create V^{net} from subsets of L
  Create T' sampling patterns from L which also are in T^{net} using $Dist^{net}$
  Network Training $\{T'\ ,\ V^{net}\}$
  Calculate misclassified vector $miss_{net}$
  Calculate error ϵ^{net}
  Calculate other parameters if required
  Reinitialize sampling distribution if required
  Update sampling distribution $\forall x \in L$
  Normalize Sampling Distribution:
    $\sum_{x=1}^{N_{patterns}} (Dist_x^{net+1}) = 1$
end for

In the methodology we propose, *Cross-Validated Boosting*, the generation of the specific training and validation sets is divided into two steps. In the first one, specifics sets are generated as in *Cross-Validation*, T^{net} and V^{net}. Then, the training set

used to train the network, T', is generated by sampling patterns, according to the sampling distribution of the particular boosting algorithm, from L which also are in T^{net}. It is important to mention that L refers to the original learning set which contains the patterns from the original training and validation sets. With this procedure, described in algorithm 8.3 the patterns which are not present in T^{net} are also not present in the training set used to train the network T' and simultaneously it is applied the boosting distribution to sample and overfit with hard to learn patterns. Moreover, each network has a validation set which is not shared with the other networks.

Figure 8.5 shows how the specific training set T' is generated for a network *net*. With this picture we want to give a visual representation of the two steps related to the procedure applied to generate the specific training sets, T'.

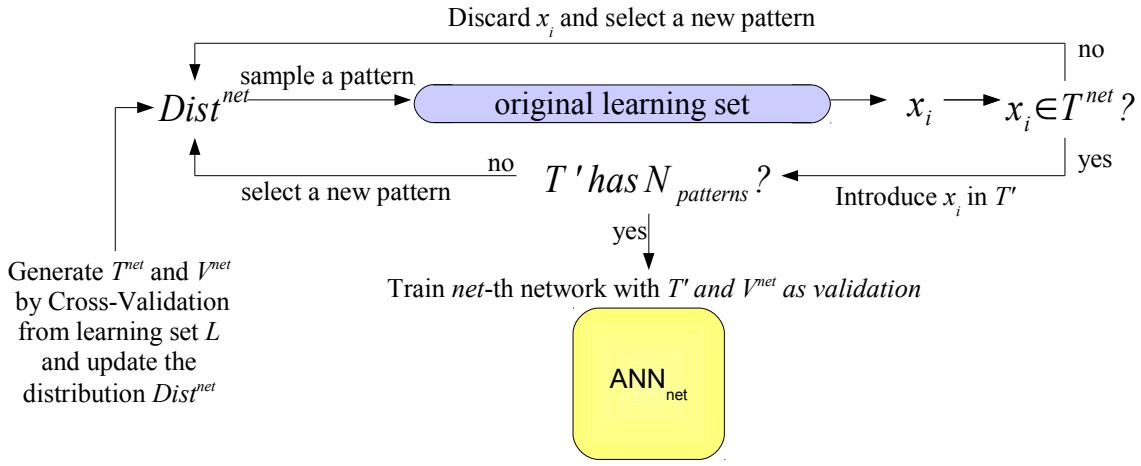


Figure 8.5: Generating T' in Cross-Validated Boosting

Cross-Validation Boosting is not just a method, its a methodology since there are several versions of *Cross-Validation Committee* and *Boosting* that can be combined among them. For instance, we can mix *CVCv2* and *Conserboost* in a single ensemble method and we can also mix *CVCv3* and *Inverboost*. However, *CVCv1* is not suggested to be employed because *CVCv1* does not provide specific validation sets and this source diversity is not used. We consider that this source of diversity is quite important and it should be present in the new methodology. Furthermore, *Boosting3* is limited to generate ensembles of 3 networks and in our experiments we also use ensembles of 9, 20 and 40 networks so it will not be used.

The ensembles derived from this new methodology will be named as the version of the *Cross-Validation* applied to generate the specific sets plus the *Boosting variant* employed to update the sampling distribution. For instance, the new *Cross-Validated Boosting* model based on *CVCv2* and *Adaboost* will be named *CVCv2Adaboost* whereas the new alternative based on *CVCv3* and *Inverboost* will be named *CVCv3Inverboost*.

8.5 Experimental Setup

In this chapter two new ensemble methods, *Averaged-Conservative Boosting* and *Weighted-Conservative Boosting*, are proposed. Moreover, a new methodology called *Cross-Validated Boosting* is also described.

As in the previous ensemble comparisons, a similar experimental setup is applied to test the performance of the new methods. Its main characteristics are:

- Nineteen datasets from the *UCI Repository*.
- Two new ensemble alternatives:
  - *Averaged-Conservative Boosting*.
  - *Weighted-Conservative Boosting*.
- Eighteen new ensembles based on Cross-Validated Boosting combining:
  - Two versions of *Cross-Validation*, *CVCv2* and *CVCv3*.
  - Nine *Boosting* approaches.
- One network architecture: *MF* network.
- Four different ensemble sizes: 3, 9, 20 and 40 networks.
- Optimized training parameters.
- Two combiners applied:
  - *Output Combiners*.
  - *Specific Boosting Combiner*.
- Experiments repeated ten times with different partitions of the training, validation and test sets to obtain:
  - Mean value of performance.
  - Error rate by standard error theory.
- Three general measurements applied to the comparison:
  - Mean *Increase of Performance*.
  - Mean *Percentage of Error Reduction*.
  - Student's Paired t-test.

The description of the nineteen datasets used in the experiments can be found in appendix A. The optimized training parameters of the networks are in appendix B.3 whereas the specific parameters of the ensembles are in appendix B.5. Futhermore, the general measurements applied to compare the ensembles studied are the mean *IoP* (equation 3.5) and the *PER* (equation 3.6). Finally, the *Student's Paired t-test*, described in section 3.6.3.2, is also used.

8.6 Results and discussion

The results of the ensembles proposed in this chapter are shown in this section. Due to their size, the section has been split into three subsections. The first one will show the discussion related to the results provided by *ACB* and *WCB* whereas the second one will be focused on the *Cross-Validated Boosting* methodology with classic boosting variants already known in the bibliography. Finally, we show the results provided with *Cross-Validated Boosting* where the new proposed boosting alternatives *ACB* and *WCB* are used as the boosting procedure.

8.6.1 Ensembles generated with *ACB* and *WCB*

In the first experiments performed in this chapter, *Averaged-Conservative Boosting*, *ACB*, and *Weighted-Conservative Boosting*, *WCB*, have been used to generate ensembles of *MF* networks.

Table 8.4 shows the mean IoP and the mean PER of the new ensembles applying *Output average*, *ave* in the table, and the specific *Boosting Combiner*, *bst* in the table, to fuse the output of the networks. Moreover, the results of *Simple Ensemble* and some boosting alternatives are included to compare them with the new methods.

Table 8.4: Results of the new ensemble methods *ACB*

	Mean IoP				Mean PER			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Simple Ensemble	4.97	5.27	5.34	5.43	21.84	23.65	23.73	24.64
Adaboost	3.69	4.55	4.93	4.98	15.4	19.5	22.96	24.54
Aveboost	4.33	5.57	5.95	6.04	18.26	26.11	27.12	26.53
Conserboost	4.42	5.31	5.65	6.06	19.72	25.63	26.62	27.84
ACB - ave	4.59	5.53	5.89	6.04	21.39	27.37	28.18	28.46
ACB - bst	4.32	5.16	5.75	6.08	17.61	24.69	26.89	28.3
WCB - ave	4.93	5.43	6.01	6.13	22.26	25.97	29.13	29.87
WCB - bst	4.68	5.49	5.89	6.18	21.69	26.11	27.97	29.54

Firstly, we can see that the results of the new ensembles depend on the combiner applied. In general, *Output average* provides the best results. However, the specific *Boosting Combiner* also provides similar results for the case of ensembles of 40 networks. For the cases of ensembles from 3 to 20 networks we should apply *Output average* whereas both combiners can be applied to ensembles of 40 networks.

Secondly, the proposed methods, *ACB* and *WCB*, improve the results provided by *Aveboost* and *Conserboost*, in general. In the case of *ACB*, it provides better results than *Aveboost* and *Conserboost* for the ensemble of 3 networks. In the first networks the sampling distribution is updated as in *Conserboost* so good results are obtained for low sized ensembles. Then the updating equation becomes softer and softer and the accuracy of the ensemble increases as new networks are added to the network as in *Aveboost*. As we can see, according to the mean PER, both classic boosting methods have been successfully mixed.

Furthermore, *WCB* also provides better results than *Conserboost*. In this case, a parameter which depends on the problem has been included in the equation to update the sampling distribution. With this parameter we can control the strength of the *update* so we can *optimize* this update for each problem. Moreover, we can also avoid those cases in which the *Adaboost* performed better than *Conserboost*. *Adaboost* is better than *Conserboost* in 17 of 76 cases (22%) but *Adaboost* is better than *WCB* in only 13 cases (17%).

Moreover, the ensembles of 20 networks generated by *WCB*, where the mean IoP and PER are 6.01% and 29.13% respectively, performs slightly better than the ensembles generated in chapters 4 and 5, where the mean IoP and PER are around 6.1% and 28.7%. Furthermore, the best overall results are provided by ensembles of 40 networks trained with *WCB* and combined with *Output average*, where the mean IoP and PER are close to 6.2% and 30% respectively.

To provide statistical information to the results, the *t-test* is applied to compare the ensembles as has been previously done. In this case, the proposed methods are combined by *Output average* and *Boosting Combiner*, whereas the original boosting methods are combined by *Boosting Combiner* as in their references.

ACB and *WCB* are statistically compared to their original boosting methods in table 8.5. *ACB* has been compared to *Adaboost*, *Aveboost* and *Conserboost* whereas *WCB* has been compared to *Adaboost* and *Conserboost*. With this procedure, it can be tested if the proposed methods perform better than the original versions.

Table 8.5: *ACB* and *WCB* compared to their original boosting methods

Methods	measure	3-net	9-net	20-net	40-net
ACB-ave vs Adaboost	t-**value**	2.77	3.17	3.13	3.50
	α	0.0061	0.0018	0.002	0.00057
ACB-bst vs Adaboost	t-**value**	2.21	2.22	2.70	3.58
	α	0.028	0.028	0.0076	0.00043
ACB-ave vs Conserboost	t-**value**	0.66	1.27	1.51	−0.13
	α	0.51	0.21	0.13	0.90
ACB-bst vs Conserboost	t-**value**	−0.38	−0.76	0.55	0.15
	α	0.70	0.45	0.58	0.88
ACB-ave vs Aveboost	t-**value**	1.01	−0.21	−0.34	−0.03
	α	0.31	0.83	0.73	0.98
ACB-bst vs Aveboost	t-**value**	−0.01	−2.13	−1.11	0.23
	α	0.99	0.035	0.27	0.82
WCB-ave vs Adaboost	t-**value**	3.49	2.79	3.22	3.63
	α	0.00061	0.0058	0.0015	0.00036
WCB-bst vs Adaboost	t-**value**	2.89	2.88	3.00	3.70
	α	0.0042	0.0045	0.0031	0.00028
WCB-ave vs Conserboost	t-**value**	1.89	0.53	1.67	0.40
	α	0.06	0.60	0.10	0.69
WCB-bst vs Conserboost	t-**value**	0.93	0.84	1.11	0.71
	α	0.36	0.40	0.27	0.48

First of all, we can see in the previous table that, in general, the ensembles combined by *Output average* provide higher t-values and lower α values than the same ensembles combined with *Boosting Combiner*. For the case of ensembles of 40 networks, both combiners provide similar general results but the ensembles combined with *Boosting Combiner* tend to perform slightly better according to the statistical values. We think that *Output average* may be used when the new ensembles, *ACB* and *WCB*, are combined according to the statistical test and general measurements. *Boosting Combiner* can also be used to combine ensembles of 40 networks.

Secondly, the improvements of *ACB* and *WCB* with respect to *Adaboost* are statistically significant because α is quite lower than 0.05. Moreover, both new methods slightly improve *Conserboost* but the results are not statistically significant because α is greater than 0.05.

Thirdly, *ACB* provides better PER than *Aveboost* for any ensemble size but it also provides worse IoP for the cases of ensembles of 9 and 20 networks. Considering the best combiner, the statistical tests show that *Aveboost* is slightly better than *ACB* for the cases of 9 and 20 networks in the ensemble but the differences between them are not statistically significant because α is quite greater than 0.05. We consider that both methods provide excellent results but *Aveboost* fits better in some datasets and *ACB* fits better on other datasets because their sources of diversity are different.

To conclude this part of the experiments, the new methods *ACB* and *WCB* provide slightly better results than the classic boosting methods according to the PER and the statistical tests. Improving an ensemble method requires some steps and the new methods may be considered as an important part, or step, in the procedure of designing better ensemble methods.

8.6.2 Results of *Cross-Validated Boosting*

In this subsection, we present the results of the new ensembles based on *Cross-Validated Boosting* where the original boosting methods are used. These classic variants are: *Adaboost*, two versions of *Aveboost*, *Aggreboost*, *Conserboost*, *Inverboost* and *ARCx4*.

Table 8.6 shows the mean IoP and the mean PER of the classic ensembles and the new proposals based on *Cross-Validated Boosting*. In these results, *Output average*, denoted by "-a" at the end of the name of the method, and *Boosting Combiner*, denoted by "-b", are used to fuse the outputs of the networks generated by the new ensembles, whereas *Boosting Combiner* is only used in the original boosting variants.

Firstly, the results provided by the new ensembles improve in general the general performance provided by the original boosting alternatives they were based on. In this way, the successive improvements of *Adaboost* have been outperformed. But, in the case of *Adaboost* and *Inverboost* the cross-validated variants are only better for the case of 3 and 9 networks. For *Inverboost*, the new ensembles also improves it for the case of 20 networks.

Table 8.6: Results of the new *Cross-Validated Boosting* methods

	Mean *IoP*				Mean *PER*			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Adaboost	3.69	4.55	4.93	4.98	15.4	19.5	22.96	24.54
CVCv2Adaboost-a	4.18	4.97	4.66	2.91	18.28	22.61	20.92	9.79
CVCv2Adaboost-b	3.09	4.96	4.75	4.58	12.21	23.31	21.58	20.79
CVCv3Adaboost-a	4.18	4.79	4.49	4.42	16.64	21.55	19.79	20.34
CVCv3Adaboost-b	3.53	4.61	4.66	4.89	11.71	20.39	21.52	22.52
Aveboost	4.33	5.57	5.95	6.04	18.26	26.11	27.12	26.53
CVCv2Aveboost-a	4.39	5.85	6.14	5.83	19.35	27.3	29.2	26.45
CVCv2Aveboost-b	3.81	5.83	6.25	5.8	15.56	27.57	29.96	26.95
CVCv3Aveboost-a	4.93	5.88	6.19	6.24	21.28	27.58	29.84	29.95
CVCv3Aveboost-b	4.26	5.61	6.02	6.15	16.32	26.44	28.47	28.77
Aveboost2	4.02	4.59	4.88	5	17.67	22.34	22.99	23.32
CVCv2Aveboost2-a	4.35	5.23	5.34	5.39	19.44	24.42	26.24	26.42
CVCv2Aveboost2-b	3.41	4.99	5.34	5.32	14.81	23.14	24.55	25.71
CVCv3Aveboost2-a	4.46	5.16	5.35	5.49	20.32	24.98	26.11	26.16
CVCv3Aveboost2-b	4.03	4.82	5.03	5.22	17.4	22.62	24.21	24.27
Aggreboost	3.56	4.79	5.53	5.83	14.82	20.26	25.27	26.56
CVCv2Aggreboost-a	3.96	5.12	5.69	5.66	16.07	23.85	27.43	27.16
CVCv2Aggreboost-b	3.35	5.11	5.77	5.83	11.78	23.42	27.54	27.43
CVCv3Aggreboost-a	3.97	4.96	5.63	5.9	14.46	21.56	27.13	28.06
CVCv3Aggreboost-b	3.41	4.92	5.61	6.19	12.27	23.18	27.25	29.25
Conserboost	4.42	5.31	5.65	6.06	19.72	25.63	26.62	27.84
CVCv2Conserboost-a	4.67	5.89	6.06	5.79	22.51	27.51	29.03	28.06
CVCv2Conserboost-b	3.79	5.75	6.16	5.93	16.88	26.45	29.52	27.83
CVCv3Conserboost-a	4.88	5.91	6.36	6.49	22.37	28.38	30.02	30.67
CVCv3Conserboost-b	4.09	5.7	6.29	6.5	17.17	27.45	30.18	31.29
Inverboost	2.89	0.95	−1.23	−2.76	9.2	−3.12	−15	−24.18
CVCv2Inverboost-a	3.04	2.15	−0.62	−4.49	9.76	2.7	−12.89	−39.55
CVCv2Inverboost-b	2.74	1.85	−1.44	−5.91	8.52	0.83	−19.66	−52.92
CVCv3Inverboost-a	3.69	1.93	−0.52	−2.71	13.89	5.42	−7.47	−20.86
CVCv3Inverboost-b	3.29	1.64	−1.39	−3.65	12.38	3.64	−13.81	−30.37
ARCx4	4.06	3.78	4.8	5.24	18.01	17.69	22.56	24.37
CVCv2ARCx4-a	3.36	3.73	4.89	5.33	16.17	19.43	24.51	24.54
CVCv2ARCx4-b	2.99	4.39	5.35	6.07	12.23	19.96	25.66	29.07
CVCv3ARCx4-a	3.93	3.91	5.2	5.73	18.33	19.52	26.56	28.32
CVCv3ARCx4-b	3.14	4.2	5.42	6.05	13.62	19.6	26.78	28.84

Secondly, the new proposed methods based on *CVCv3* provide, in general, better results than the ones proposed and based on *CVCv2*. However, there are a several cases for ensembles of 9 networks in which the ensembles based on *CVC2* are better.

Thirdly, *Output average* has a good performance for any ensemble, in general, whereas *Boosting Combiner* fits better on high sized ensembles. Furthermore, *Output average* is the best combiner for the new alternatives based on *Aveboost v2*.

The analysis of the general results is a complex task. There are seven boosting methods which have been mixed with two version of *CVC* and two ensemble combiners have been used. For these reasons, the two following *resume* tables are introduced which denote which is the best *CVC* method and combiner for each case.

Table 8.7: Best combiner of the new *Cross-Validated Boosting* methods

	Mean *IoP*				Mean *PER*			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
CVCv2Adaboost	*ave*	*ave*	*bst*	*bst*	*ave*	*bst*	*bst*	*bst*
CVCv3Adaboost	*ave*	*ave*	*bst*	*bst*	*ave*	*ave*	*bst*	*bst*
CVCv2Aveboost	*ave*	*ave*	*bst*	*ave*	*ave*	*bst*	*bst*	*bst*
CVCv3Aveboost	*ave*	*ave*	*ave*	*ave*	*ave*	*ave*	*ave*	*ave*
CVCv2Aveboost2	*ave*	*ave*	*bst*	*ave*	*ave*	*ave*	*ave*	*ave*
CVCv3Aveboost2	*ave*	*ave*	*ave*	*ave*	*ave*	*ave*	*ave*	*ave*
CVCv2Aggreboost	*ave*	*ave*	*bst*	*bst*	*ave*	*ave*	*bst*	*bst*
CVCv3Aggreboost	*ave*	*ave*	*ave*	*bst*	*ave*	*bst*	*bst*	*bst*
CVCv2Conserboost	*ave*	*ave*	*bst*	*bst*	*ave*	*ave*	*bst*	*ave*
CVCv3Conserboost	*ave*	*ave*	*ave*	*bst*	*ave*	*ave*	*bst*	*bst*
CVCv2Inverboost	*ave*	*ave*	*bst*	*bst*	*ave*	*ave*	*bst*	*bst*
CVCv3Inverboost	*ave*	*ave*	*bst*	*bst*	*ave*	*ave*	*bst*	*bst*
CVCv2ARCx4	*ave*	*bst*	*bst*	*bst*	*ave*	*bst*	*bst*	*bst*
CVCv3ARCx4	*ave*	*bst*	*bst*	*bst*	*ave*	*bst*	*bst*	*bst*

According to table 8.7, the best combiner is *Output average* for ensembles of 3 and 9 networks, whereas *Boosting Combiner* fits better on ensembles of 20 and 40 networks. As table 8.8 shows, *CVCv3* fits better on ensembles of 3 and 40 network and *CVCv2* fits slightly better on ensembles of 9 and 20 networks.

Table 8.8: Best *CVC* method of the new *Cross-Validated Boosting* methods

	Mean *IoP*				Mean *PER*			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Adaboost-ave	*v3*	*v2*	*v2*	*v3*	*v2*	*v2*	*v2*	*v3*
Adaboost-bst	*v3*	*v2*	*v2*	*v3*	*v2*	*v2*	*v2*	*v3*
Aveboost-ave	*v3*	*v3*	*v3*	*v3*	*v3*	*v3*	*v3*	*v3*
Aveboost-bst	*v3*	*v2*	*v2*	*v3*	*v3*	*v2*	*v2*	*v3*
Aveboost2-ave	*v3*	*v2*	*v3*	*v3*	*v3*	*v3*	*v2*	*v2*
Aveboost2-bst	*v3*	*v2*	*v2*	*v2*	*v3*	*v2*	*v2*	*v2*
Aggreboost-ave	*v3*	*v2*	*v2*	*v3*	*v2*	*v2*	*v2*	*v3*
Aggreboost-bst	*v3*	*v2*	*v2*	*v3*	*v3*	*v2*	*v2*	*v3*
Conserboost-ave	*v3*	*v3*	*v3*	*v3*	*v2*	*v3*	*v3*	*v3*
Conserboost-bst	*v3*	*v2*	*v3*	*v3*	*v3*	*v3*	*v3*	*v3*
Inverboost-ave	*v3*	*v2*	*v2*	*v2*	*v3*	*v3*	*v2*	*v2*
Inverboost-bst	*v3*	*v2*	*v2*	*v2*	*v3*	*v3*	*v2*	*v2*
ARCx4-ave	*v3*	*v3*	*v3*	*v3*	*v3*	*v3*	*v3*	*v3*
ARCx4-bst	*v3*	*v2*	*v3*	*v2*	*v3*	*v2*	*v3*	*v2*

The *t-test* is also applied to all the ensembles in order to statistically compare them. Firstly, the new ensembles based on *Cross-Validated Boosting* are compared to *Adaptive Boosting* (tables 8.9 and 8.10). Then, each new alternative based on *Cross-Validated Boosting* is compared to its corresponding original boosting version (tables 8.11 and 8.12).

Table 8.9 shows the statistical results when *Adaboost* is compared to the new *Cross-Validated Boosting* alternatives based on *CVCv2*. *Boosting Combiner* is used to combine the ensembles in *Adaboost*, whereas *Output average* (*ave* in the table) and *Boosting Combiner* (*bst* in the table) are used to combine the new ensembles.

Table 8.9: *Cross-Validated Boosting* vs. *Adaboost* (I)

Methods	measure	3-net	9-net	20-net	40-net
Adaboost vs CVCv2Adaboost-ave	*t*-value	−1.81	−1.62	−0.21	1.06
	α	0.071	0.11	0.84	0.29
Adaboost vs CVCv2Adaboost-bst	*t*-value	0.48	−1.66	−0.39	−1.25
	α	0.64	0.1	0.69	0.21
Adaboost vs CVCv2Aveboost-ave	*t*-value	−2.25	−3.62	−3.51	−3.13
	α	0.025	0.0004	0.0006	0.002
Adaboost vs CVCv2Aveboost-bst	*t*-value	−1.08	−3.65	−3.85	−3.1
	α	0.28	0.0003	0.0002	0.0023
Adaboost vs CVCv2Aveboost2-ave	*t*-value	−2.07	−2.32	−1.64	−2.41
	α	0.04	0.021	0.1	0.017
Adaboost vs CVCv2Aveboost2-bst	*t*-value	−0.19	−1.78	−1.62	−2.3
	α	0.85	0.077	0.11	0.023
Adaboost vs CVCv2Aggreboost-ave	*t*-value	−1.40	−2.09	−2.36	−2.82
	α	0.16	0.038	0.019	0.0052
Adaboost vs CVCv2Aggreboost-bst	*t*-value	−0.06	−1.98	−2.6	−3.15
	α	0.95	0.05	0.01	0.0019
Adaboost vs CVCv2Conserboost-ave	*t*-value	−3.01	−3.82	−3.26	−3.02
	α	0.0029	0.0002	0.0013	0.0029
Adaboost vs CVCv2Conserboost-bst	*t*-value	−1.02	−3.53	−3.51	−3.3
	α	0.31	0.0005	0.0006	0.0012
Adaboost vs CVCv2Inverboost-ave	*t*-value	0.56	3.81	7.67	7.59
	α	0.58	0.0002	≈ 0	≈ 0
Adaboost vs CVCv2Inverboost-bst	*t*-value	1.16	4.2	8.25	8.3
	α	0.25	≈ 0	≈ 0	≈ 0
Adaboost vs CVCv2ARCx4-ave	*t*-value	−0.08	0.81	−0.64	−2.29
	α	0.94	0.42	0.52	0.023
Adaboost vs CVCv2ARCx4-bst	*t*-value	1.15	0.08	3.27	−1.49
	α	0.25	0.94	0.0013	0.14

According to the previous table, the new cross-validated ensembles based on *Aveboost*, *Aveboost v2*, *Aggreboost* and *Conserboost* highly improve the original *Adaboost* because the *t*-value is negative and high. Moreover, this improvement is statistically significant because the value of α is quite lower than 0.05, except for the case of three networks in the ensemble. The differences of the new methods based on *ARCx4* and

Adaboost with respect to the original *Adaboost* are not, in general, statistically significant. Furthermore, *Adaboost* is better than the new methods based on *Inverboost* and their differences are statistically significant, but this is also the case with the original or classic *Inverboost*.

Table 8.10 shows the statistical results when *Adaboost* is compared to the new *Cross-Validated Boosting* models based on *CVCv3* and combiners *Output average* "-ave" and *Boosting Combiner* "-bst". As in the previous table, the original boosting variants are combined with *Boosting Combiner*.

Table 8.10: *Cross-Validated Boosting* vs. *Adaboost* (II)

Methods	measure	3-net	9-net	20-net	40-net
Adaboost vs CVCv3Adaboost-ave	t-value	-1.78	-1.32	0.17	-0.97
	α	0.08	0.19	0.86	0.33
Adaboost vs CVCv3Adaboost-bst	t-value	-0.42	-0.91	-0.22	-1.74
	α	0.67	0.36	0.83	0.08
Adaboost vs CVCv3Aveboost-ave	t-value	-3.53	-3.81	-3.65	-3.69
	α	0.0005	0.0002	0.0003	0.0003
Adaboost vs CVCv3Aveboost-bst	t-value	-1.89	-3.18	-3.27	-3.57
	α	0.06	0.0017	0.0013	0.0005
Adaboost vs CVCv3Aveboost2-ave	t-value	-2.43	-2.10	-1.61	-2.52
	α	0.016	0.037	0.11	0.013
Adaboost vs CVCv3Aveboost2-bst	t-value	-1.5	-1.33	-0.98	-2.12
	α	0.14	0.19	0.33	0.035
Adaboost vs CVCv3Aggreboost-ave	t-value	-1.42	-1.72	-2.42	-3.22
	α	0.16	0.087	0.016	0.002
Adaboost vs CVCv3Aggreboost-bst	t-value	-0.2	-1.61	-2.46	-3.66
	α	0.84	0.11	0.015	0.00033
Adaboost vs CVCv3Conserboost-ave	t-value	-3.38	-3.76	-3.94	-4.11
	α	0.0009	0.0002	0.0001	≈ 0
Adaboost vs CVCv3Conserboost-bst	t-value	-1.69	-3.36	-3.69	-4.14
	α	0.093	0.001	0.0003	≈ 0
Adaboost vs CVCv3Inverboost-ave	t-value	-0.73	3.86	7.35	6.42
	α	0.47	0.0002	≈ 0	≈ 0
Adaboost vs CVCv3Inverboost-bst	t-value	0.05	4.18	8.05	7.25
	α	0.96	≈ 0	≈ 0	≈ 0
Adaboost vs CVCv3ARCx4-ave	t-value	-1.21	0.54	-1.29	-2.90
	α	0.23	0.59	0.20	0.0041
Adaboost vs CVCv3ARCx4-bst	t-value	0.66	0.57	3.71	-1.43
	α	0.51	0.57	0.0003	0.16

As table 8.10 shows, the new ensembles based on the mixture of *CVCv3* and *Aveboost*, *Aveboost v2*, *Aggreboost* and *Conserboost* also highly improve the original *Adaboost*. Furthermore, t-values shown in the table denote higher difference than the t-values shown in table 8.9, specially for the case of ensembles of 3 and 40 networks. Moreover, it also occurs the same with the α values which tend to be lower in this table. This means that the results of the new alternatives based on *CVCv3* tend to be better than the results of the new ensembles based on *CVCv2*.

For instance, *CVCv3Conserboost* provides better results than *CVCv2Conserboost* for ensembles of 40 networks. The t-value is -3.02 and α is 0.0029 in *CVCv2Conserboost* with *Output average* and the t-value is -4.11 and α is almost 0% in *CVCv3Conserboost* with *Output average*. In the case of *CVCv3Conserboost* the t-value is higher, which means that it has better performance and lower dispersion ratio than *CVCv2Conserboost* when both are compared to *Adaboost*. Moreover, the value of α is quite lower, almost 0%, in *CVCv3Conserboost* which denotes that the difference of the results is higher for *CVCv3Conserboost* than for *CVCv2Conserboost* when both are compared to *Adaboost*.

Furthermore, table 8.11 shows the statistical results when each original boosting variant is compared to its new *Cross-Validated Boosting* version based on *CVCv2*. As in the previous tables (8.9 and 8.10), *Boosting Combiner* is used to fuse the networks in the original boosting variants.

Table 8.11: *Cross-Validated Boosting* vs. Original methods (I)

Methods	measure	3-net	9-net	20-net	40-net
Adaboost vs CVCv2Adaboost-ave	t-value	-1.81	-1.62	-0.21	1.06
	α	0.071	0.11	0.84	0.29
Adaboost vs CVCv2Adaboost-bst	t-value	0.48	-1.66	-0.39	-1.25
	α	0.64	0.1	0.69	0.21
Aveboost vs CVCv2Aveboost-ave	t-value	-0.04	-1.21	-0.99	1.23
	α	0.97	0.23	0.33	0.22
Aveboost vs CVCv2Aveboost-bst	t-value	2.00	-1.17	-1.63	1.38
	α	0.047	0.25	0.1	0.17
Aveboost2 vs CVCv2Aveboost2-ave	t-value	-1.13	-3.15	-2.85	-2.47
	α	0.26	0.0019	0.0049	0.014
Aveboost2 vs CVCv2Aveboost2-bst	t-value	2.37	-1.83	-2.36	-2.13
	α	0.019	0.069	0.019	0.034
Aggreboost vs CVCv2Aggreboost-ave	t-value	-1.42	-1.20	-0.58	0.81
	α	0.16	0.23	0.56	0.42
Aggreboost vs CVCv2Aggreboost-bst	t-value	0.53	-1.09	-0.97	0.00
	α	0.60	0.28	0.33	1.00
Conserboost vs CVCv2Conserboost-ave	t-value	-0.92	-2.69	-1.77	1.28
	α	0.36	0.0078	0.08	0.20
Conserboost vs CVCv2Conserboost-bst	t-value	2.26	-2.04	-2.27	0.67
	α	0.025	0.043	0.024	0.50
Inverboost vs CVCv2Inverboost-ave	t-value	-0.53	-3.64	-1.79	3.72
	α	0.60	0.0004	0.076	0.0003
Inverboost vs CVCv2Inverboost-bst	t-value	0.55	-2.83	0.58	5.25
	α	0.58	0.005	0.56	≈ 0
ARCx4 vs CVCv2ARCx4-ave	t-value	1.00	1.66	-6.58	-3.56
	α	0.32	0.1	≈ 0	0.0005
ARCx4 vs CVCv2ARCx4-bst	t-value	3.16	1.13	-0.41	-1.66
	α	0.0018	0.26	0.68	0.1

According to table 8.11, *Aveboost2* is improved by its new cross-validated alternatives based on *CVCv2* and the differences are statistically significant. *Conserboost*

with both combiners and *Inverboost* with *Output average* are improved for the cases of ensembles of 9 and 20 networks and the differences are also statistically significant. Furthermore, *CVCv2ARCx4* with *Output average* improves the original *ARCx4* and the differences are significant only for the cases of ensembles of 20 and 40 networks.

As commented before, the alternative of *CVCv2* is best suited for ensembles of 9 and 20 networks in comparison with *CVCv3*. In this sense, for the case of 9 and 20 networks in table 8.11, 25 of a total of 28 t-values are negative which means that the proposed cross-validated ensembles outperform the original ones for the case of medium size ensembles (9 and 20 networks in the ensemble). In ensembles of 3 and 40 networks, the original variant is better than a new alternative in 17 of 28 cases.

Finally, table 8.12 shows the statistical results when each original boosting method is compared to its new *Cross-Validated Boosting* version based on *CVCv3*. As in table 8.11, the original boosting ensembles are fused by *Boosting Combiner* and the two combiners are included for cross-validated boosting.

Table 8.12: *Cross-Validated Boosting* vs. Original methods (II)

Methods	measure	3-net	9-net	20-net	40-net
Adaboost vs CVCv3Adaboost-ave	t-value	−1.78	−1.32	0.17	−0.97
	α	0.076	0.19	0.86	0.33
Adaboost vs CVCv3Adaboost-bst	t-value	−0.42	−0.91	−0.22	−1.74
	α	0.67	0.36	0.83	0.084
Aveboost vs CVCv3Aveboost-ave	t-value	−2.31	−1.54	−1.27	−1.15
	α	0.022	0.13	0.21	0.25
Aveboost vs CVCv3Aveboost-bst	t-value	0.24	−0.18	−0.36	−0.62
	α	0.81	0.85	0.72	0.53
Aveboost2 vs CVCv3Aveboost2-ave	t-value	−1.58	−2.64	−2.39	−2.97
	α	0.12	0.0089	0.018	0.0034
Aveboost2 vs CVCv3Aveboost2-bst	t-value	−0.06	−1.04	−0.82	−1.33
	α	0.95	0.30	0.41	0.19
Aggreboost vs CVCv3Aggreboost-ave	t-value	−1.49	−0.60	−0.37	−0.30
	α	0.14	0.55	0.71	0.76
Aggreboost vs CVCv3Aggreboost-bst	t-value	0.29	−0.46	−0.35	−2.00
	α	0.77	0.65	0.73	0.05
Conserboost vs CVCv3Conserboost-ave	t-value	−1.82	−2.40	−2.90	−2.14
	α	0.071	0.017	0.0042	0.034
Conserboost vs CVCv3Conserboost-bst	t-value	1.29	−1.64	−2.73	−2.46
	α	0.20	0.1	0.0068	0.015
Inverboost vs CVCv3Inverboost-ave	t-value	−2.63	−3.19	−2.25	−0.12
	α	0.0092	0.0017	0.025	0.90
Inverboost vs CVCv3Inverboost-bst	t-value	−1.39	−2.22	0.46	2.47
	α	0.17	0.028	0.64	0.014
ARCx4 vs CVCv3ARCx4-ave	t-value	−0.68	1.66	−7.31	−5.89
	α	0.50	0.1	≈ 0	≈ 0
ARCx4 vs CVCv3ARCx4-bst	t-value	2.44	2.00	0.00	−1.63
	α	0.016	0.046	1.00	0.1

As shown in table 8.12, the proposed ensembles based on *CVCv3* improved the original variants and provided statistically significant differences in nearly 50% of the cases. Moreover, *Aveboost2*, *Conserboost* and surprinsingly *Inverboost* are statistically improved by, at least, one of the two combiners employed.

Comparing the t-values for the case of *Output average* as combiner, 26 of a total of 28 values are negative which means that the cross-validated version seems to perform better. Furthermore, in 12 of the 28 cases the result is even statistically significant.

As a resume, table 8.13 shows the statistical comparison among each original boosting variant and its cross-validated version. This table is a resume of the two previous ones and it can be used to perform an easier comparison. In the table, a negative *t-value* means that the proposed method is statistically better than its original boosting alternative whereas a positive value means that it is statistically worse. Analogously, − and + denote the tendency of the *t-values* when the differences are not statistically significant. Finally, symbol '=' denotes that the proposed method and its original have almost the same performance.

Table 8.13: Resume of the Statistical comparison

	Output Average				**Boosting Combiner**			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
CVCv2Adaboost	−	−	−	+	+	−	−	−
CVCv2Aveboost	−	−	−	+	+2	−	−	+
CVCv2Aveboost2	−	−3.15	−2.85	−2.47	+2.37	−	−2.36	−2.13
CVCv2Aggreboost	−	−	−	+	+	−	−	=
CVCv2Conserboost	−	−2.69	−	+	+2.26	−2.04	−2.27	+
CVCv2Inverboost	−	−3.64	−	+3.72	+	−2.83	+	+5.25
CVCv2ARCx4	+	+	−6.58	−3.56	+3.16	+	−	−
CVCv3Adaboost	−	−	+	−	−	−	−	−
CVCv3Aveboost	−2.31	−	−	−	+	−	−	−
CVCv3Aveboost2	−	−2.64	−2.39	−2.97	−	−	−	−
CVCv3Aggreboost	−	−	−	−	+	−	−	−2
CVCv3Conserboost	−	−2.4	−2.9	−2.14	+	−	−2.73	−2.46
CVCv3Inverboost	−2.63	−3.19	−2.25	−	−	−2.22	+	+2.47
CVCv3ARCx4	−	+	−7.31	−5.89	+2.44	+2	=	−

As the previous *resume* table shows, the best alternative to generate the new boosting methods is based on *CVCv3* and *Output average*. These new variants provide statistically better results in nearly 50% of the cases and in the other cases the tendency is also to be better except in two cases. In the case of the new ensembles based on *CVCv3* and *Boosting Combiner* we can see that there are four cases in which they are statistically better than their original. However, there are some cases in which the original method is statistically better than the new proposed one. In the other new alternatives based on *CVCv2*, the behavior is similar because there are only a few statistically improvements and there are some cases in which the new proposed models are statistically worse than their original boosting variant.

Taking into account all the results shown in this subsection, we consider that the new cross-validated boosting ensembles improve the original boosting alternatives. In order to obtain the best ensemble, the *Cross-Validated Boosting* models should be based, in general, on *CVCv3* and combined with *Output average*. In the procedure of designing better ensembles, the new proposed methodology may be also considered as an important step.

8.6.3 *ACB* and *WCB* with *Cross-Validated Boosting*

In the third experiments performed in this chapter, *ACB* and *WCB* have been used to generate ensembles of *MF* networks with the *Cross-Validated Boosting* methodology. The new ensembles are *CVCv2ACB*, *CVCv3ACB*, *CVCv2WCB* and *CVCv3WCB*. With this procedure, we mix three important and different alternatives, proposed by us, which have been successfully applied to design new ensembles as shown in the previous experiments of this chapter.

In this case, the results of the four new ensembles are shown in table 8.14. In this table, it is included their mean *IoP* and *PER*. As in the previous experiments, *Output average*, *ave*, and the specific *Boosting Combiner*, *bst*, are used to fuse the outputs of the networks. Moreover, in table 8.15 we show which combiner fits better for each new ensemble and size.

Table 8.14: General performance of *ACB* and *WCB* with *Cross-Validated Boosting*

	Mean *IoP*				**Mean *PER***			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
ACB-ave	4.59	5.53	5.89	6.04	21.39	27.37	28.18	28.46
CVCv2ACB-ave	4.21	5.97	6.15	5.97	20.42	29.86	29.83	27.95
CVCv3ACB-ave	4.68	5.91	6.3	6.58	20.65	28.59	29.82	31.42
ACB-bst	4.32	5.16	5.75	6.08	17.61	24.69	26.89	28.3
CVCv2ACB-bst	4.01	5.78	6.19	6.16	18.2	28.59	29.8	28.57
CVCv3ACB-bst	4.3	5.48	6.06	6.44	18.81	25.92	29.44	31.33
WCB-ave	4.93	5.43	6.01	6.13	22.26	25.97	29.13	29.87
CVCv2WCB-ave	3.85	5.28	6.12	5.86	16.47	26.52	28.46	27.17
CVCv3WCB-ave	4.57	5.79	5.47	6.63	19.85	27.98	28.5	32.49
WCB-bst	4.68	5.49	5.89	6.18	21.69	26.11	27.97	29.54
CVCv2WCB-bst	3.49	5.32	6.18	6.12	14.02	26.81	30.01	29.12
CVCv3WCB-bst	4.02	5.71	5.83	6.27	15.75	27.27	29.66	30.44

Table 8.15: Better combiner for the new methods

	Mean *IoP*				**Mean *PER***			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
CVCv2ACB	ave	ave	bst	bst	ave	ave	ave	bst
CVCv3ACB	ave	ave	ave	ave	ave	ave	ave	ave
CVCv2WCB	ave	bst	bst	bst	ave	bst	bst	bst
CVCv3WCB	ave	ave	bst	ave	ave	ave	bst	ave

Some important conclusions can be derived from the results shown in the two previous tables. Firstly, the new ensembles based on the combination of *Cross-Validated boosting* and *ACB* provide better results than *ACB* alone. However, there are a few cases in which *CVCv2ACB* is not better because of the inherited problems of *CVCv2* in small (3 networks) and high (40 networks) ensembles. Although *Output average* and *Boosting Combiner* work well, in general, on these new methods, it seems that *Output average* is the best combiner for *CVCv3* in both cases, *CVCv3ACB* and *CVCv3WCB*. In the case of *CVCv2* the result varies, the best combiner for *CVCv2ACB* is *Output average* except for the case where the number of networks is high (40 networks). And the best combiner for *CVCv2WCB* is the specific *Boosting Combiner* except for a low number of networks in the ensemble (3 networks).

Secondly, *CVCv2ACB* and *CVCv3ACB* have a good performance for any ensemble size. The ensembles of 40 networks trained with *CVCv3ACB* and combined with *Output average* provide a mean IoP of 6.58% and a mean PER of 31.42%. Furthermore, the ensembles of 9 networks trained with *CVCv2ACB* and combined with *Output average* provide excelent results, the mean IoP is 5.97% and the mean PER is 29.86%, and its performance is similar to the best classic ensembles according to the results shown in chapters 4 and 5, where the mean IoP and PER are, respectively, 6.16% and 28.68% on ensembles of 40 networks trained with *CVCv3*.

Thirdly, in the case of the cross-validated versions of *WCB*, the original boosting variant outperform the cross-validated version for a low number of networks in the ensemble (3 networks). For medium size ensembles (9 and 20 networks in the ensemble), the best alternative is *CVCv2WCB* with *Boosting Combiner* and, finally, for a high number of networks in the ensemble (40 networks) *CVCv3WCB* with *Output average* is the best alternative.

Fourthly, the best results are provided by the ensembles of 40 networks trained with *CVCv3WCB* and combined with *Output average* where the mean IoP is 6.63% and the mean PER is 32.49%. In this case, these values are the best overall results we have obtained in this thesis.

Finally, we have applied the t-test to compare the ensembles as it has been done in the previous sections to see if the results are statistically significant. In this case, *ACB*, *WCB* and the proposed ensembles are combined with *Output average* and *Boosting Combiner*, whereas the original boosting variants are combined only with *Boosting Combiner* as described in the original references.

The *Cross-Validated Boosting* versions of *ACB* and *WCB* are statistically compared to their original boosting variants in tables 8.16 and 8.17. Moreover, they have also been compared to *Adaboost*, *Aveboost* and *Conserboost*. With this procedure, it can be tested if the proposed methods perform better than *Adaboost*, which is the "first" boosting method or perform better than *Aveboost* and *Conserboost* which are the best boosting alternatives according to the ensemble comparison performed in previous chapters.

In table 8.16, we can see the results of the statistical tests related to the combination of *Cross-Validation Boosting* and *ACB*. First of all, better results are provided when *Output average* is applied to combine the networks instead of *Boosting Combiner* because the t-values related to *Output average* are lower. There are a few cases, specially in ensembles of 20 and 40 networks, in which *Boosting Combiner* is also a good combiner for *CVCv2ACB*. The conclusions derived from this table will be based on the use of the best combiner.

Table 8.16: Statistical test-*ACB* and *Cross-Validated Boosting*

Methods	measure	3-net	9-net	20-net	40-net
Adaboost vs CVCv2ACB-ave	t-**value**	−1.91	−4	−3.58	−3.62
	α	0.06	0.00009	0.00044	0.0004
Adaboost vs CVCv2ACB-bst	t-**value**	−1.44	−3.51	−3.68	−3.33
	α	0.15	0.0006	0.0003	0.00104
Adaboost vs CVCv3ACB-ave	t-**value**	−2.8	−3.86	−3.83	−4
	α	0.01	0.00015	0.00017	0.0001
Adaboost vs CVCv3ACB-bst	t-**value**	−2.06	−2.85	−3.25	−4.19
	α	0.041	0.0048	0.0014	0.00004
Aveboost vs CVCv2ACB-ave	t-**value**	0.42	−2.16	−1.14	−0.64
	α	0.67	0.032	0.26	0.52
Aveboost vs CVCv2ACB-bst	t-**value**	1.18	−1.04	−1.38	0.41
	α	0.24	0.3	0.17	0.69
Aveboost vs CVCv3ACB-ave	t-**value**	−1.17	−1.95	−2.05	−2.13
	α	0.24	0.05	0.042	0.035
Aveboost vs CVCv3ACB-bst	t-**value**	0.13	0.52	−0.57	−3.2
	α	0.9	0.61	0.57	0.0016
Conserboost vs CVCv2ACB-ave	t-**value**	0.8	−3.1	−2.39	0.56
	α	0.42	0.0022	0.018	0.58
Conserboost vs CVCv2ACB-bst	t-**value**	1.61	−2.07	−2.57	−0.55
	α	0.11	0.039	0.011	0.58
Conserboost vs CVCv3ACB-ave	t-**value**	−0.91	−2.83	−3.24	−3.02
	α	0.36	0.005	0.0014	0.0029
Conserboost vs CVCv3ACB-bst	t-**value**	0.5	−0.84	−1.85	−2.21
	α	0.62	0.4	0.07	0.03
ACB-ave vs CVCv2ACB-ave	t-**value**	1.41	−2.19	−1.42	0.4
	α	0.16	0.03	0.16	0.69
ACB-bst vs CVCv2ACB-bst	t-**value**	1.27	−3.03	−2.32	−0.4
	α	0.21	0.003	0.021	0.69
ACB-ave vs CVCv3ACB-ave	t-**value**	−0.34	−1.72	−2.39	−3.61
	α	0.73	0.09	0.018	0.00039
ACB-bst vs CVCv3ACB-bst	t-**value**	0.13	−1.41	−1.54	−2.05
	α	0.9	0.16	0.13	0.04

Secondly, we can see that the new ensembles, *CVCv2ACB* and *CVCv3ACB*, provide better statistical results (t-value is negative), in general, than the original alternatives except for the case of 3 networks in the ensemble. In this specific case, ensembles of 3 networks, the results are not statistically significant because α is,

in general, higher than 0.05. Moreover, in the case of 3 networks in the ensemble, *CVCv3ACB* is only slightly better than the original boosting alternatives but *CVCv2ACB* is slightly worse than *Aveboost*, *Conserboost* and *ACB* according to the positive t-values.

Thirdly, *CVCv2ACB* and *CVCv3ACB* are better than *Adaboost* and their differences are statistically significant in all the cases except for low sized ensembles (3 networks) generated by *CVCv2ACB* where α is higher than 0.05.

Fourthly, *CVCv3ACB* performs better than *Aveboost* according to the t-value. For this method the differences are statistically significant for ensembles of 9, 20 and 40 networks. *CVCv2ACB* performs better than *Aveboost* and the differences are statistically significant only for the case of 9 networks in the ensemble. In the other three cases (ensembles of 3, 20 and 40 networks) α is higher than 0.05 but, only for low sized ensembles (3 networks), the t-value denotes that *Aveboost* is slightly better.

In addition, the new ensembles, *CVCv2ACB* and *CVCv3ACB*, are better than *Conserboost*, in general, but the differences are statistically significant only for the cases of ensembles of 9 and 20 networks. Moreover, it also occurs for *CVCv3ACB* and ensembles of 40 networks. In the other cases, α does not denote a statistical difference but the new ensembles tend to be slightly better than *Conserboost* except for the ensembles of 3 networks generated by *CVCv2ACB*.

Furthermore, the statistical results shows that *CVCv3ACB* improves the original *ACB* but the differences are statistically significant only for medium and high sized ensembles (20 and 40 networks in the ensemble). In the other two cases the differences are not statistically significant because α is higher than 0.05. Moreover, *CVCv2ACB* is better than *ACB* and their differences are statistically significant only for the case of 9 networks in the ensemble. In the other cases, the original *ACB* is slightly better for low sized ensembles (ensembles of 3 networks) but it is slightly worse for medium-high sized ensembles (20 and 40 networks).

Finally, *CVCv3ACB* should be considered better alternative, in general, than *CVCv2ACB* because it reports better general and statistical results. Considering the best combiner, *CVCv3ACB* is statistically better than the original boosting alternatives in 12 of 16 cases but *CVCv2ACB* is only statistically better in 8 cases. Moreover, *CVCv3ACB* is never worse than any original boosting alternative considering the best combiner, whereas *CVCv2ACB* performs slightly worse than *Aveboost*, *Conserboost* and *ACB* for low sized ensembles (3 networks). Furthermore, when the proposed ensembles are statistically compared to the four original alternatives, the best t-value is given in 12 of 16 cases by *CVCv3ACB*. The other 4 cases, in which *CVCv2ACB* provides better t-values, always correspond to 9 networks in the ensemble.

In table 8.17, the results of the statistical tests of the new methods based on *WCB* are shown. Firstly, it seems that, in general, *Output average* fits better in

CVCv3WCB and *Boosting Combiner* is better choice for *CVCv2WCB* according to the resume table and the statistical results. Furthermore, *Output average* also provides good results for low sized ensembles (3 networks) generated by *CVCv2WCB*. In this case, the following analysis will be also based on the best combiner.

Table 8.17: Statistical test- *WCB* and *Cross-Validated Boosting*

Methods	measure	3-net	9-net	20-net	40-net
Adaboost vs CVCv2WCB-ave	t-**value**	−1.12	−2.33	−3.39	−3.21
	α	0.26	0.021	0.00086	0.0016
Adaboost vs CVCv2WCB-bst	t-**value**	−0.35	−2.42	−3.51	−3.6
	α	0.72	0.016	0.00056	0.00041
Adaboost vs CVCv3WCB-ave	t-**value**	−2.57	−3.57	−1.74	−3.82
	α	0.01	0.00045	0.08	0.00018
Adaboost vs CVCv3WCB-bst	t-**value**	−1.45	−3.28	−2.75	−3.72
	α	0.15	0.0013	0.0066	0.00026
Aveboost vs CVCv2WCB-ave	t-**value**	1.31	1.17	−0.83	0.94
	α	0.19	0.24	0.41	0.35
Aveboost vs CVCv2WCB-bst	t-**value**	2.2	0.98	−1.2	−0.39
	α	0.03	0.33	0.23	0.7
Aveboost vs CVCv3WCB-ave	t-**value**	−0.8	−1.1	1.5	−1.47
	α	0.42	0.27	0.14	0.14
Aveboost vs CVCv3WCB-bst	t-**value**	1.06	−0.6	0.53	−1.14
	α	0.29	0.55	0.6	0.26
Conserboost vs CVCv2WCB-ave	t-**value**	1.65	0.11	−2.04	0.98
	α	0.1	0.92	0.043	0.33
Conserboost vs CVCv2WCB-bst	t-**value**	2.62	−0.04	−2.38	−0.31
	α	0.01	0.97	0.018	0.75
Conserboost vs CVCv3WCB-ave	t-**value**	−0.5	−2.22	0.52	−1.31
	α	0.62	0.027	0.6	0.19
Conserboost vs CVCv3WCB-bst	t-**value**	1.39	−1.74	−0.77	−1.03
	α	0.17	0.08	0.44	0.3
WCB-ave vs CVCv2WCB-ave	t-**value**	2.98	0.58	−0.49	1.29
	α	0.003	0.56	0.63	0.2
WCB-bst vs CVCv2WCB-bst	t-**value**	3.27	0.62	−1.41	0.26
	α	0.0013	0.53	0.16	0.8
WCB-ave vs CVCv3WCB-ave	t-**value**	1.27	−1.57	1.64	−1.04
	α	0.21	0.12	0.1	0.3
WCB-bst vs CVCv3WCB-bst	t-**value**	2.35	−0.79	0.23	−0.49
	α	0.02	0.43	0.82	0.62

Secondly, *Adaboost* performs worse than *CVCv2WCB* and *CVCv3WCB* and the differences are statistically significant in all the cases except for the case of *CVCv2WCB* and 3 networks in the ensemble because α is quite higher than 0.05.

Thirdly, the differences among the new ensembles and the other boosting alternatives (*Aveboost*, *Conserboost* and *WCB*) are not statistically significant because α is greater than 0.05, in general. There are two cases, ensembles of 9 networks generated by *CVCv3WCB* and ensembles of 20 networks generated by *CVCv2WCB*, in which

Conserboost performs worse than the proposed methods and the differences are statistically significant. Moreover, there is a special case, 3 networks in the ensemble, in which *CVCv2WCB* performs statistically worse than *WCB* considering the "best" combiner because the t-value is positive and α is lower than 0.05.

Finally, according to the t-values, *CVCv3WCB* performs better than *CVCv2WCB* except for the medium sized ensembles (20 networks) because the t-values provided by *CVCv3WCB* are lower. i.e, *Aveboost vs CVCv2WCB-ave* provides a t-value of 0.94 for 40 networks but this value is -1.47 when the same original ensemble (*Aveboost* of 40 networks) is statistically compared to *CVCv3WCB-ave*. In this example, we can see how the t-value of *CVCv3WCB-ave* is lower than the one provided by *CVCv2WCB-ave* when they are compared to the same original ensemble.

We can conclude that we have successfully combined *ACB* and *WCB* with *Cross-Validation*. On the one hand, *CVCv3WCB* provides the best overall results according to the general measurements. On the other hand, *CVCv3ACB* statistically improves in same cases the best boosting methods such as *Adaboost*, *Aveboost*, *Conserboost* and *ACB*.

8.7 Conclusions

Several new ensembles based on *Adaptive Boosting* have been proposed in this chapter. First, *Averaged-Conservative Boosting* and *Weighted-Conservative Boosting* have been analyzed. Second, the *Cross-Validated Boosting* methodology was successfully introduced.

As it is shown during this chapter, *ACB* and *WCB* slightly improve the boosting variants they were based on, despite the results are not statistically significant. *ACB* is a bridge method between *Aveboost* and *Conserboost*. *ACB* is interesting because it successfully mixes two approaches, *Aveboost* and *Conserboost*, which differ on generating the different training sets. *WCB* provides better general results than any ensemble alternative described in chapter 5. This new ensemble is also interesting because the equation used to update the sampling distribution can adapt its strength depending on the problem because the value of the new parameter added its "optimized" to the problem.

Though these two new boosting alternatives can be considered only slight improvements on *boosting* methods, they provide good results which may be probably improved by refining them.

Secondly, a completely new ensemble methodology, *Cross-Validated Boosting*, has been successfully proposed. With this new methodology we can propose several new ensembles since it has been applied to:

- 7 classic boosting variants.
- 2 versions based on *Cross-Validation Committee*.
- 2 ensemble combiners.

Furthermore, it is a new methodology, it will be probably possible to employ it with any *CVC* version or *Boosting* method proposed in the future.

In general, the new ensembles based on *Cross-Validated Boosting* improve *Adaptive Boosting* and the original boosting variant they were based on, specially when the new ensemble models are based on *CVCv3* and the combiner applied is *Output average*. *Boosting combiner* also provides good results when it is applied to ensembles of 40 networks previously trained with the proposed *Cross-Validated Boosting* based on *CVCv3*.

Thirdly, the new alternatives *ACB* and *WCB* have been also used to generate *Cross-Validated Boosting* variants. Four new ensembles are introduced: *CVCv2ACB*, *CVCv3ACB*, *CVCv2WCB* and *CVCv3WCB*. As we pointed out before, *Cross-Validated Boosting* is a new methodology and can be applied to new *Boosting* methods such as *ACB* and *WCB*. On the one hand, *ACB* and *WCB* provide excellent results. On the other hand, the classic boosting variants have been improved when they have been mixed with *CVC* methods. Among the four new variants, *CVCv3ACB* and *CVCv3WCB* should be seriously considered. Furthermore, the combiner *Output average* is, in general, the most appropriate way to combine the networks in these two new ensembles. *CVCv3ACB* statistically improves the results provided by *Adaboost*, *Conserboost*, *Aveboost* and *ACB* for the case of ensembles of 20 and 40 networks. It is the first boosting alternative that statistically improves these four classic ensembles. *CVCv3WCB* provides the best overall results of this thesis and its best mean IoP, in ensembles of 40 networks, is higher than 6.5%. We consider that both new methods, *CVCv3ACB* and *CVCv3WCB*, are quite interesting according to all the measurements and tests applied.

Finally, the best overall ensemble performance is provided by 40-net ensembles trained by *CVCv3WCB* and combined by *Output average*. The maximum value of the mean *IoP* is 6.63% and the mean *PER* is 32.49%. The results of the original ensembles, *CVCv3* and *WCB*, were close to 6.15% (IoP) and lower than 30% (PER) so *CVCv3WCB* provides an extra increase of mean IoP close to 0.5% and mean PER higher than 2.5% with respect to the original ensembles. Moreover, ensembles of 40 networks trained with *CVCv3ACB* and *CVCv3Conserboost* also provide excellent results.

To conclude this chapter it is important to remark that good results have been obtained with the proposed ensemble models. We consider that the new methods based on *Cross-Validated Boosting* and the boosting variants *ACB* and *WCB* are the best overall ensemble alternatives. First, *CVCv3ACB* has improved the classic boosting variants and *ACB*. Furthermore, the differences with respect to them are statistically significant in some cases. Second, *CVCv3WCB* is slightly better than the best classic boosting alternatives but it provides the best overall results of this thesis for the case of 40 networks in the ensemble. Both, *CVCv3ACB* and *CVCv3WCB*, are based in the new methodology, *Cross-Validation Boosting*, and in new *boosting methods*, *ACB* or *WCB*. All of them have been proposed in this thesis. For this

reason, it is also important to remark that any slight improvement to an ensemble should be seriously considered. A sequence of improvements is the way to obtain the best ensembles because an *original* ensemble proposal is slightly refined step by step. In the case of *CVCv3WCB*, the *final* ensemble is a high refined version of the *original* ensembles *CVCv1* and *Adaboost*.

The results related to *ACB*, *WCB* and *Cross-Validated Boosting* have been published in references [MFBI1,MFBI2,MFBI3]. Their description can be found in the conclusions chapter.

Chapter 9

Stacked Generalization

9.1 Introduction

In the previous chapters the research has been focused on analyzing and improving *Multiple Classifier Systems* based on the *Ensemble model*. However, there are some other important approaches to generate *Multi-net* systems.

Among the approaches to build a classification system, *Stacked Generalization* is a well-known model. *Stacked Generalization* is an architecture which is composed by two or more layers of generalizers. A generalizer is a "black box" which solves any kind of problem and it can be a classifier (neural net). Firstly, the level-0 generalizers solve the original problem, whereas level-1 generalizers process the information provided by the previous layer in order to give a single output of the system.

This chapter, which will be focused on the application of the principles of *Stacked Generalization* in ensembles of neural networks and it is organized as follows. Firstly, in section 9.2 the original *Stacked Generalization* procedure is described and two new combiner methods based in principles similar to *Stacked Generalization* are proposed. Secondly, the experimental setup of the experiments is introduced in section 9.3. Finally, the results and their discussion are shown in section 9.4.

9.2 Stacked Generalization

Although most of the methods to create a Multi-Net System are based on the *ensemble approach* (*Boosting*, *Bagging*, *Cross-Validation*), there are important approaches, like *Stacked Generalization*, in which the neural networks can also be introduced.

Stacked Generalization was introduced by Wolpert in 1994 [95] as a scheme for minimizing the generalization error rate of one or more generalizers. The *Stacked* model consists of two or more layers of generalizers as shown in figure 9.1.

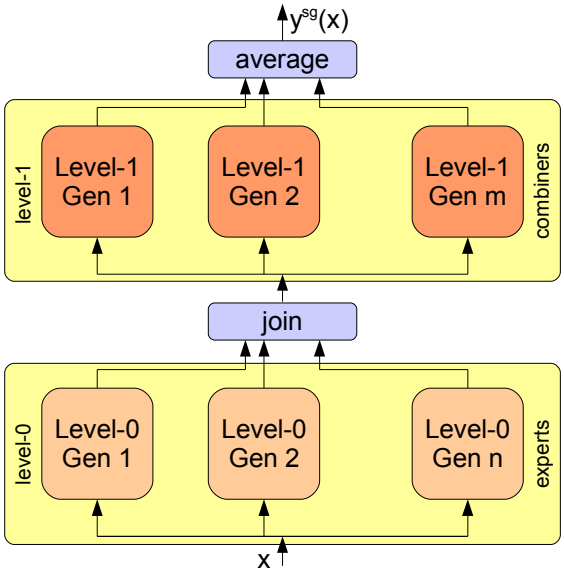


Figure 9.1: Stacked Generalization model

In first place, the generalizers of the level-0 (experts) solve the original problem. Secondly, the generalizers of the second layer, level-1, process the information provided by the previous layer, the outputs of the experts, in order to reduce the bias of the generalizers and give a more accurate output. Wolpert proposed this scheme to provide a technique which achieves a generalization accuracy as higher as possible. Moreover, Wolpert also introduced a specific procedure to design the different learnings sets for the level-0 and level-1 generalizers.

At the beginning of the training procedure, a set of generalizers called *level-0 generalizers* are built using the original input data, x, and the class label, $d(x)$. Each generalizer is trained with the patterns of a specific training set T^i. Then, a set of generalizers called *level-1 generalizers* are designed using the patterns, z, of the level-1 training set, T^{sg}, and the class label $d(z)$. A pattern from T^{sg}, z, is given by the outputs provided by *level-0* generalizers, $[y^1(z), \ldots, y^n(z)]$, on this pattern. Figure 9.2 shows how T^{sg} is generated.

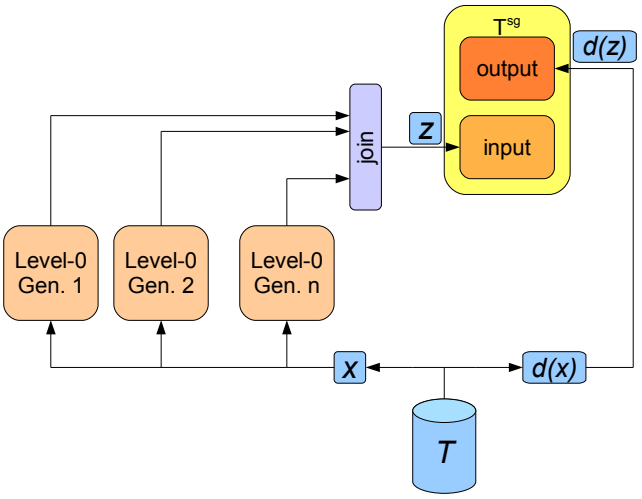


Figure 9.2: Generating the level-1 training set of Stacked Generalization

Wolpert proposed *Stacked Generalization* in a general way. Some authors like *Ghorbani & Owrangh* [113] and *Ting & Witten* [114, 115] have implemented specific versions of Wolpert's model which can be directly applied to neural networks. These two implementation will be reviewed in this section.

The original *Stacked Generalization* can be incorrectly catalogued as an ensemble combiner because the level-0 generalizers may be wrongly identified as an ensemble and the level-1 generalizers as their combiner. The training procedure employed in the scheme proposed by *Wolpert* involves the level-0 and level-1 generalizers and they can not be trained independently so *Stacked Generalization* is a new model different to the *ensemble* approach. However, the level-1 generalizers can be applied as an ensemble combiner if the constraints that links *level-0* and *level-1* training are omitted. In this way, two combiners called *Stacked* and *Stacked+* will be proposed in this chapter.

Finally, we will show a comparison among all the implementations and combiners based on the *Stacked* model.

9.2.1 Ting & Witten's implementation

Ting & Witten proposed a version of *Stacked Generalization* that can be applied to the *Multilayer Feedforward* architecture in [114, 116] and it was reviewed in [115].

In this implementation we will talk about expert networks, or experts, and combination networks. These networks are, respectively, the *level-0* and *level-1* generalizers according to the original model. The number of expert and combination networks are, respectively, $N_{experts}$ and $N_{combiners}$.

Firstly, the training set is randomly split into k equal subsets by applying k-fold cross-validation. According to the original references, k does not depend on the number of the experts. The subsets, T_i, of the original training set, T, are given by the following equation:

$$T = \{T_1, T_2, \ldots, T_k\} \tag{9.1}$$

Secondly, all the expert networks are trained with the same specific training set, $T^{experts}$ or $\mathcal{L}^{-j}$ according to the original reference. This specific training set corresponds to the union of all the previous subsets except one, j.

$$T^{experts} = \mathcal{L}^{-j} = \{T - T_j\} = \bigcup_{\substack{i=1 \\ i \neq j}}^{k} T_i \tag{9.2}$$

Finally, the combination networks are trained with the patterns from a specific training set, T^{sg} or $\mathcal{L}_{CV}$ according to the original reference. This new set has the patterns from the subset which were not used in the training process of the experts, T_j. With this procedure, we are training the combination networks with patterns which were not used to train the experts. The patterns which are used to train the combination networks are:

$$T_j = \{x_1, \ldots, x_{Subset_{elem}}\} \tag{9.3}$$

However, the data provided by the patterns from T_j are not enough to train the *combination networks* because these networks process the outputs of the patterns from T_j. For this reason, the specific training set, T^{sg}, is really calculated with equation 9.4:

$$T^{sg} = \mathcal{L}_{CV} = \mathcal{L}^j = \bigcup_{i=1}^{Subset_{elem}} [(y^1(x_i), y^2(x_i), \ldots, y^{N_{experts}}(x_i)); d(x_i)] \tag{9.4}$$

Where $(y^1(x_i), y^2(x_i), \ldots, y^{N_{nets}}(x_i))$ denotes the outputs of the experts on the i-th pattern x from T_j, whereas $d(x_i)$ denotes its desired output, or target. $Subset_{elem}$ denotes the number of patterns of each subset, including T_j.

Furthermore, this implementation of *Stacked* can not be successfully applied if the number of subsets, k, is high. The size of T^{sg} inversely depends on the number of subsets so it can not contain enough patterns to train the combination networks. In the original reference, k was set to 10 but we noticed that the size of $\mathcal{L}^j$ can be considerably low in the databases used in this thesis. So we performed some experiments with different values of k and we finally set the value of j and k to 5.

Figure 9.3 shows how the training set of the expert networks and combination networks are generated. In the figure, a n-experts *Stacked* system is supposed.

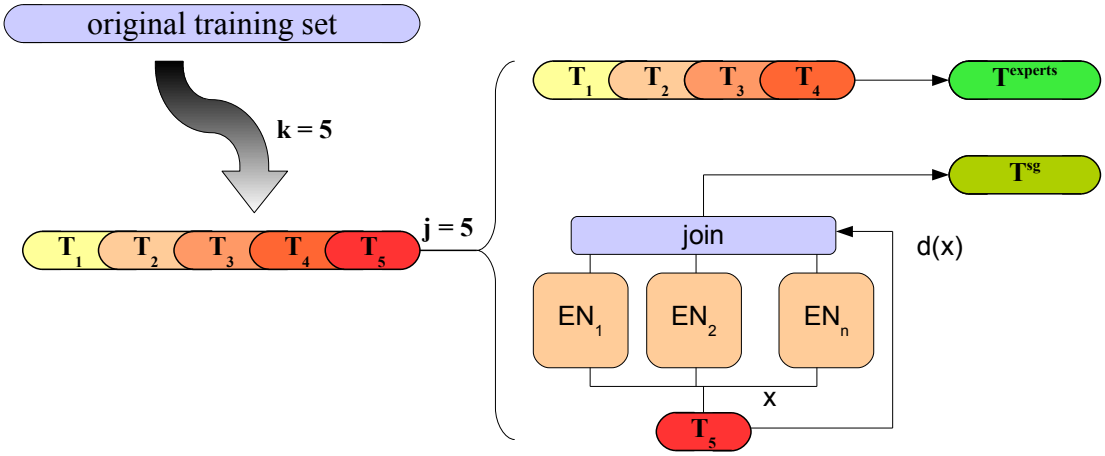


Figure 9.3: Stacked Generalization applied by Ting & Witten

This implementation was initially based on the following generalizers: *Decision Tree C4.5*, *Naive Bayesian Classifier* and *IB1*. However, this version proposed by *Ting & Witten* can also be applied to neural networks as shown in [117].

9.2.2 Ghorbani & Owrangh's implementation

Ghorbani & Owrangh proposed a version of *Stacked Generalization* that was applied directly to Artificial Neural Networks [113]. In this implementation, *Cross-validation* is also applied to create the different training sets of the experts by randomly splitting the training set into k equal subsets:

$$T = \{T_1, T_2, \ldots, T_k\} \tag{9.5}$$

With this procedure, k different experts can be built with different training sets by changing the subset that is left out. For this reason, k corresponds to the number of expert networks $N_{experts}$ in this method.

Firstly, each expert network, *net*, is trained with its specific training set, T^{net}.

$$T^{net} = \bigcup_{\substack{i=1 \\ i \neq net}}^{N_{experts}} T_i \tag{9.6}$$

Then, the patterns from the union of all the subsets are used to train the combination networks.

$$\bigcup_{i=1}^{N_{experts}} T_i = \{x_1, \ldots, x_{N_{experts} \cdot Subset_{elem}}\} \tag{9.7}$$

However, the input data provided by the patterns from the union of all the subsets is not used to train the *combination networks* because these networks only process the outputs of the experts. For this reason, the specific training set, T^{sg}, is really given by the following equation:

$$T^{sg} = \bigcup_{i=1}^{N_{experts} \cdot Subset_{elem}} [(y^1(x_i), y^2(x_i), \ldots, y^{N_{experts}}(x_i)); d(x_i)] \tag{9.8}$$

Where $(y^1(x_i), y^2(x_i), \ldots, y^{N_{nets}}(x_i))$ are the outputs of the experts on the i-th pattern x from the union of all the subsets, $d(x_i)$ is the desired output of the pattern, and finally, $Subset_{elem}$ is the number of patterns of each subset.

In figure 9.4, we show how the training set of the expert networks and combination networks are generated for a system with 5 experts.

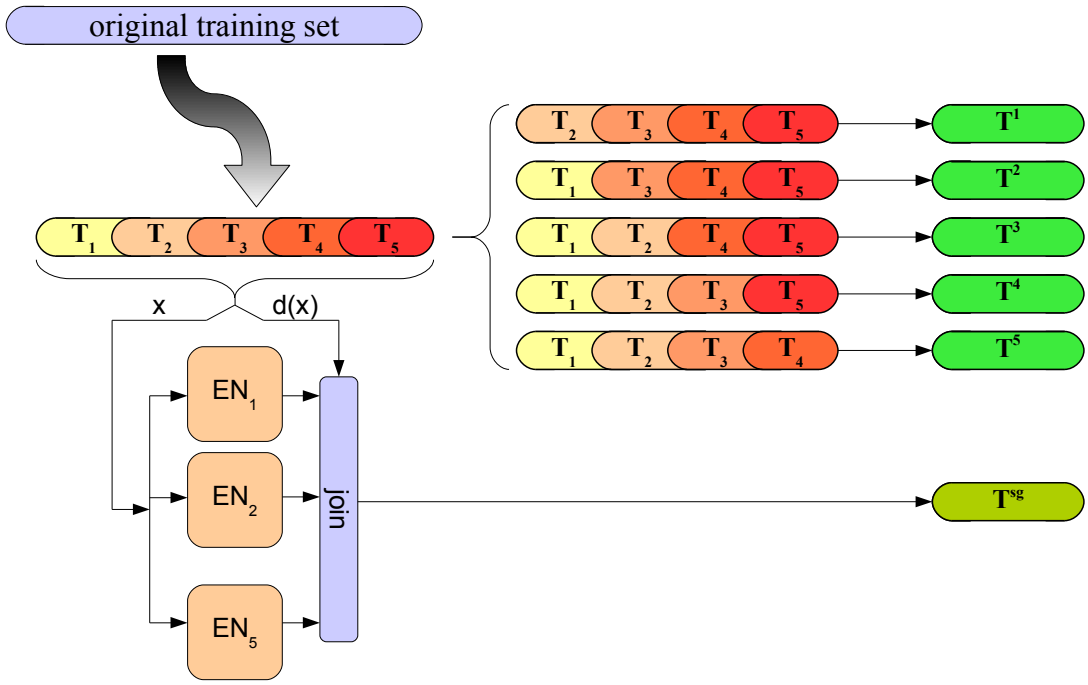


Figure 9.4: Stacked Generalization applied by Ghorbany & Owrangh

9.2.3 Two new combiners based on *Stacked Generalization*

Although the *Stacked* model corresponds to a specific *Multi-net* system, it can also be introduced in ensembles of neural networks. In fact, an optimal neural network can be applied to combine the outputs provided by the networks of an ensemble and reduce the error of the entire system [118, 119]. This combination network can learn the mistakes done by the ensemble in order to avoid them.

In the *Stacked combiners* proposed in this thesis, *Stacked* and *Stacked+*, there is not any associated constraint related to the training sets of the combination networks. The predictions of the networks of the ensemble on the whole training set are used to train the *combination networks*. These two combiners should not be considered as pure 2-leveled models because the two levels are independently generated.

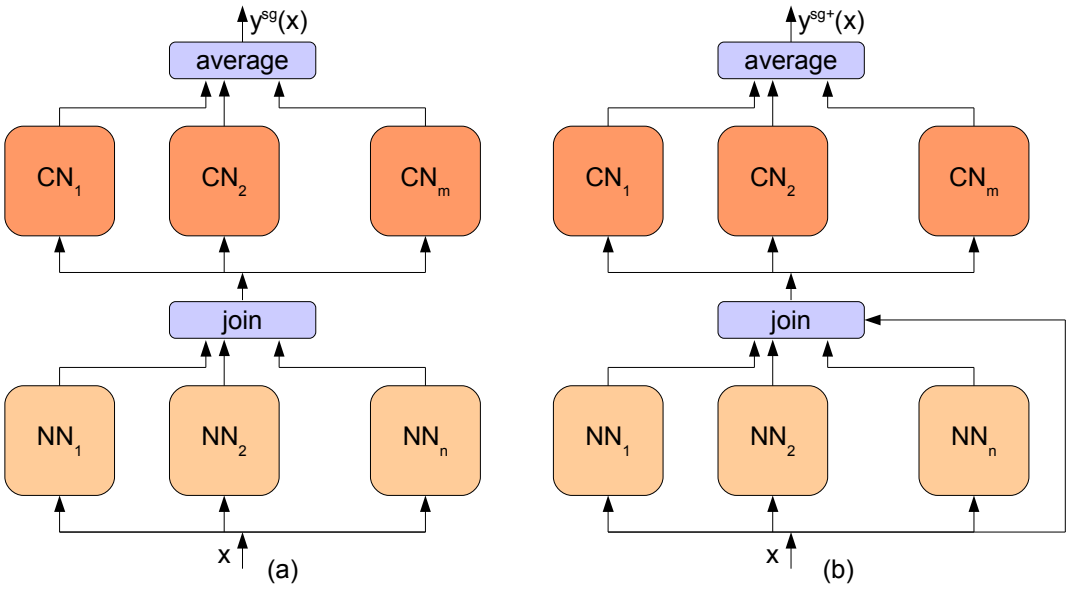


Figure 9.5: Diagrams of a) *Stacked* & b) *Stacked+* combiners

It can be noticed in the previous figure that the predictions of the networks are the basis of the combination networks training in the combiner *Stacked*, whereas in *Stacked+* the original input data, x, is also used. We consider that the use of the original input data may be important in the combination networks. The training sets for the combination networks in *Stacked*, T^{sg}, and *Stacked+*, T^{sg+}, are given by:

$$T^{sg} = \bigcup_{i=1}^{N_{patterns}} \left[(y^1(x_i), \ldots, y^k(x_i)); d(x_i)\right] \tag{9.9}$$

$$T^{sg+} = \bigcup_{i=1}^{N_{patterns}} \left[(y^1(x_i), \ldots, y^k(x_i), x_i); d(x_i)\right] \tag{9.10}$$

Where x_i is the input data of the i-th pattern, $y^1(x_i), \ldots, y^k(x_i)$ are the outputs of the experts on this pattern and $d(x_i)$ is its desired output or target.

9.3 Experimental Setup

In this chapter, the *Stacked Generalization* model, as Multi-Net system and two new ensemble combiners based on the ideas of *Stacked Generalization* are studied. A deep comparison among them is performed.

As in the previous ensemble comparisons, a similar experimental setup is employed to test the performance of *Stacked Generalization*. However, there are some special features because this model differs from the ensemble model.

Firstly, we test the two implementations of *Stacked Generalization* as Multi-Net System. These implementations were proposed by *Ting & Witten* and *Ghorbani & Owrangh*.

Secondly, the new combiners, *Stacked* and *Stacked+*, are tested using a single *MF* combination network in the level-1. In this case, four ensembles of *MF* networks (*Simple Ensemble*, *Adaboost*, *CVCv3Conserboost* and *Inverboost*) are used as experts (level-0). Moreover, an ensemble of *RBF* networks (*Simple Ensemble*) is also employed as level-0 generalizers.

Thirdly, *Stacked* and *Stacked+*, are tested using ensembles of *MF* combination networks as level-1 generalizers. In this case, *Simple Ensemble* is used to build the level-0 generalizers and three ensembles of *MF* networks (*Simple Ensemble*, *CVCv3ACB* and *CVCv3Conserboost*) are employed to generate the combination networks (level-1).

Finally, the main characteristics of the performed comparison are:

- Nineteen datasets from the *UCI Repository*.
- Optimized training parameters for expert and combination networks.
- Experiments repeated ten times with different partitions of the training, validation and test sets to obtain:
  - Mean value of performance.
  - Error rate by standard error theory.
- Three general measurements applied to the comparison:
  - Mean *Increase of Performance*.
  - Mean *Percentage of Error Reduction*.
  - *Paired Student's t-test*.

The description of the nineteen datasets used in the experiments can be found in appendix A. The optimized training parameters of the expert networks and combination networks are in appendixes B.3, B.4 and B.7. Finally, the general measurements applied to compare the combiners studied are the mean *Increase of Performance* (equation 3.5), the mean *Percentage of Error Reduction* (equation 3.6) and the *Student's Paired t-test*, described in subsection 3.6.3.2.

9.4 Results and discussion

The results of the original *Stacked* systems and the combiners based on the *Stacked Generalization* principles, *Stacked* and *Stacked+*, studied in this chapter are shown in this section. As previously mentioned, the *Stacked* approach can be applied in different ways. For this reason, this section has been split into three subsections. The first one will be focused on the original *Stacked* models shown in section 9.2. The second one will be focused on the results provided by combining ensembles of neural networks with a single network using *Stacked* and *Stacked+*. Finally, the last subsection will show the results of combining ensembles, expert networks trained with *Simple Ensemble*, with another ensemble of combination networks.

9.4.1 Original Stacked methods

In this subsection, the results obtained by the implementations proposed by *Ting & Witten*, denoted with *T&W* in the tables, and *Ghorbani & Owrangh*, *G&O* in the tables, are shown. Firstly, table 9.1 shows the general results of these models. Moreover, the results of the experts combined by *Output average* as an ensemble are also shown. In the table, we denote with " - *Stacked*" the general results of the whole model with the implementations previously described and " - *Experts*" refers to the results of the ensemble only composed by the expert networks.

Table 9.1: Original Stacked Models

	Mean IoP				**Mean** PER			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Simple Ensemble	4.97	5.27	5.34	5.43	21.84	23.65	23.73	24.64
T&W - Experts	4.37	4.63	4.76	4.83	17.70	19.41	19.66	19.60
T&W - Stacked	4.46	4.81	5.06	5.18	18.87	20.26	21.52	22.00
G&O - Experts	4.71	5.56	5.64	5.46	20.59	25.69	25.21	23.72
G&O - Stacked	5.02	5.69	5.29	5.45	22.75	26.03	22.70	23.73

At first sight, it can be noticed that the version proposed by *Ting & Witten* performs worse than *Simple Ensemble*. Moreover, the ensembles conformed by their experts are also outperformed by *Simple Ensemble*. This behavior was expected for this *Stacked* model because the training set applied to generate the combination network has a lower number of training examples than the one used in *Simple Ensemble*.

Moreover, the version proposed by *Ghorbani & Owrangh* slightly improves the results provided by *Simple Ensemble* in the cases of a low number of expert networks (3 and 9 networks). Moreover, the results provided by this model improve the results provided by the ensemble composed by their expert networks in the same cases (3 and 9 networks).

Furthermore, it is interesting to printout that in the case of *Ghorbani & Owrangh* and a high number of experts (20 and 40 experts), the simple *Output Average* of the experts is a better combination system than *Stacked*. The reason may be the curse of dimensionality of the stacked network whose inputs are the outputs of a 40

networks system, i.e., the number of inputs of this network seems to be high and the number of patterns in the training set is low to successfully train such a high input dimensional network.

The *t-test* was applied to determine if the slight improvement provided by the experts and Stacked model proposed by *Ghorbani & Owrangh* is statistically significant. Firstly, the results provided by *Simple Ensemble* are compared to the results of the experts networks as ensemble combined with *Output Average* and to the whole Stacked model. Finally, the results provided by the experts as ensemble are statistically compared to the results provided by the whole Stacked model. The results of this comparisons are in table 9.2 which show that the differences among *Simple Ensemble*, the experts as ensemble and the whole model are not statistically significant except for the case of 9 networks and the comparison of *Simple Ensemble* with *Ghorbani & Owrangh* Stacked model where *Stacked* supposes a statistically improvement of *Simple Ensemble*.

Table 9.2: Statistical Results - Original Stacked Models

Methods	measure	3-net	9-net	20-net	40-net
SE vs G&O - Experts	t-**value**	1.09	−1.63	−1.83	−0.18
	α	0.28	0.10	0.07	0.86
SE vs G&O - Stacked	t-**value**	−0.04	−2.11	0.25	−0.15
	α	0.97	0.036	0.80	0.88
G&O - Experts vs G&O - Stacked	t-**value**	−1.99	−0.76	1.83	0.03
	α	0.05	0.45	0.07	0.98

Finally, the best overall results are provided by the model proposed by *Ghorbani & Owrangh* with 9 expert networks. This concrete model, nine experts and one combination network, slightly outperforms the ensembles of 40 networks generated by *Simple Ensemble* without training a high number of networks. As the results of the experts of *Ghorbani & Owrangh* shows, the proposed model maybe do not fit well on ensembles of a high number of networks (40 networks) because the experts are generated as in *CVCv1* (see references [73, 74]) so they have the same drawbacks related to the size of the specific training sets.

9.4.2 Ensemble combiners based on the *stacked* idea

The *Stacked* model can be introduced to propose new combiners of ensembles, the combination networks learn from the output provided by the networks. In these new combiners, there is not any constraint that directly links the training set applied to train the ensemble and the training set applied to train the combination networks so it is not exactly a *Stacked model*.

In this case, we will apply a single combination network to fuse the outputs provided by previously trained ensembles. The networks of the ensemble are trained with a "full" training set and the combiner network is also trained with a "full" training

set provided by the output of the ensemble and the targets of the original training set.

We will perform some experiments on ensembles of *MF* networks and a few ones on ensembles of *RBF* networks. The results of these experiments are shown, respectively, in subsections 9.4.2.1 and 9.4.2.2.

9.4.2.1 Combining an ensemble of *MF* networks

In the case of *MF* networks, the new combiners *Stacked* and *Stacked+* have been applied to the following ensembles:

- *Simple Ensemble*.
- *Adaptive Boosting*.
- *CVCv3Conserboost*.
- *Inverboost*.

Simple Ensemble and *Adaptive Boosting* (*Adaboost*) have been selected because they are simple and original ensemble methods. *CVCv3Conserboost* was selected because it is one of the best overall ensembles. We have not obtained results with the *CVC* versions due to the similarity with the model proposed by *Ghorbany & Owrangh*.

Inverboost was selected because it is a special variant which try to overfit the training set with easy to learn patters. Applying the *Stacked model* can provide interesting results because the ensemble performs worse than a single network.

The results of the selected ensembles along with the results provided by *Stacked* and *Stacked+* are shown in table 9.3. The notation in the table *STC* means that the *Stacked* combiner is applied to the particular ensemble, and *STCP* means that in this case the combiner is *Stacked+* (*Stacked Plus*). The name of the ensemble method applied as expert networks is included and *SN* is also included to denote a single combination network.

Table 9.3: Combining ensembles of *MF* networks with *Stacked*

	Mean *IoP*				Mean *PER*			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Simple Ensemble	4.97	5.27	5.34	5.43	21.84	23.65	23.73	24.64
STC-SE-SN	5.30	5.72	5.74	5.74	23.79	26.44	27.15	27.52
STCP-SE-SN	5.12	5.63	5.78	5.62	22.63	26.38	27.26	26.71
Adaboost	3.69	4.55	4.93	4.98	15.40	19.50	22.96	24.54
STC-Adaboost-SN	4.22	4.51	4.45	4.45	18.57	18.33	19.33	19.70
STCP-Adaboost-SN	4.50	4.72	4.57	4.36	20.95	21.34	20.18	20.02
CVCv3Conserboost	4.09	5.70	6.29	6.50	17.17	27.45	30.18	31.29
STC-CVCv3Conserboost-SN	4.63	5.48	5.53	5.24	20.63	25.42	25.19	24.20
STCP-CVCv3Conserboost-SN	4.78	5.65	5.63	5.32	22.28	26.28	25.84	24.62
Inverboost	2.89	0.95	−1.23	−2.76	9.20	−3.12	−15.00	−24.18
STC-Inverboost-SN	3.69	4.15	4.07	4.11	11.38	17.77	17.43	16.22
STC-Inverboost-SN	4.50	4.72	4.57	4.36	20.95	21.34	20.18	20.02

Firstly, *Stacked* and *Stacked+* are successfully applied as combiner of the ensembles generated by *Simple Ensemble*. Although *Stacked* slightly performs better than *Stacked+* in general, both provide general good results. In this case, the additional information of the inputs of *Stacked+* increases the curse of dimensionality in the combination network but does not provide useful information, so the performance of *Stacked+* decreases with respect to *Stacked*.

Secondly, in the case of *Boosting* ensembles, *Stacked* and *Stacked+* improve the results of the boosting combiner only in the case of a low number of networks in the ensemble (3 and 9 networks for *Adaboost* and 3 networks for *Conserboost*). It seems again an effect of the curse of dimensionality. As the number of networks in the ensemble increases, the number of inputs of the combiner networks also increases (it is 40 times the number of classes of the classification problem) and, maybe, the number of examples in the training dataset it is not enough to train the combiner network with good generalization.

Similar results were obtained when the *Stacked* combiners were applied to other boosting variants such as: *Aveboost*, *ACB*, *WCB* and most of the boosting models based on *Cross-Validated Boosting*. The results of these experiments have not been published and they are not included in this thesis because their conclusions are similar to the conclusions reached with the three representative boosting methods shown in this subsection. We consider that adding more results to this research would not add new useful information because all the *Boosting* variants report similar analysis when the new *Stacked combiners* are used. Moreover, the comprehension of this subsection might be negatively affected because the huge number of results, in the table we show three rows for each ensemble alternative (original results, performance with *Stacked* and performance with *Stacked+*) so adding all these *Boosting* variants to the research supposes a large table of results because we have generated ensembles with more than 25 different boosting variants in this thesis.

Moreover, *Inverboost* is a *boosting* variant which has been highly improved with the new combiners. This improvement is quite important because the general results are good but the experts as ensemble are quite bad. The ensembles of 40 networks performed quite worse than a single network, concretely the mean IoP was -2.76%, whereas the results provided by the new Stacked combiners on these same ensembles are quite better than a single network, the mean IoP is higher than 4%. Applying *Stacked* and *Stacked+* have provided an increase of performance close to 7% with respect to the results provided by the original ensemble. It seems that *Stacked* and *Stacked+* can highly improve the performance of a "*bad*" ensemble correcting the decissions taken by a just simple combiner as the specific *Boosting Combiner* used in *Inverboost*.

9.4.2.2 Combining an ensemble of *RBF* networks

It can be seen in the previous subsection that the new Stacked combiners based on a single *MF* network can improve the accuracy of the ensembles of *MF* networks based on *Simple Ensemble*. For this reason, those combiners, which were based on

a single *MF* combination network, were also applied to combine ensembles of *RBF* networks.

Stacked and *Stacked+* have been only applied to ensembles previously trained with *Simple Ensemble* due to the similarity in the performance of all the ensemble methods with *RBF* networks. However, the outputs of a *RBF* network are not $[0, \ldots, 1]$ ranged. A normalization procedure is applied to the outputs of a *RBF* network in order to obtain values in the range $[0, \ldots, 1]$. As shown in chapter 6, there are combiners that can not fit well if the outputs are not in the mentioned range but they can provide good results when a normalization procedure is applied. We think that applying an output normalization can be also useful in *Stacked* and *Stacked+*. For this reason, the new Stacked combiners have been applied to fuse the original and normalized *RBF* networks. The three normalizations we have used (*threshold*, *min-max* and *sum*) are described in equations 6.66 to 6.68. The results of applying the Stacked combiners on *RBF* ensembles are in the following table:

Table 9.4: Combining ensembles of *RBF* networks with *Stacked*

	Mean IoP				**Mean** PER			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
RBFSE	0.02	0.14	0.14	0.23	0.15	3.71	3.06	3.29
STC RBFSE	0.04	0.34	−0.05	−0.13	2.58	4.56	0.4	−0.26
STCP RBFSE	−0.31	−0.04	−0.13	−0.05	−2.87	1.72	1.33	0.91
RBFSE$_{threshold}$	0.01	0.14	0.16	0.24	0.09	3.78	3.27	3.36
STC RBFSE$_{threshold}$	0.07	0.32	0.08	−0.01	3.24	4.97	3.56	0.83
STCP RBFSE$_{threshold}$	−0.19	−0.02	−0.16	−0.04	0.01	1.39	1.1	1.79
RBFSE$_{sum}$	0.01	0.13	0.14	0.20	0.28	3.88	3.05	3.02
STC RBFSE$_{sum}$	−0.02	0.17	0.12	−0.13	2.65	0.6	2.41	−0.23
STCP RBFSE$_{sum}$	−0.23	−0.06	0.00	−0.09	−0.77	1.96	1.24	−0.39
RBFSE$_{min-max}$	−0.02	0.16	0.15	0.18	1.69	4.23	3.41	3.29
STC RBFSE$_{min-max}$	−0.50	−0.56	−0.51	−0.51	−2.14	−6.26	−5.15	−4.57
STCP RBFSE$_{min-max}$	−0.71	−0.64	−0.37	−0.46	−3.9	−5.06	−1.78	0.02

At first sight, the new combiners perform better if the *threshold* normalization is applied. This is the only normalization procedure which reports positive PER for any ensemble size. In fact, the best overall results (mean PER) are provided by applying *Stacked* to the 9-net ensembles with *threshold* outputs (PER equal to 4.97%). However, there are some cases in which the Stacked combiners provide good results when they are applied to combine the networks without normalization, the highest mean IoP is obtained with *Stacked* and the original 9-net ensembles (IoP equal to 0.34%). *Stacked+* also provides good results with the *sum* normalization and medium sized ensembles (9 and 20 networks) but they provide negative values for the other sizes (ensembles of 3 and 40 networks).

Secondly, it can be noticed that *Stacked* provides better overall results, in general, than *Stacked+* when they are applied to ensembles of *RBF* networks. However, both combiners do not improve the original ensemble when the number of expert networks is high (20 or 40 networks).

Both effects may be explained again by the curse of dimensionality. In *Stacked+*, we introduce the original inputs of the training set as additional inputs of the combiner network. It seems that this extra information is not useful in this case and performs as a noise deteriorating the classification accuracy of the network.

In the case of *Stacked*, the performance deteriorates as the number of networks in the ensemble increases. An increase in the number of networks means a proportional increase in the number of inputs of the combiner network keeping the same size of the training set. So the size of the original training sets seems to be appropriate for training the combiner network only in ensembles of 3, 9 and, perhaps, 20 networks for the case of the *threshold* normalization.

Thirdly, *Stacked* and *Stacked+* perform worse than a single *RBF* network in some cases, specially when the *min-max* normalization is applied. It can be observed that there is an important decrease of performance when *Stacked* and *Stacked+* are applied to the ensembles with *min-max*-normalized outputs. For this kind of *RBF* networks, *Stacked+* provides worse results than a single *RBF* network but it is better than using the new combiner *Stacked*.

Finally, only *RBFSE* and $RBFSE_{threshold}$ have been improved by one new combiner, *Stacked* (*STC* in the table), when the number of experts (the number of networks in the ensemble) is low, 3 or 9 networks, and perhaps 20 networks.

9.4.3 *Stacked* and *Stacked+*: Ensembles as combination networks

In this subsection, an ensemble of combination networks, henceforth *combination ensemble*, will be applied to combine another ensemble of expert networks. In this case, an ensemble will be on charge of combining another ensemble as shown in figure 9.5.

Though any ensemble can be used to generate the experts and the combination networks, the ensemble alternatives applied will be limited. The ensembles previously trained with *Simple Ensemble* will be only employed as expert networks whereas a few ensemble models will be introduced to generate the combination networks according to *Stacked* and *Stacked+*.

The models selected for the *combination ensemble* were: *Simple Ensemble*, *CVCv3ACB* and *CVCv3Conserboost*. Their results are in tables 9.5 and 9.6. Although *Output average*, denoted by "*a*" in the tables, has been initially applied to combine the combination networks, *Boosting Combiner*, denoted by "*b*" in the tables, has also been applied to combine the combination ensembles based on *Cross-Validated Boosting*.

Firstly, the results of the *combination ensembles* designed with *Stacked* in order to combine ensembles of expert networks previously trained with *Simple Ensemble* are shown in table 9.5.

Table 9.5: *Stacked* with ensembles of combination networks

		Mean *IoP*				Mean *PER*			
Method	**CN**	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Simple Ensemble	–	4.97	5.27	5.34	5.43	21.84	23.65	23.73	24.64
SE-SN	1	5.3	5.72	5.74	5.74	23.79	26.44	27.15	27.52
SE-SE	3	5.28	5.79	5.69	5.76	23.56	26.95	27.05	27.44
SE-SE	5	5.32	5.81	5.68	5.75	23.95	27.15	26.95	27.44
SE-SE	9	5.3	5.81	5.69	5.72	23.88	27.13	26.8	27.33
SE-CVCv3ACB a	3	5.27	5.8	5.69	5.75	24.69	26.48	26.97	27.31
SE-CVCv3ACB a	5	5.28	5.82	5.81	5.79	24.74	26.89	27.71	28.37
SE-CVCv3ACB a	9	5.26	5.77	5.74	5.78	24.49	26.26	26.78	28.27
SE-CVCv3ACB b	3	5.32	5.83	5.67	5.82	24.74	26.26	26.7	27.11
SE-CVCv3ACB b	5	5.35	5.87	5.68	5.89	25.53	27.04	27.25	28.65
SE-CVCv3ACB b	9	5.34	5.92	5.71	5.87	25.02	27.43	27.55	28.23
SE-CVCv3Conserboost a	3	5.27	5.66	5.71	5.71	23.76	25.71	26.77	26.75
SE-CVCv3Conserboost a	5	5.19	5.69	5.74	5.64	22.99	25.89	26.72	25.69
SE-CVCv3Conserboost a	9	5.25	5.57	5.65	5.55	24.32	25.58	26.54	25.32
SE-CVCv3Conserboost b	3	5.29	5.58	5.76	5.6	24.38	25.05	27.44	25.83
SE-CVCv3Conserboost b	5	5.41	5.59	5.78	5.68	25.89	25.47	27.47	27.01
SE-CVCv3Conserboost b	9	5.38	5.73	5.81	5.6	25.78	26.3	28.35	26.67

The first four rows of table 9.5 show the results of combining ensembles previously trained with *Simple Ensemble* by the *combination ensembles*, also trained by *Simple Ensemble* as the combiner *Stacked*. The other rows show the results of combining the same ensembles by the other *combination ensembles* (*CVCv3ACB* and *CVCv3Conserboost*). Furthermore, in the table, *CN* refers to the number of networks in the combination ensemble. For example, in the third row we can find *SE-SE* as "Method" and the value 5 for "CN". This means that the expert networks are a *Simple Ensemble* (the first *SE* in *SE-SE*), the combiner networks are also a *Simple Ensemble* (the second *SE* in *SE-SE*) and the number of networks in the combiner ensemble is 5 ($CN = 5$). In the first row we can find also *SE-SN*, where *SN* stands for *Single Network*, meaning that a single network combiner, not an ensemble, is used to fuse the outputs of the experts.

According to the mentioned table, the results provided by *Simple Ensemble* can be improved by applying a combination ensemble. However, the results provided by a combination ensemble are quite similar to the results provided by a single combination network (*SE-SN*), specially when they are combining a few experts.

Fortunately, it can observed that the results provided by *Stacked* can slightly outperform *Simple Ensemble* without training a huge number of networks. For instance, an ensemble of 3 experts trained by *Simple Ensemble* which has been combined by an ensemble of 5 combination networks trained by *CVCv3ACB* provides a performance slightly better than the ensembles of 40 networks trained with *Simple Ensemble*. Moreover, the results provided by combining ensembles of 9 expert networks with *Stacked* clearly outperforms the ensembles of 40 networks combined by *Output average*. Furthermore, *SE-CVCv3ACB* with *Boosting Combiner* provides better results

than *SE-CVCv3ACB* with *Output average* for the low and medium sized ensembles (3 and 9 expert networks). And *SE-CVCv3Conserboost* with *Boosting Combiner* is better than *SE-CVCv3Conserboost* with *Output average* for the case of low sized ensembles (3 expert networks). In these last cases, it is also better to use a combination ensemble than a single network as combiner.

Secondly, table 9.6 shows the results of the *combination ensembles* designed with *Stacked+*, when they are applied to combine ensembles of expert networks previously trained with *Simple Ensemble*.

Table 9.6: *Stacked+* with ensembles of combination networks

		Mean *IoP*				**Mean** *PER*			
Method	**CN**	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Simple Ensemble	–	4.97	5.27	5.34	5.43	21.84	23.65	23.73	24.64
SE-SN	1	5.12	5.63	5.78	5.62	22.63	26.38	27.26	26.71
SE-SE	3	5.13	5.73	5.79	5.61	22.88	26.89	27.27	26.89
SE-SE	5	5.07	5.72	5.79	5.67	22.61	27.02	27.3	27.1
SE-SE	9	5.13	5.73	5.78	5.69	22.6	27.05	27.44	27.3
SE-CVCv3ACB a	3	5.52	5.84	5.81	5.92	25.71	26.55	27.5	27.49
SE-CVCv3ACB a	5	5.57	5.97	5.86	5.97	26.77	27.81	27.6	28.17
SE-CVCv3ACB a	9	5.60	5.92	5.89	5.90	27.36	27.71	27.14	27.70
SE-CVCv3ACB b	3	5.38	5.84	5.77	5.99	25.06	26.73	26.16	28.2
SE-CVCv3ACB b	5	5.53	5.99	5.84	6.09	26.11	27.69	27.31	28.82
SE-CVCv3ACB b	9	5.48	5.93	5.9	6.14	25.39	27.72	27.03	28.68
SE-CVCv3Conserboost a	3	5.38	5.88	5.8	5.85	23.62	27.38	26.85	26.47
SE-CVCv3Conserboost a	5	5.57	5.8	5.91	5.87	25.45	26.74	27.42	27.12
SE-CVCv3Conserboost a	9	5.65	5.96	5.71	5.78	27.53	27.24	25.66	26.3
SE-CVCv3Conserboost b	3	5.27	5.85	5.69	5.65	23.87	28.09	26.31	25.71
SE-CVCv3Conserboost b	5	5.56	5.9	5.95	5.73	25.69	27.17	28.04	25.79
SE-CVCv3Conserboost b	9	5.77	5.88	5.77	5.87	28.5	27.43	26.89	26.89

In general, the combiner *Stacked+* provides better results than the combiner *Stacked* when an ensemble is employed as a combiner model and the model of this ensemble is complex and of high performance as *Cross-Validated Boosting*. This improvement with respect to the combiner *Stacked* can be clearly seen when *Cross-Validated Boosting* methods are applied to generate the *combination ensembles*. With this procedure, the performance is improved but the complexity of the combination ensemble is also increased.

Moreover, the best results provided by a *Simple Ensemble* of 40 networks are clearly improved by this model by training only a few networks. Applying *Stacked+* to an ensemble of 9 networks (9 experts) with a combination ensemble of 5 networks (5 combiners) can provide a *mean IoP* close to 6% and a *mean PER* close to 28% instead of 5.4% and 24.6% of a *Simple Ensemble* of 40 networks. So, with only 14 networks we achieve better resutls than with 40 networks changing the design model. Higher values on these general measurements can be reached with more experts and combination networks as we can see that the best overall results, mean *IoP* equal to 6.14%, are obtained with a system (*SE-CVCv3ACB b*) with

40 experts and 9 combination networks according to the mean IoP. However, the highest mean PER, 28.82%,is obtained with the same system but the number of combination networks is 5. This behavior is due to the use of an heterogeneus set of classification problems and the origins of the mean PER (relative values) and IoP (absolute measurement). Both systems, with 5 and 9 combination networks, can be considered the best performing combiners for *Simple Ensemble* among all the alternatives shown in this chapter.

Under the point of view of the combiner employed in the combination ensemble, we have tested *Output average* (denoted by "*a*" in the table) and *Boosting Combiner* (denoted by "*b*" in the table) as before. In this case, the results are varied. On the one hand, for *SE-CVCv3ACB* the best combiner seems to be *Output average* for a wide number of expert networks ranging from 3 to 20, but in the case of 40 expert networks (a high number of experts) the most appropriate combiner seems to be *Boosting Combiner*. On the other hand, for *SE-CVCv3Conserboost* the results are the contrary, from 3 to 20 networks *Boosting Combiner* is the appropriate combiner and for 40 experts is *Output average.*

Finally, applying a combination ensemble provides better results than a single combination network when they are used to fuse the networks of a *Simple Ensemble*. This is the case of the systems with 3 and 9 expert networks in *SE-CVCv3Conserboost*, and also it is the case for every number of networks in *SE-CVCv3ACB* if we select the most useful combiner for the concrete number of expert networks.

9.5 Conclusions

Stacked Generalization is a model which was proposed by Wolpert in 1994. This model is composed by two or more levels of classifiers. The classifiers of the first level, the experts, process the original information whereas the classifiers of the second level, or combiners, process the information provided by the first level. This model is a special *Multiple Classifier System*, however there can be implemented as a whole model or the two levels can be generated independently which is the approach followed to propose the combiners *Stacked* and *Stacked+*. For this reason, the study has been split into three subsections.

The stacked idea has been applied in three different ways: According to the *original stacked models*, as an *ensemble combiner* formed by a single neural network in the two versions, *Stacked* and *Stacked+*, and as an *ensemble combiner* formed by a *combination ensemble* following again the two versions of combiners *Stacked* and *Stacked+*.

As was mentioned before, two authors adapted the original methods in order to apply it to neural networks and other classifiers as base classifiers. But only one version, the implementation proposed by *Ghorbani & Owrangh*, slightly outperformed *Simple Ensemble*. Their model with 9 experts and a single combination network is statistically better than a *Simple Ensemble* composed by 9 networks and it is slightly better than a *Simple Ensemble* composed by 40 networks. Moreover, we also saw

that this implementation was slightly worse than *Simple Ensemble* according to the mean PER when the number of expert was high (20 and 40 experts).

Moreover, the *Stacked* idea was also applied to propose new variants of ensemble combiners, *Stacked* and *Stacked+*. Firstly, *Stacked* and *Stacked+* with a unique *MF* network as combiner were applied to ensembles of *Multilayer Feedforward* networks and the results showed that, in general, the combiners proposed are good on low and medium sized ensembles (3 and 9 networks). Moreover, they are also good combiners for high sized ensembles (20 and 40 experts) in *Simple Ensemble* and *Inverboost*. In this case we have to remark that the results provided by the combination of *Inverboost* with *Stacked* and *Stacked+* are quite good because the same ensembles combined by *Boosting Combiner* performed quite worse than a single network. It seems that the proposed combiners have corrected the bad decisions taken by the original ensemble. However, the new combiners do not provide good results on the other boosting methods when the number of experts is high (20 and 40 networks).

Furthermore, *Stacked* and *Stacked+* have been used to combine ensembles of *RBF* networks (experts) with a single *MF* combination network. As previously suggested, some normalization procedures were applied to the outputs of the *RBF* networks in order to get the output values in the range $[0, \ldots, 1]$. The results showed that the new combiners fit well when they were applied to combine low and medium sized ensembles (3 and 9 networks) with the original and *threshold* normalized *RBF* networks. Moreover, the *sum* and *min-max* normalization procedures should not be applied along with the proposed *Stacked* and *Stacked+*.

Finally, the information provided by an ensemble of expert networks can be processed by another ensemble of neural networks, a *combination ensemble* in this case. The results show that good performance can be reached with the new stacked combiners with a few expert networks and a few combination networks. The best overall results of combining *Simple Ensemble* with *Stacked* are mean IoP equal to 5.92% and PER equal to 28.65 and the best results obtained by *Stacked+* are mean IoP equal to 6.14% and PER equal to 28.82%. In both cases, the best results were reached when *CVCv3ACB* was used to generate the ensemble of combination networks. In general, *Simple Ensemble* is improved by both new combiners when a combination ensemble is applied. Moreover, they provide better results than the combiners shown in chapter 6 for *Simple Ensemble*. According to the results shown here, a combination ensemble is a good way to combine ensembles of expert networks and it should be seriously considered.

It is important to conclude by remarking that a stacked system can be affected by the curse of dimensionality if it is composed by a high number of experts, i.e 40 experts, because the number of inputs of the combination network increases as the number of experts increases. In the combiners proposed, the input data of the combination networks are the outputs of the experts on the training set. For instance, 40 experts mean that the number of inputs of a combination network is 40 times the number of classes of the original problem (outputs of the experts). Maybe, the

number of patterns for training this kind of networks is not enough and we do not achieve good generalization in systems with a high number of experts. As further work, we will research in codifying better these inputs in order to reduce the curse of dimensionality in the combination networks.

The results of shown in this chapter have been published in references [AM1,AM2,AM3]. These references are described in the conclusions chapter.

Chapter 10

Mixture of Experts

10.1 Introduction

As has been pointed out in this thesis, there are some other important approaches to generate *Multiple Classifier Systems*. Here, *Mixture of Experts* will be introduced and modified. In this model, all the networks are trained simultaneously and one of them is used to weight the output of the other networks.

This chapter, is organized as follows. Firstly, the *Mixture of Experts* is described as a whole classification system in section 10.2. Moreover, some important network architectures are proposed to be used along with this important model. Furthermore, we also introduce a new way to combine ensembles based on this approach. Then, the experimental setup is introduced in section 10.4. Finally, the results and their discussion are shown in section 10.5.

10.2 Mixture of Experts

Mixture of Experts is one of the important approaches to generate *Multiple Classifier Systems* and it was proposed by Jacobs et al. in [94]. In the literature, it has been also named *Mixture of Neural Networks* or *Mixture of Expert Networks*.

The *Mixture of Experts* is a modular model to build a complex classification system which consists in training different neural networks, also called *expert networks* or simply *experts*, along with a *gating network*. Then, the problem is divided by the system into some subproblems and each new subproblem tends to be solved by one expert. The gating network is used to weight the outputs of the experts in order to calculate the final output of the whole classifier. Figure 10.1 shows a graphical description of this system.

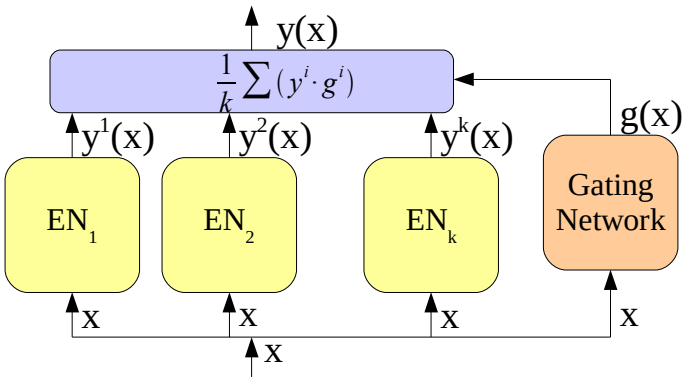


Figure 10.1: *Mixture of Experts* Structure

In the original references, the base network architecture used does not contain any hidden node as shown in figure 10.2. According to the literature, this *Basic Network* can only solve linear problems [24, 27]. The main idea of this model is that a classifier with a few networks plus the gating network can achieve similar results than more complex network architectures, *Multilayer Feedforward* (*MF*) or *Radial Basis Functions* (*RBF*). Moreover, the computational cost of the new model is lower because

the network architecture chosen is simpler. Furthermore, this process guarantees that the interference among the experts is reduced [120, 94].

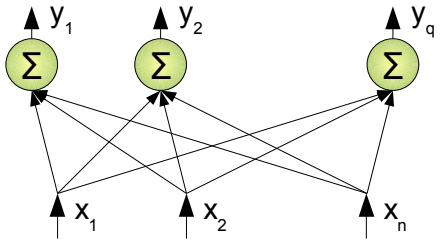


Figure 10.2: Basic Network Structure

The output vector of the experts, y^{net}, and gating network, g, are given by the following two equations.

$$y_{class}^{net} = x^T \cdot w_{class}^{net} \tag{10.1}$$

$$g_{net} = \frac{\exp\left(x^T \cdot a_{net}\right)}{\sum_{j=1}^{N_{experts}} \exp\left(x^T \cdot a_j\right)} \tag{10.2}$$

Where x^T is the transpose of the input data vector and w refers to the weights. Concretely, w_{class}^{net} denotes a vector whose elements are the weights between each input, x_i, and an output unit, $class$, of network net. The outputs of experts, y_{class}^{net}, are calculated with the scalar product of these two vectors. The outputs of the gating network, g_{net}, are calculated in a similar way but, in this case, the weights are denoted by a. The last two equations are similar but they are not identical. Although the same architecture is selected to generate all the networks of the system, the gating network requires a special normalization (the sum of all weights, g, is one) based on calculating the exponential function (exp) of the output neurons.

The description of the procedure used to design a classifier with *Mixture of Experts* is described in algorithm 10.1.

Algorithm 10.1 Mixture of Experts

```
Random initialization of networks
for ite = 1 to N_ite do
  for each pattern from training set do
    for net = 1 to N_experts do
      Adapt expert weights
    end for
    Adapt gating weights
  end for
  Calculate L_ite over Validation set and save temporally its weights
end for
Select iteration, ite, with highest L_ite and set its weight to the system.
```

In *Mixture of experts*, the whole classification system is trained for some iterations. In each iteration, all the patterns are iteratively presented to all the networks of the system (experts and gating) and the weights are adjusted. The weights of these networks are adapted by maximizing a sophisticated error function based on the *Mean Squared Error*, L in equation 10.3. Equations 10.4 and 10.5 are used to adjust the weights of the experts and gating network. Once all the patterns have been presented, the mean of the cost function of the iteration, L_{ite}, is calculated with all the patterns from the validation set, V, and the weights are temporally stored. When the procedure finishes, the system configuration is set with the weights of the iteration with highest L.

$$L = \log\left(\sum_{net=1}^{N_{experts}} g_{net} \cdot \exp\left(-\frac{1}{2} \cdot \left\|d - y^{net}\right\|^2\right)\right) \tag{10.3}$$

L is considered a sophisticated error function because the error of a network is weighted by the corresponding output of the gating network, g, in the cost function of the system. The error of an expert is only considered in L if this expert is in charge of solving this part of the problem according to the gating network. The error of each network is calculated with the exponent part, exp. This exponent is a vector norm which corresponds to the negative value of the *Mean Squared Error*, MSE, described in chapter 2 (equation 2.10). If the error is high, L is low because the exponent is also low. L inversely depends on the error of the networks. For this reason, L is maximized in this case. A high value of L denotes that the weighted error of all the networks is low.

The equations applied to update the weights of the experts, w, and gating network, a, are obtained with a gradient ascent algorithm. The final equations used to update each weight value of the experts and gating network are:

$$w_{i,class}^{net}(t+1) = w_{i,class}^{net}(t) + \eta \cdot hm^{net} \cdot \left(d_{class} - y_{class}^{net}\right) \cdot x_i \tag{10.4}$$

$$a_{i,net}(t+1) = a_{i,net}(t) + \eta \cdot \left(hm^{net} - g_{net}\right) \cdot x_i \tag{10.5}$$

Where η is the adaptation step which must be set by a trial and error procedure and hm^{net} is calculated with equation 10.6. This last variable, hm, is used to simplify the equations applied to adjust the weights. The variable t is used to denote the current value of the parameters (weight values) and $t+1$ refers to the new value of the same parameters after adjusting them.

$$hm^{net} = \frac{g_{net} \cdot \exp\left(-\frac{1}{2} \cdot \left\|d - y^{net}\right\|^2\right)}{\sum_{n=1}^{N_{nets}} g_n \cdot \exp\left(-\frac{1}{2} \cdot \left\|d - y^{n}\right\|^2\right)} \tag{10.6}$$

In this model, the original problem is split into some subproblems, each subproblem tends to be solved by one expert. Moreover, the boundaries between classes are given by hyperplanes when the basic network architecture is applied [24, 121]. In fact, each expert will provide an hyperplane to separate two classes and the gating network decides which hyperplane fits better for a determined pattern. A simple graphical description is shown in figure 10.3 where: (a) is a subspace of the input space which shows the real boundary between $class_1$ and $class_2$, (b) and (c) represent the boundary predicted by $expert_1$ and $expert_2$ respectively and (d) represents the boundary of the mixture system on the subspace when the gating network weights the experts.

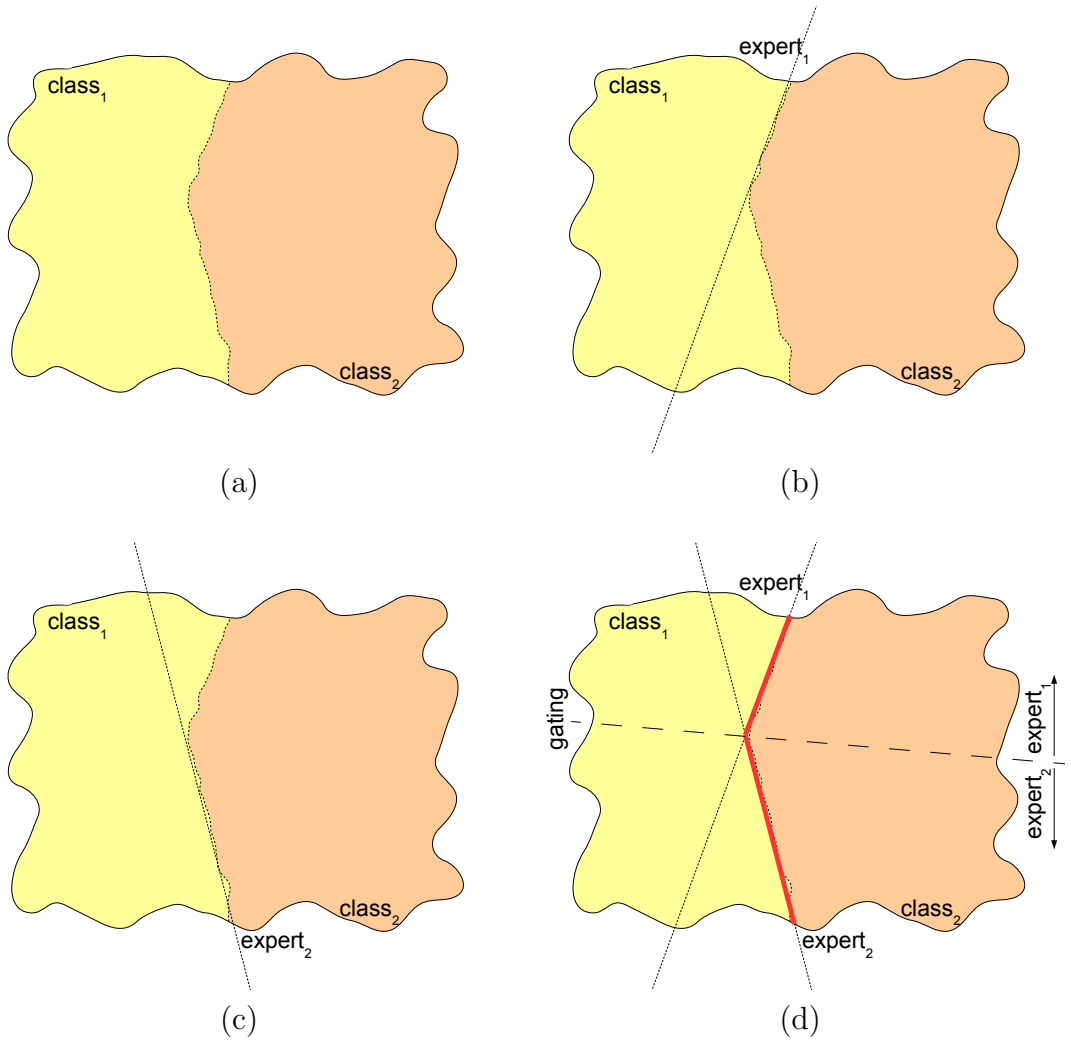


Figure 10.3: Mixture - Graphical representation

As it can be seen in the previous example, each expert represents a portion of the complex boundary between the two classes. In (b) the boundary predicted by $expert_1$ is similar to the top part of the real boundary but it is quite different to the bottom part of the real boundary. The opposite behavior occurs with the boundary predicted

by $expert_2$ in (c) because the inferior region of the real boundary is well represented by this expert but the superior part is not. In (d), a complexer boundary is generated from the two first predicted boundaries and the information of the gating network. This last boundary represents better the real boundary in any region.

Finally, *Mixture of Experts* is based on simple networks so the resources required are low. However, its performance should be similar to a single classifier based on more complex networks. Although this model was improved by applying other structures [122, 123, 124, 125], this thesis will be focused on the original version.

10.2.1 Applying other network architectures

In the original model, the simple *Basic network* (*BN*) is the basis of the *Mixture of Experts*. However, we propose to use other alternatives such as *Multilayer Feed-forward* (*MF*) or *Radial Basis Functions* (*RBF*). In this way, better base classifiers are applied to generate the experts and the gating network.

The basic algorithm of *Mixture of Experts* is not modified when a new network architecture is introduced. However, the *parameters* of the experts and gating network will be optimized according to the network chosen.

10.2.1.1 Mixture and the *MF* network

In the case of *MF* networks, we have used this network architecture to generate the experts and the gating network. The outputs of each expert, y^{net}, are calculated as a single *MF* network (equation 2.8). This equation is reproduced and simplified here in equation 10.7. In the equations described in this subsection, *wih* represents the weights between the input and hidden layers of a network and *who* denotes the weights between the hidden and output neurons as we did in previous chapters. The supperindex *net* denotes the number of the expert network but *g* refers to the gating network.

$$y_{class}^{net} = \phi_{sig}\left(\sum_{h=1}^{Nhidden} who_{h,class}^{net} \cdot ho_h^{net}\right) \tag{10.7}$$

Where ϕ_{sig} is the sigmoid transfer function, equation 2.1, and *ho* is given by:

$$ho_h^{net} = \phi_{sig}\left(\sum_{i=1}^{N_{input}} wih_{i,h}^{net} \cdot x_i\right) \tag{10.8}$$

However, the outputs of the gating network, g, are given by the following equation:

$$g_{net} = \frac{\exp\left(y_{net}^g\right)}{\sum_{n=1}^{N_{nets}} \exp\left(y_n^g\right)} \tag{10.9}$$

Where:

$$y_{net}^{g} = \phi_{id}\left(\sum_{h=1}^{Nhidden} who_{h,net}^{g} \cdot ho_{h}^{g}\right) = \sum_{h=1}^{Nhidden} who_{h,net}^{g} \cdot ho_{h}^{g} \tag{10.10}$$

$$ho_{h}^{g} = \phi_{sig}\left(\sum_{i=1}^{N_{input}} wih_{i,h}^{g} \cdot x_i\right) \tag{10.11}$$

The structure chosen for the gating network slightly differs from the original *MF* network because the identity transfer function (ϕ_{id} is described in equation 2.2) is applied to calculate the output units, y_{net}^{g}, in equation 10.10. Then, the output vector, y^{g}, is normalized in 10.9 to obtain the final values of the gating network, g_{net}. In a normal *MF* network, the sigmoid transfer function is used to calculate the output units as done, for instance, in equation 10.7 to calculate the output of the experts. We had to adapt the *MF* network because the gating network worked as the combiner *Output average* with the original equations. The outputs of the *MF* network are ranged between 0 and 1 and, maybe, this range is not appropriate for a gating network. Moreover, there is not any restriction related to the summatory of all the elements of the output vector in the traditional *MF* network. Maybe, the weights provided by the gating network may be relative (their summatory should be one) and not absolute. For this reason, the output layer was modified and, now, it is "similar" to the *Basic Network*.

Finally, the weights of the experts, wih^{net} and who^{net}, and gating network, wih^{g} and who^{g}, are adjusted in the training procedure with the following equations where the superindex *net* denotes the number of the expert and g is the gating network:

$$wih_{i,h}^{net}(t+1) = wih_{i,h}^{net}(t) + \eta \cdot hm^{net} \cdot H_{h}^{net} \cdot \sum_{c=1}^{N_{classes}} \left(\delta_{c}^{net} \cdot who_{h,c}^{net}\right) \cdot x_i \tag{10.12}$$

$$who_{h,class}^{net}(t+1) = who_{h,class}^{net}(t) + \eta \cdot hm^{net} \cdot \delta_{class}^{net} \cdot ho_{h}^{net} \tag{10.13}$$

$$wih_{i,h}^{g}(t+1) = wih_{i,h}^{g}(t) + \eta \cdot \sum_{n=1}^{N_{nets}} \left((hm^{n} - g_n) \cdot \rho_{h,n} \cdot x_i\right) \tag{10.14}$$

$$who_{h,net}^{g}(t+1) = who_{h,net}^{g}(t) + \eta \cdot \left(hm^{net} - g_{net}\right) \cdot ho_{h}^{g} \tag{10.15}$$

In the previous equations, η is the adaptation step, hm and ho are given by equations 10.6 and 2.6. Finally the other variables are calculated as follows:

$$H_{h}^{net} = ho_{h}^{net} \cdot (1 - ho_{h}^{net}) \tag{10.16}$$

$$\delta_{class}^{net} = (d_{class} - y_{class}^{net}) \cdot (1 - y_{class}^{net}) \cdot (y_{class}^{net}) \tag{10.17}$$

$$\rho_{h,net} = ho_{h}^{g} \cdot (1 - ho_{h}^{g}) \cdot who_{h,net}^{g} \tag{10.18}$$

The three last equations have been introduced to simplify the equations used to update the weights of the experts and gating network (equations 10.12 to 10.15).

10.2.1.2 Mixture and the *RBF* network

In the case of *RBF* networks, the outputs of the experts, y^{net}, and the outputs of the gating network, g, are calculated with equation 10.19 and 10.20.

$$y^{net}_{class} = \sum_{ct=1}^{N_{clusters}} w^{net}_{ct,class} \cdot h^{net}_{ct} \tag{10.19}$$

$$g_{net} = \sum_{ct=1}^{N_{clusters}} w^{g}_{ct,net} \cdot h^{g}_{ct} \tag{10.20}$$

Where the subindex ct denotes the number of the clusters or hidden nodes of the network and w refers to the weights between the hidden units (clusters) and the output neurons of the *RBF* network. Moreover, h^{net}_c and h^g_c corresponds to the Gaussian function (*Exponential generator Function*) and they are calculated as:

$$h^{net}_{ct} = \exp\left(-\frac{\|x - v^{net}_{ct}\|^2}{(\sigma^{net}_{ct})^2}\right) \tag{10.21}$$

$$h^{g}_{ct} = \exp\left(-\frac{\|x - v^{g}_{ct}\|^2}{(\sigma^{g}_{ct})^2}\right) \tag{10.22}$$

These previous equations are based on the output provided by the original *RBF* network (equation 2.16). Furthermore, σ is the width of the gaussian function and it is set by a trial an error procedure and v denotes the center of the *RBF* networks.

In this case, the *RBF* network architecture should not have to be adapted, as we did in *MF*, when it is applied as gating network.

Therefore the centers and weights of the expert networks, v^{net} and w^{net}, and gating network, v^g and w^g, are adjusted in the training procedure with the following four equations.

$$w^{net}_{ct,class}\,(t+1) = w^{net}_{ct,class}\,(t) + \eta \cdot hm^{net} \cdot (d_{class} - y^{net}_{class}) \cdot h^{net}_{ct} \tag{10.23}$$

$$v^{net}_{i,ct}\,(t+1) = v^{net}_{i,ct}\,(t) + \eta \cdot hm^{net} \cdot \alpha^{net}_{i,ct} \cdot \sum_{c=1}^{N_{classes}} \left((d_c - y^{net}_c) \cdot w^{net}_{ct,c}\right) \tag{10.24}$$

$$w^{g}_{ct,net}(t+1) = w^{g}_{ct,net}(t) + \eta \cdot \left(hm^{net} - g_{net}\right) \cdot h^{g}_{ct} \tag{10.25}$$

$$v^{g}_{i,ct}(t+1) = v^{g}_{i,ct}(t) + \eta \cdot \alpha^{g}_{i,ct} \cdot \sum_{n=1}^{N_{nets}} \left((hm^{n} - g_n) \cdot w^{g}_{ct,r}\right) \tag{10.26}$$

Where α is given by:

$$\alpha^{net}_{i,ct} = \frac{2 \cdot h^{net}_{ct}}{(\sigma^{net}_{ct})^2} \cdot \left(x - v^{net}_{i,ct}\right) \tag{10.27}$$

$$\alpha^{g}_{i,ct} = \frac{2 \cdot h^{g}_{ct}}{(\sigma^{g}_{ct})^2} \cdot \left(x - v^{g}_{i,ct}\right) \tag{10.28}$$

This network structure was successfully applied as experts in [126], the performance of the modified model was considerably higher than the original version.

10.3 Mixture as an ensemble combiner

Although the *Mixture model* corresponds to a modular classifier, it can also be related to ensembles of neural networks. This model was adapted to improve the generalization ability of *Optimal Linear Combinators* in which a weighted average was applied [127]. For this reason, we propose to apply a gating network to weight the outputs provided by the networks of an ensemble to reduce the error of the whole classifier. We think that the gating network can learn which expert, network of the ensemble, fits better for a determined pattern. In this way, the gating network can switch the control to the most appropriate networks of the ensemble for a determined pattern, x.

In the *Mixture combiner* proposed, the ensemble is set as expert networks and then the gating network is trained while the experts are kept unchanged. This combiner can be seen as a sophisticated version of *Weighted Average* whose success will depend on the gating network. The description of this combiner is given by algorithm 10.2 whereas a graphical diagram of the training is given by figure 10.4.

Algorithm 10.2 Ensemble combiner based on Mixture of Experts

```
Random initialization of gating network
Generate Ensemble or Select previously trained ensemble
Set networks of the ensemble as experts
for ite = 1 to iterations do
   for each pattern from training set do
      Adapt gating network parameters
   end for
   Calculate L_ite over Validation set
   Save gating network weights
end for
Select iteration with highest L (best iteration)
Set best iteration parameters to gating network and save final configuration
```

In this new combiner, the ensemble is used as a set of expert networks. In this point a new ensemble can be trained or a previously generated ensemble can be used. Then, the gating network is trained for some iterations. In each iteration, all the patterns from the training set are presented to the network. The trainable parameters of the gating network, usually its weights, are adapted with the same equations introduced in the previous section. Depending on the network architecture chosen for the gating network, the corresponding equations will be used to adjust the network parameters. It is important to mention that the trainable parameters of the experts are kept unchanged during this training procedure because we are introducing an ensemble combiner. At the end of each iteration, the mean of the

target function, L, is calculated with all the patterns from the validation set and the parameters of the gating network are temporarily stored. Finally, the configuration of the best iteration, the one with highest L, is set to the final gating network. With this procedure, we set the most suitable configuration of the gating network avoiding overfitting.

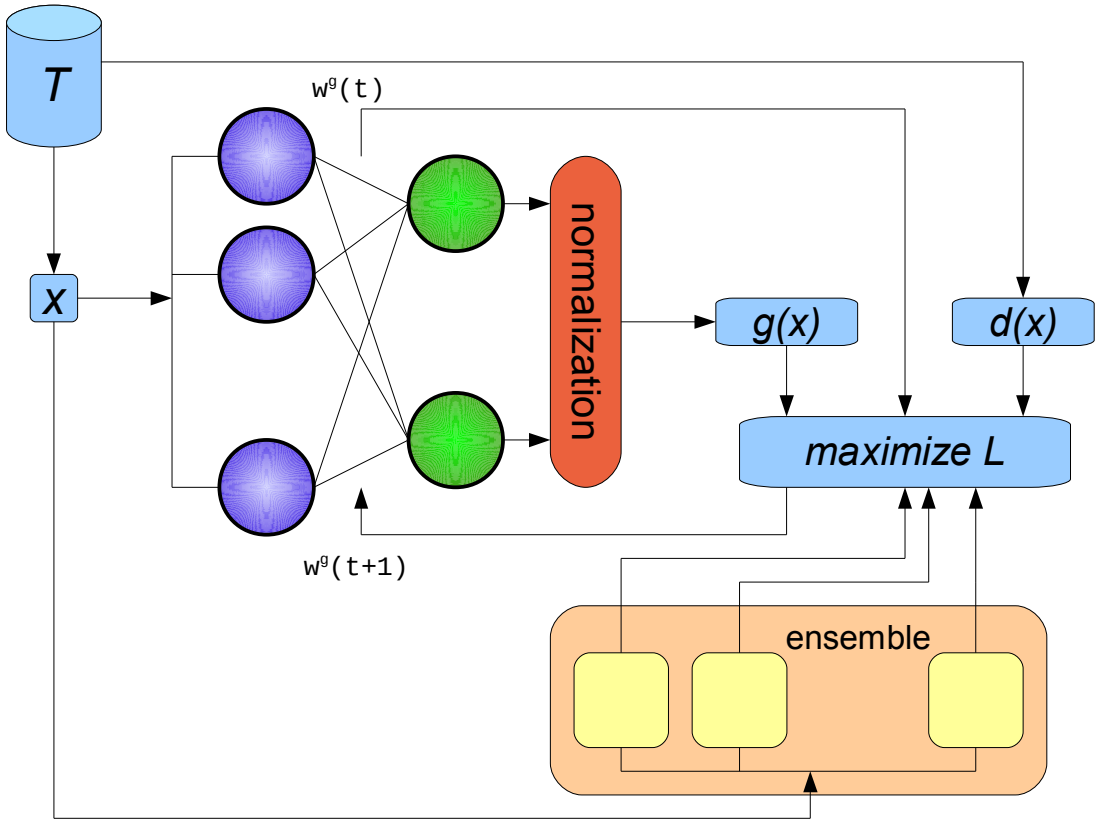


Figure 10.4: Example of the training algorithm of Mixture as an ensemble combiner

In algorithm 10.2 and figure 10.4, it can be noticed that the proposed *Mixture combiner* differs from the two *Stacked combiners* introduced in the previous chapter. Firstly, a quasi *non-supervised* learning algorithm is applied to train the gating network whereas the combination networks are generated by a *supervised* algorithm. The desired output of the gating network are not previously defined but the targets of the combination network in *Stacked* are the real class (target) of the pattern. Secondly, the original pattern input is used to train the gating network but the combination network of *Stacked combiners* mainly processes the output of the experts. Finally, the outputs of the gating network are the weights associated to the experts in a special average, in contrast, the combination network of the proposed *Stacked* and *Stacked+* provides a final prediction for each pattern. Although the combiners we proposed in the last and current chapter are based on the use of a neural network, they differ on the way they are designed and in their final purpose.

10.4 Experimental Setup

In this chapter the *Mixture of Neural Networks* model, has been analyzed and modified. In all the experiments, we have used a single gating network according to the original references.

We have to take into consideration that the original implementation was an alternative to more complex networks so its aim was not to provide a high increase in performance with respect to the other advanced classifiers.

The experimental setup applied is similar to the experiments carried out for the ensemble methods. In this case, the number of experts corresponds to the number of networks trained in an ensemble. With this procedure the classification systems generated can be compared to the single *MF* network and to *Simple Ensemble*. The main characteristics of the experiments done in this chapter are:

- Nineteen classification problems from the *UCI Repository*.
- Three implementations of the original model with two network architectures *Basic Network* and *Multilayer Feedforward*.
    - Multi-net systems with 3, 9, 20 and 40 experts.
    - 1 unique gating network.
- Two implementations of a new ensemble combiner with two network architectures *Basic Network* and *Multilayer Feedforward*.
    - Ensembles of 3, 9, 20 and 40 networks trained with *Simple Ensemble* as experts.
    - 1 unique gating network.
- Optimized training parameters.
- The experiments have been repeated ten times with different partitions of training, validation and test sets in order to obtain:
    - Mean value of performance.
    - Error rate by standard error theory.
- Two general measurements applied to the comparison.
    - Mean *Increase of Performance*.
    - Mean *Percentage of Error Reduction*.

The description of the nineteen datasets used in the experiments can be found in appendix A. The training parameters of the experts and gating network are in appendix B.8. Finally, the general measurements applied to compare the methods and combiners studied are the mean *Increase of Performance* (equation 3.5) and the mean *Percentage of Error Reduction* (equation 3.6). Furthermore, the complete results will also be considered to compare all the alternatives.

10.5 Results and discussion

The results related to the *Mixture of Neural Networks* are shown in this section. As previously mentioned, the *Mixture* approach can be applied in different ways. For this reason, this section has been split into two different subsections. The first one will be focused on the original *Mixture* model whereas the results provided by combining ensembles with a gating network are shown in the second part.

10.5.1 Original Mixture models

In this subsection, three different implementations of the original *Mixture* model have been considered. They are based on the *Basic Network* (*BN*) and the *Multilayer Feedforward* network (*MF*). Concretely, these methods are:

- *Mix-BN-BN*: The experts and the gating network are designed with the simple architecture *BN*.
- *Mix-BN-MF*: The architecture *BN* is chosen to generate the experts but the gating network is generated with a *MF* network.
- *Mix-MF-MF*: The architecture *MF* is selected to create the experts and the gating network.

The results of these three alternatives along with the results of *Simple Ensemble* are shown in the following table. The three new classifiers are tested with the mean IoP and mean PER with respect to a single *MF* network as we did to evaluate the different ensemble alternatives. To compare easier the behavior of *Mixture* with respect to ensembles, we have generated systems with 3, 9, 20 and 40 experts. In this way, we can compare *Simple Ensemble* of 9 networks and *Mix-MF-MF* composed by 9 *MF* experts.

Table 10.1: Mixture of Neural Networks

	Mean IoP				**Mean PER**			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Simple Ensemble	4.97	5.27	5.34	5.43	21.84	23.65	23.73	24.64
Mix-BN-BN	−5.25	−0.94	0.53	0.76	−64.40	−21.83	−11.90	−13.40
Mix-MF-BN	4.08	4.32	4.40	4.36	17.13	18.98	18.63	18.36
Mix-MF-MF	5.06	4.62	5.03	4.88	21.23	18.69	21.11	19.20

At first sight, *Mix-MF-MF* is the alternative which provides the best overall results among all the three implementations based on the *Mixture model*. The *MF* network is much better than the simple *BN* so the original *Mixture model* is improved by the new implementation we propose with *MF* networks. Although *Mix-MF-BN* also provides good results, it is slightly worse than *Mix-MF-MF*. Finally, *Mix-BN-BN* provides quite worse results than the other two mixture versions and it is worse than a *Single network* in most of the cases.

Moreover, it can also be noticed that the two new versions we propose, *Mix-MF-BN* and *Mix-MF-MF*, are overcome by *Simple Ensemble*. However, the *ensemble*

model and the *Mixture model* are two different approaches and they highly differ on classifying patterns so both models should be seriously considered.

Furthermore, the purpose of the original mixture method, *Mix-BN-BN*, is to obtain similar results than a more complex classification system. It means, that a mixture of simple networks should provide similar performance than a simple *MF* network but the computational requirements should be lower. Moreover, the procedures to set the parameters and train the system are much faster for *Mix-BN-BN* than for a single *MF* network. According to the mean IoP, *Mix-BN-BN* can provide better results than a single network when the number of experts is 20 and 40 because the mean IoP is positive. Although *Mix-BN-BN* provides worse results than a single *MF* networks in the other two cases, the absolute difference can be considered low for 9 networks because it is lower than 1%. Despite the mean IoP is positive in two cases, there are a few datasets in which *Mix-BN-BN* performs quite worse than a single *MF* network according to the general results shown in appendix C for 20 and 40 experts (table C.155). Moreover, this behavior can also be seen in table 10.1 because the mean PER is negative but the mean IoP is positive for the systems with 20 and 40 experts. This means than the classifier performs well for some datasets but there is another set of classification problems in which the performance is not good.

To illustrate this behavior, the following resume table shows an extract of the complete results of a single *MF* network, an ensemble of 40 *MF* networks generated with *Simple Ensemble* and *Mix-BN-BN* with 40 experts and a single gating network. In the table, the performance of the three classifiers is shown as the percentage of correctly classified patterns in the test set. This performance is shown for eight representative classification problems. The complete results for all classification problems can be seen in appendix C.

Table 10.2: Extract of the complete results

Database	MF net	SE	Mix-BN-BN
aritm	75.6 ± 0.7	73.8 ± 1.1	73.22 ± 1.04
band	72.4 ± 1.0	73.8 ± 1.3	75.45 ± 1.25
bupa	58.3 ± 0.6	72.7 ± 1.1	72.14 ± 1.65
ecoli	84.4 ± 0.7	86.9 ± 0.7	74.71 ± 1.43
flare	82.1 ± 0.3	81.6 ± 0.5	81.81 ± 0.58
mok2	65.9 ± 0.5	91.1 ± 1.2	68.38 ± 2.31
vote	95.0 ± 0.4	95.6 ± 0.5	96.50 ± 0.67
vowel	83.4 ± 0.6	92.2 ± 0.7	77.58 ± 0.73

In the previous table, it can be observed that the results provided by the three approaches are not correlated. Firstly, there are a few cases in which a single *MF* network provides the best results, datasets *aritm* and *flare*. Secondly, *Simple Ensemble* (*SE* in the table) provides similar o better results to the single *MF* network and it is clearly the best performing classifier for *mok2* and *vowel*. Thirdly, *Mix-BN-BN* provides similar results to the single network in general but in a few datasets this original model can highly improve or worsen the single network. For instance,

this model provides the worst overall results for datasets *ecoli* and *vowel* but it is the best classifier for *band* and *vote*. *Mix-BN-BN* highly depends on the dataset and case so, maybe, the low and negative general results are given by this fact. We consider that the performance of the worst cases, databases *ecoli* and *vowel*, can penalize the general behavior of *Mix-BN-BN*.

A similar analysis can be derived from the results provided by the new two *mixture methods*, *Mix-MF-BN* and *Mix-MF-MF*. The new mixture implementations provide the best results only on some databases whereas there is another set of classification problems in which the performance is slightly lower. However, the results are not as bad as in the original implementation because the base classifiers (*MF* networks) are better. Now, the experts can predict better the boundaries between classes because they are not limited by simple hyperplanes. In any case, the general performance of *Mixture of Experts* has been improved with the *MF* network at the expense of the complexity and computational cost in these new methods proposed by us.

As mentioned above, ensembles and mixture are quite different. The first one is composed by similar networks which try to solve a problem whereas the second one is composed by a set of classifiers which are specialists in solving a part of the problem. With this appreciation, mixture methods can be successfully applied to the classification problems which are susceptible of being divided into subproblems. Maybe, not all the datasets have this characteristic so this model will only fit well on the classification problems which have this feature.

Finally, *Mix-BN-BN* should only be seriously considered if it provides similar or better results than a *Single MF Network* or *Simple Ensemble*. It should be dismissed in the cases in which it is considerably worse than the other two alternatives. The training procedure of the original mixture method is quite faster if compared to the single network and ensembles and it is an important advantage. Moreover, this model can classify the patterns in a different way if compared to other advanced networks or ensembles so in further researches this feature can be interesting.

10.5.2 Mixture as an ensemble combiner

The *Mixture* model can also be applied to combine ensembles of neural networks as we proposed in subsection 10.3. In this case, the networks of the ensemble are used as experts and they are keep unchanged during the training. Only the weights of the gating network are adapted in order to maximize the cost function L.

It is important to mention that the combination network of the *Stacked combiners* highly differs from the *gating network* of the *Mixture combiner*. The first ones classify the patterns with the information provided by the networks whereas the second one provide the weights of a special average.

In the experiments, a single gating network will be used to determine which networks provide the most appropriate outputs. The ensembles previously trained with *Simple Ensemble* will be applied as expert networks. Moreover, the *BN* and *MF* networks will be used in the experiments as a gating network.

Concretely, the two new combiners based on the *Mixture model* we propose are:

- *Mix-SE-BN*: An ensemble trained with *Simple Ensemble* (*SE*) is used as experts and a single *Basic Network* (*BN*) is applied as gating network.
- *Mix-SE-MF*: Here the *Simple Ensemble* is combined with a single *MF* network as gating network.

The results of *Simple Ensemble* along with the results provided by the *Mixture combiners*, *Mix-SE-BN* and *Mix-SE-MF*, are shown in table 10.3. As in the previous subsection, the mean IoP and the mean PER with respect to a single *MF* network are used to compare all the alternatives.

Table 10.3: Mixture as ensemble combiner

	Mean IoP				**Mean** PER			
Method	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
Simple Ensemble	4.97	5.27	5.34	5.43	21.84	23.65	23.73	24.64
Mix-SE-BN	4.76	5.39	5.42	5.42	20.89	24.22	24.68	24.92
Mix-SE-MF	4.90	5.43	5.31	5.40	21.74	24.11	23.71	24.70

According to the mean PER, it can be observed that both *Mixture combiners*, *Mix-SE-BN* and *Mix-SE-MF*, slightly improve the performance of the ensembles previously trained with *Simple Ensemble* for the cases of medium and high sized ensembles, 9 to 40 experts, except for the case of *Mix-SE-MF* and 20 networks. However, the mean IoP shows that the *Output average* is slightly better than the new combiners we have proposed for ensembles of 40 networks.

Although the *MF* network performs better than the *BN*, the combiner *Mix-SE-BN* provides better mean IoP and PER than *Mix-SE-MF* when medium and high sized ensembles are combined, 20 to 40 networks in the ensemble. Moreover, the procedure to set the parameters of the gating network is faster and less complex for *BN*. For these reasons, the *Mix-SE-BN* combiner should be considered instead of *Mix-SE-MF*.

Finally, the two *Stacked combiners*, *Stacked* and *Stacked+* described in the previous chapter, perform also better than the two *Mixture combiners* proposed here. According to the results shown in chapter 9 (table 9.3), *Stacked* and *Stacked+* provided a mean PER higher than 23% for the case of low sized ensembles (3 networks) generated by *Simple Ensemble*. In the other cases (ensembles of 9, 20 and 40 networks), the mean PER was higher than 26% for both *Stacked combiners* and the highest value with *Simple Ensemble* was 27.52% (*Stacked* and ensembles of 40 networks). However, the way in which the two models are applied is quite different so the four alternatives should be seriously considered. As we concluded in chapter 6, a punctual high improvement may be obtained by applying the most appropriate combiner, even those which do not provide good general results.

10.6 Conclusions

The *Mixture of experts* is a model which was proposed by Jordan in 1991 and it has been successfully applied to neural networks. In this model, a set of expert networks are trained along with a gating network to solve a problem. Each expert tends to specialize on solving a part of the problem whereas the gating network is in charge of selecting which expert fits better on classifying a pattern. Originally, the *Basic Network* was the architecture chosen to be applied to this model. The main idea of the model is that a set of quite simple networks can perform as advanced network architectures, such as the *Multilayer Feedforward* network, with lower training requirements. In this research we want to evaluate this alternative to generate *MCS* and propose new implementations based on better network architectures in order to generate good classifiers.

This model is a special *Multiple Classifier System* in which all the networks are trained simultaneously. However there can be also used as an ensemble combiner. For this reason, this research has been split into two sections.

Firstly, the whole model has been implemented as described in the literature. In the original version, the *BN* has been used as expert and gating networks. Moreover, the *MF* network has also been successfully applied to the *Mixture model* in the experiments performed as we have proposed. The results provided by the original mixture method, *Mix-BN-BN*, the single *MF* network and the *Simple Ensemble* of *MF* networks are not correlated. The best results are provided by *Mix-BN-BN* in some datasets whereas in other datasets the best performance is provided by a single *MF* network or *Simple Ensemble*. Although the performance of *Mix-BN-BN* depends on the dataset, in general, the low performance was expected. This model was proposed as a simple alternative to more complex network architectures such as *MF* and *RBF*. In the majority of datasets, its performance is similar to the performance of the single *MF* network because *Mix-BN-BN* was not an alternative to ensembles. Furthermore, *Simple Ensemble* has also the behavior expected, it performs better than the single *MF* network and *Mix-BN-BN*.

The *Mixture* model provides good results if advanced network architectures are used as experts and gating networks according to the results shown here and in reference [126]. *Mix-MF-BN* provides better results than a *Single MF network* and the original implementation *Mix-BN-BN*. In this first alternative we proposed, the experts are better because the *MF* network can approximate any function with a defined precision [27, 28]. The second version we proposed, *Mix-MF-MF* also provided good results and it is the best mixture alternative. Although *Simple Ensemble* is better that the two implementations we proposed, there can be specific datasets in which the mixture models can be better than this ensemble.

Secondly, we proposed to use the *Mixture model* as ensemble combiner. The networks of the ensemble have been set as the expert networks and then their weights have been kept unchanged during the training of gating network. The results show that

the application of a gating network can slightly improve the performance of an ensemble in general. This improvement also depends on the dataset, being higher in some datasets than in other ones because the two general measurements do not completely agree.

It is important to conclude the chapter by remarking that the *Mixture model* is a special *Multiple Classifier System* which has to be carefully applied. It can improve the results provided by a more complex network architecture and reduce the computational cost and other training requirements. The way in which the model is trained differs from other approaches, such as the ensemble approach, so they should not fail on classifying the same patterns. The classifiers and combiners generated by the *Mixture model* should be seriously considered even if their performance is similar or a little bit worse if compared to a single *MF* network or ensembles. However, there can be some classification problems or datasets in which the results provided by the mixture methods and combiners can be considerably low so the *mixture methods* should be dismissed for these *conflictive* classification problems. This behavior can be seen if the two general measurements are compared between them. Similar values of the mean IoP do not provide similar values of the mean PER and vice versa. In this way, a model or classifier can be better than another according to one measurement but it can be worse according to the other one. Furthermore, there are cases in which a general measurement is positive and the other one is negative.

Finally, applying more complex network structures as we propose will improve the accuracy of the system as we saw for *Mix-MF-BN* and *Mix-MF-MF*. On the one hand, the experts based on *MF* networks represent better the boundaries in any region. On the other hand, the gating network assigns the weights in a effective way because the influence regions of the experts are not bounded by simple hyperplanes. However, the resources required by the mixture model based on complex networks are considerably higher if they are compared to the original model with the *Basic Network*.

The results of the research done in this chapter has been published in [AM4,AM5,AM6]. These references are detailed in the last chapter of this thesis.

Chapter 11

Conclusions and future work

11.1 Summary of the Thesis

This thesis has been focused on the analysis and development of ensembles of neural networks with the purpose of solving classification problems. This kind of *Multiple Classifier Systems* has been introduced to replace the classifiers based on a single neural network because it has been demonstrated that the generalization capability of an ensemble is high when the individual networks that compose it are not correlated and they do not commit the same errors [72, 48]. In this thesis, the most important alternatives to generate these different networks are reviewed and new variants are also proposed. Moreover, any ensemble should provide a single global output or a final hypothesis (predicted class). For this reason, selecting the most appropriate way to combine the networks of an ensemble is also an important key when the final ensemble is being designed. Some different ways to fuse the networks have also been analyzed and new combiners have been successfully introduced. Furthermore, other complex approaches to solve classification systems based on *Multiple Classifier Systems*, such as *Stacked Generalization* and *Mixture of Experts*, have also been considered and some experiments have been carried out. Finally, we introduce a detailed summary of the research performed in this thesis.

First of all, two important network architectures, the *Multilayer Feedforward* (*MF*) network and the *Radial Basis Functions* (*RBF*) network, were analyzed. Moreover, they were tested using a heterogeneous set of classification problems. Furthermore, their performance was also compared to the statistical classifier *K-Nearest Neighbors* (*KNN*) because it is an alternative classifier commonly used in the literature. A first research showed that *KNN* could not outperform the two network architectures. Moreover, those results showed that the two network architectures should be seriously considered for classification tasks. On the one hand, the *RBF* network provides the highest general performance on some classification problems. On the other hand, the *MF* network also provides good general results and the procedures applied to set the optimal training parameters and to train the final networks is faster if compared to *RBF* networks trained by supervised gradient descent.

Secondly, the *MF* and *RBF* networks were used to generate basic ensembles and their performance was tested with *Simple Ensemble*. The general results showed that the ensemble approach fits better when the *MF* network is used to generate the individual networks. On the one hand, the ensembles of *MF* networks clearly outperform the results provided by the single *MF* network. Probably, the different *MF* networks of the ensemble have similar performance but they fall into different *local optima* so the global classifier is much better than any of the single networks. On the other hand, the single *RBF* networks and the ensembles of *RBF* networks have similar performance according to the experiments introduced in chapter 3. The reason may be that the different *RBF* networks of an ensemble are quite similar among them so the global output is similar to the output provided by any single network. In other words, the ensembles of *RBF* networks are not suggested because the networks are highly correlated and they classify nearly in the same way.

Thirdly, an exhaustive comparison of ensemble methods was performed in chapters 4 and 5. At this point, only the *MF* network was chosen to be used as base classifier in the ensemble comparison as suggested in the first research done with *Simple Ensemble*. In order to make this comparison more comprehensible, it was divided into two chapters of the thesis. The ensembles were divided into two main groups depending on how the diversity, or dissimilarity among the networks, was generated. These groups were: *Ensembles which modify the training algorithm* and *Ensembles which modify the learning set*. Some different alternatives from the literature were applied in this global comparison. The most important ensemble methods, according the general results shown in this thesis, are:

- *Simple Ensemble*.
- *Averaged Boosting* (*Aveboost*).
- *Bootstrap Aggregating* (*Bagging*).
- *Cross-Validation Committee version 2* (*CVCv2*).
- *Cross-Validation Committee version 2.5* (*CVCv2.5*).
- *Cross-Validation Committee version 3* (*CVCv3*).
- *Conservative Boosting* (*Conserboost*).
- *Decorrelated version 1* (*DECOv1*).
- *Decorrelated version 2* (*DECOv2*).

Although there are some important alternatives in order to generate diversity on ensembles, the results showed that the modification of the learning set can be considered one of the most important sources of diversity because the best overall results are provided by the ensembles based on *Bagging*, *Boosting* and *Cross-Validation Committee*. Moreover, both versions of *Decorrelated*, in which the target equation used to update the weights of the network is altered to penalize correlated networks, are also a good alternative. Finally, *Simple Ensemble* has been included in the list because it provides good results if compared to the single *MF* network and it is the simplest way to generate ensembles. We consider that this basic ensemble should be the first alternative in order to test if the ensemble model is appropriate for a concrete classification problem or final application.

There have been some important ensembles, such as *Adaptive Boosting* (*Adaboost*) or *Evolutionary Ensemble with Negative Correlation Learning* (*EENCL*), that have been dismissed because they did not outperform *Simple Ensemble*. Other alternatives, i.e. *Cooperative Ensemble Learning System* (*CELS*), are not included in the list because they are only slightly better than *Simple Ensemble*. The ensemble methods included in the previous list are only those alternatives which have provided statistically better results than *Simple Ensemble*. Finally, there are a few methods, i.e. *Totally Corrective Adaboost* (*TCA*), *Ensembles Voting On-Line* (*EVOL*) or *Observational Learning Algorithm* (*OLA*), whose results were not expected because they

perform worse than a single *MF* network (their mean IoP and PER values were negative).

Among the three versions of *Cross-Validation Committee* shown in the list, we consider that the best choice is *Cross-Validation Committee version 3* because it reports the best general results and its performance is not negatively affected by the size of the ensemble. As we mentioned in chapters 5 and 8 the ensembles based on *CVCv2* are not suitable when the system is composed by a large number of networks (i.e, ensembles of 40 networks) because the partition used depend on the number of networks, but the ensembles based on *CVCv3* do not have this problem and they provide better performance. Each new version of *Cross-Validation Committee* can be seen as an improvement of the previous version so the last version can be considered the best choice. Furthermore, it is important to remark that *CVCv3* has been proposed by us. In this way, we consider that this new ensemble, *CVCv3*, which provides the best overall results of this ensemble comparison (its mean IoP is 6.17% and its mean PER is 28.68%), is a good contribution that can be used by any researcher to solve their classification problems.

Fourthly, an exhaustive comparison of combiners was done in chapter 6 using ensembles of *MF* networks previously generated by *Conservative Boosting*, *Cross-Validation Committee version 2*, *Decorrelated version 1* and *Simple Ensemble*. Moreover, the comparison was also done with ensembles of *RBF* networks previously trained with *Simple Ensemble*. In the case of *RBF* networks, we had to use three normalization procedures because the outputs were not in the range $[0, \ldots, 1]$. The results showed that *Output average* is the first combiner to be seriously considered because it provides excellent general results and it is the most simple combiner. Its general performance is high for ensembles of *MF* and *RBF* networks and does not depend on the ensemble size. Furthermore, *Majority Voting* and *Borda Count* are also good alternatives for ensembles of *RBF* networks because they provide good results and their performance do not depend on the normalization procedure. Finally, there have been some cases in which a "bad" combiner has provided high performance in a specific dataset (i.e. dataset *ionos* combined with *Bayesian Combination*). For this reason, we think that a deep study on combiners should be performed when a final ensemble is being designed for a specific classification problem.

Fifthly, two new reordering algorithms of the training set (*Static reordering* and *Dynamic reordering*) were proposed to be used with ensembles of neural networks in chapter 7. The main idea is that the order in which the patterns are presented to the networks should be different for each network of the ensemble and each iteration of the training procedure *Backpropagation*. The first results showed that we can improve the performance of *Simple Ensemble* if reordering is applied during training. Then, we tested if reordering was a better source of diversity than using different random initialization of the networks. Although we have seen that reordering is an important source of diversity, the best results are obtained if it is used along with using different initial network configurations or *starting points*. Both reordering algorithms slightly improved the best classic ensembles with traditional training but

Dynamic reordering is a slightly better alternative than *Static reordering*, in general, according to the results shown. Moreover, *Dynamic reordering* could be applied to any ensemble but the features of some ensembles (*CELS*, *EECNL*, *Boosting*, among others) do not allow to use *Static reordering* with them. However, as an example, *Static reordering* is a better choice for *CVC* and ensembles of 40 networks. The two new algorithms should be seriously considered and we think that they are another good contribution of this thesis.

Sixthly, some improvements on *Boosting variants* were successfully introduced in chapter 8. The new ensembles can be divided into three groups. In the first group, two new ensembles, *Averaged-Conservative Boosting* (*ACB*) and *Weighted-Conservative Boosting* (*WCB*), were successfully introduced. Their results slightly improve the results provided by the original boosting methods, *Adaboost* and *Conserboost* mainly. In the second group, the ensembles based on the new *Cross-Validated Boosting* methodology were also successfully proposed. In this new approach, the *Cross-Validation* model and the *Boosting* model were mixed into a single ensemble methodology. Fourteen new alternatives based on *Cross-Validated Boosting* were tested using two different combiners (*Output average* and *Boosting Combiner*). The new ensembles can provide high performance, specially when they are based on *CVCv3* and *Output average*, according to the general and statistical results shown. Moreover, the new alternatives based on *CVCv2* also provide good results when the number of networks in the ensemble is 9. This special behavior is caused by the partition done in *CVCv2* which benefits these ensembles but penalizes the performance of high sized ensembles. Finally, in the last group, we introduced new ensembles based on the two new proposed methods *ACB*, *WCB* and *Cross-Validated Boosting* (*CVCv2ACB*, *CVCv3ACB*, *CVCv2WCB* and *CVCv3WCB*). The results showed that *CVCv3ACB* and *CVCv3WCB* are the best overall ensemble methods. The first one, *CVCv3ACB*, statistically improved the best classic boosting methods such as *Adaboost*, *Aveboost* and *Conserboost*. The last one, *CVCv3WCB* provided the best overall general results in this thesis when the combiner *Output average* is applied (its mean PER is 32.49% and its mean IoP is 6.63%).

At this point, we have to remark that *CVCv3ACB* and *CVCv3WCB* are high refined ensembles. Although their origins are based on *Adaptive Boosting* and *Cross-Validation Committee version 1*, they have not inherited the problems of these two classic ensembles. In these new ensembles, a good partitioning procedure to generate the specific training and validations sets with *Cross-Validation* (the subsets are generated as in *CVCv3*) and the most advanced equations to update the sampling distribution of boosting (the "complex" soft equations introduced in *ACB* and *WCB*) are used together to design a more robust ensemble. The ensembles *Cross-Validation Committee v3*, *Averaged-Conservative Boosting* and *Weighted-Conservative Boosting*, and the idea of mixing *Cross-Validation* and *Boosting* into a single methodology have, **all of them**, been proposed in this thesis as improvements of classic ensembles. Furthermore, we introduced the global combination of all of these individual elements, and this "idea" provided the best overall results in this thesis as *CVCv3ACB*

and *CVCv3WCB* report. We consider that the new boosting variants proposed in this thesis (*ACB* and *WCB*), the new *Cross-Validation Boosting* methodology (with classic boosting ensembles) and their global combination (*CVCv2ACB*, *CVCv3ACB*, *CVCv2ACB* and *CVCv3ACB*), are an important contribution in the field of *Multiple Classifiers Systems* as the global results show.

Finally, other *Multiple Classifier Systems* have been analyzed. Concretely, the original versions of *Stacked Generalization* and *Mixture of Experts* have been implemented in order to compare their accuracy to *Ensembles of Neural Networks*. These alternatives were introduced in chapters 9 and 10 respectively.

Although the original classifiers based on *Stacked Generalization* provide good results, specially the *Ghorbani & Owrangh's* implementation, their performance is "similar" to the general results provided by *Simple Ensemble*. However, the *Stacked* model provides better results if it is issued to propose an ensemble combiner. In this thesis, the new combiners *Stacked* and *Stacked+* have been introduced. The results showed that good performance can be obtained by combining low-medium sized ensembles of *MF* and *RBF* networks by a single *MF* combination network with *Stacked* and *Stacked+*. Maybe, the new combiners do not work as desired for large ensembles because of the curse of dimensionality. Perhaps, the combination networks require so many inputs and the final prediction can not be properly obtained for these high sized ensembles.

Moreover, we have tested the new combiners using an ensemble of expert networks (generated with *Simple Ensemble*) and some ensemble alternatives to generate a set of combination networks. The ensemble alternatives used to generate the combination ensemble were: *Simple Ensemble*, *CVCv3ACB* and *CVCv3Conserboost*. The results showed that the ensemble model *Simple Ensemble* whose best PER result is 24.65% can be highly outperformed by using a combination ensemble. In fact, we achieve a mean PER equal to 28.82% with *Simple Ensemble* combined by *Stacked+* and a combination ensemble generated with *CVCv3ACB*.

The new combiners based on *Stacked Generalization*, *Stacked* and *Stacked+*, are a good contribution in the field of combining ensembles of neural networks. With them, the performance of the ensembles can be improved because the combination networks can learn from the errors committed by the expert networks. As we clearly see when we use *Stacked* and *Stacked+* to combine ensembles generated with *Inverboost* (*Inverboost* provided worse results than a single *MF* network but *Stacked combiners* fixed it). However, they can not fit on high sized ensembles in most of the cases. As we pointed out, this behavior may be due to the curse of dimensionality because the number of inputs of the combination networks increases as the number of networks increases.

The *Mixture* model was also successfully applied as a whole model and as an ensemble combiner. The original version (called *Mix-BN-BN* in this thesis) was designed as an alternative to complex network architectures, it provided better results than expected for some datasets and, in some cases, it clearly outperforms the single

MF network and *Simple Ensemble*. However, there were a few cases in which it performed quite worse than a single *MF* network. Maybe, this model only fits in those classification problems in which the original problem can be divided into some subproblems. Moreover, this original version has been improved by applying the *Multilayer Feedforward* network as experts and gating network. The new versions, *Mix-MF-BN* and *Mix-MF-MF*, perform better than the original implementation but slightly worse than a *Simple Ensemble* in general. We think that the three versions are important because they are based on *divide and conquer* so they are "different" to the classifiers generated with the ensemble approach.

Furthermore, this model was used to combine ensembles generated with *Simple Ensemble*. *Mix-SE-BN* and *Mix-SE-MF* were introduced to combine ensembles with a special weighted average. In the first alternative, a *Basic Network* was used to generate the gating network whereas the *Multilayer Feedforward* network was in charge of the gating network in the second combiner. The results showed that they slightly improve the performance of the original ensemble combined with *Output average*. Moreover, it seems that *Mix-SE-BN* provides slightly better results and it is less complex than the other alternative proposed, *Mix-SE-MF*.

11.2 General Conclusions

The final and most important conclusions of this thesis are discussed in this section. The presented conclusions clearly synthesize all the research done for more than seven years in the field of ensembles of neural networks and other advanced classification systems.

Firstly, neural networks are base generalizers which provide good results in classification tasks according the experiments we have realized in this thesis. In fact, this kind of classifiers are better than *K-Nearest Neighbors*, an important statistical classifier commonly used in the literature. Between the two architectures of neural networks analyzed in this thesis, *Multilayer Feedforward* (*MF*) and *Radial Basis Functions* (*RBF*), the last one provides better results but it can not be properly used with ensembles because its diversity degree is low (all the networks of an ensemble of *RBF* networks tend to be similar among them). To generate an ensemble of *MF* networks may be a better alternative than using a system based on *RBF* networks according to the performance, complexity and training requirements of the system.

Secondly, several classic ensemble alternatives from the literature have been tested and a few of them have been selected as the best ensembles when they are applied to solve classification problems. All these methods included in the list shown in the previous section provide excellent results for any database so they should be seriously considered to apply to any classification task. Moreover, *Simple Ensemble* has also been included in it because it is simple, reports good results and it can be used as base classifier to statistically compare the new alternatives. A new ensemble alternative should appear in the list of best ensembles if it improves *Simple Ensemble* and the statistical test denotes that the differences between this new alternative

and *Simple Ensemble* are statistically significant (if a new ensemble does not improve *Simple Ensemble* or their differences are not statistically significant, this new ensemble should not be considered a good alternative).

Moreover, new ensembles have been proposed in this thesis and most of them should also be included in the previous list because they provide better performance than *Simple Ensemble* and accomplish the statistical requirements. For instance, two new versions of *CVC* were proposed in this thesis and they were included in the first list of best ensembles because they are better than *Simple Ensemble* and their differences with respect to *Simple Ensemble* are statistically significant.

Furthermore, we introduced new ensembles in advanced researches. Using the new reordering algorithms, we have improved the performance of the best classic ensembles (*Simple Ensemble*, some *CVC* versions, *Decorrelated*, *Conserboost*, *Aveboost* among others) so the new versions of these ensembles should be seriously considered instead of the "traditional" versions without reordering. Moreover, the new *Averaged-Conservative Boosting* and *Weighted-Conservative Boosting* ensembles improve the best classic *Boosting variants* so they should be considered better alternatives than *Aveboost* and *Conserboost* (the best boosting variants according to the classic ensemble comparison). In addition, the best overall performance with ensembles of neural networks are provided by the new ensembles generated with the *Cross-Validated Boosting* methodology. In general, the boosting variants have been improved by using this new methodology when it is based on the partition of the learning set done by *CVCv3* and the final combiner is *Output average*. In fact, *CVCv3ACB* and *CVCv3WCB* (ensembles based on advanced boosting variants proposed by us and the new *Cross-Validated Boosting* methodology introduced in this thesis) provide the best overall performance. However, the new *Cross-Validated* alternatives based on *CVCv2* provide the best results in ensembles of 9 networks and *Boosting combiner* performs well in high sized ensembles (40 networks).

We think that the reordered versions of the classic ensembles, *ACB*, *WCB* and the cross-validated boosting ensembles based on *CVCv3* and *Output average* (even the most sophisticated versions, *CVCv3ACB* and *CVCv3WCB*) should be included in the list of the best ensembles.

Thirdly, as has been previously shown, the ensemble alternatives are improved step by step and sometimes the differences between one single improvement and the original ensemble are not statistically significant. Although the best overall results are provided by 40-network ensembles generated by *CVCv3ACB* and *CVCv3WCB*, our example will be based in *CVCv3ACB*. This ensemble is based on *CVCv3* and *ACB*. On the one hand, *CVCv3* is proposed in this thesis as an improved version of *CVCv2* and, therefore, *CVCv1*. On the other hand, *ACB* is also proposed in this thesis as refined version of *Conserboost* and *Aveboost* (both ensembles are "classic" improvements of *Adaboost*). In the results shown in chapter 8 (tables 8.4, 8.5 and 8.16), we can see that the differences between *ACB* and *Aveboost* are not statistically significant. Moreover, the differences of *ACB* with respect to *Conserboost* are not

also statistically significant. In both cases, *ACB* is slightly better than these classic boosting ensembles according to the mean *PER*. However, *CVCv3ACB* is clearly better than the classic boosting variants because it provides better mean *PER* and the differences with respect them are statistically significant. We can consider *ACB* as a slight improvement of the classic boosting variants but its refined version, *CVCv3ACB*, is clearly better than all of them.

With the last example we want to remark that we do not have to focus our conclusions in the intermediate results because a global view shows us that a sequence of improvements (even slight improvements) can provide us the best ensemble alternative and it can be quite better than the original one. In figure 11.1, we show a representative view of the sequence of improvements done on ensembles (along with the mean *IoP*) to achieve the best ensembles, *CVCv3WCB* and *CVCv3ACB*.

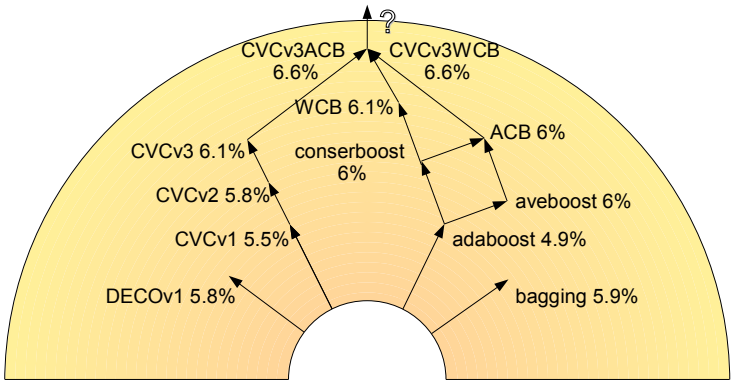


Figure 11.1: Diagram of best ensemble alternatives

Fourthly, selecting the most appropriate combiner for an ensemble is not an easy task. The combiner which should be applied to a specific ensemble will be chosen depending on the dataset, size of ensemble and alternative applied to design the ensemble. Initially, *Output average* should be always be used to fuse the networks because it provides good results in a wide majority of cases. Moreover, other combiners such as *Voting* and *Borda Count* can also be used for ensembles of *RBF* networks. Although the different *Boosting* variants introduce a specific combiner (*Boosting combiner*), *Output average* provides the best results for *Averaged-Conservative Boosting*, *Weighted-Conservative Boosting* and the best ensembles based on the *Cross-Validated Boosting* methodology. However, the *Boosting combiner* should also be considered because it provides good results for ensembles of 40 networks, maybe it does not work well on low and medium sized ensembles because there are not enough networks to apply its special weighted average.

Furthermore, the proposed *Stacked Combiners* and *Mixture Combiners* should also be seriously considered despite their training requirements because they provide an alternative view to fuse the networks. Specially, the new *Stacked Combiners* can learn from the mistakes done by an ensemble and they can provide high performance when they are used to combine low and medium sized ensembles. In the case of high

sized ensembles, the combination networks may not properly process the outputs of a high set of networks because of the curse of dimensionality.

Fifthly, the research of new ensembles and combiners should not be considered as finished. Although a simple improvement can not be considered "important", the summatory of some improvements can derive in good alternatives. We do not have to partially see how a new ensemble improves a previous version because a global view can show us that it is clearly a good choice to design ensembles of neural networks. As shown in this thesis, the best results are provided by *CVCv3ACB* and *CVCv3WCB*. Both methods have been designed after performing several improvements on *Cross-Validation* and *Boosting*. Maybe, with all the information provided in this thesis or by using a new alternative point of view, we can slightly improve them. Although the new hypothetical ensembles can be considered "similar" to our best ensembles, their refined versions could statistically outperform our best versions. As depicted in figure 11.1 with symbol '?', we do not know yet the performance of a new hypothetical ensemble based on *CVCv3ACB* or *CVCv3WCB* but we think it might be better.

Sixthly, other *Multiple Classifier Systems* have been studied despite this thesis has been focused on ensembles of neural networks. The original versions of both models have been implemented but they do not report extraordinary results. However, those alternative models are interesting because they address the classification problems in a different way so they might not commit exactly the same errors than an ensemble may.

Finally, we provide a deep analysis to other researchers. Some techniques are studied in this thesis and the "worst" ones can be discarded. Moreover, new good models are also introduced here. Other researchers have enough information in order to decide which alternatives can be used for their own classification problems. Moreover, they can avoid the systems which provide the worst results. Although we can inform about which is the alternative (or alternatives) with best overall performance, each final application is unique so its 'best' classification system will depend on the characteristics or features of the application.

For instance, there can be applications in which an error rate is tolerated whereas there are other applications where the highest performance is required. It is senseless to use an advanced ensemble if the performance provided by *Simple Ensemble* is enough for the problem. However, we should apply the best ensembles, i.e *CVCv3ACB* and *CVCv3WCB*, if we need a classifier with the highest performance.

Furthermore, there are applications in which the prediction should accomplish with some timing restrictions and we have to balance them with the performance of the classifier. An example is depicted in figure 11.2 where we use an advanced classifier to establish the path that a robot should follow in an orange grove. This classification system processes an image captured by the vision system (a) and predicts the class which belongs to each region of the input image (b). With this generated image and the information provided by other sensors, the final path, shown as the light blue line in (c), can be established. In this case, a high performance system composed

by a few networks has to be selected because it is better than a single network but also the computational requirements are lower than for a high sized ensembles. For instance, *CVCv2ACB* or *CVCv3ACB* can be used to generate a small ensemble or we can combine a small *Simple Ensemble* with *Stacked+* and a few combination networks.

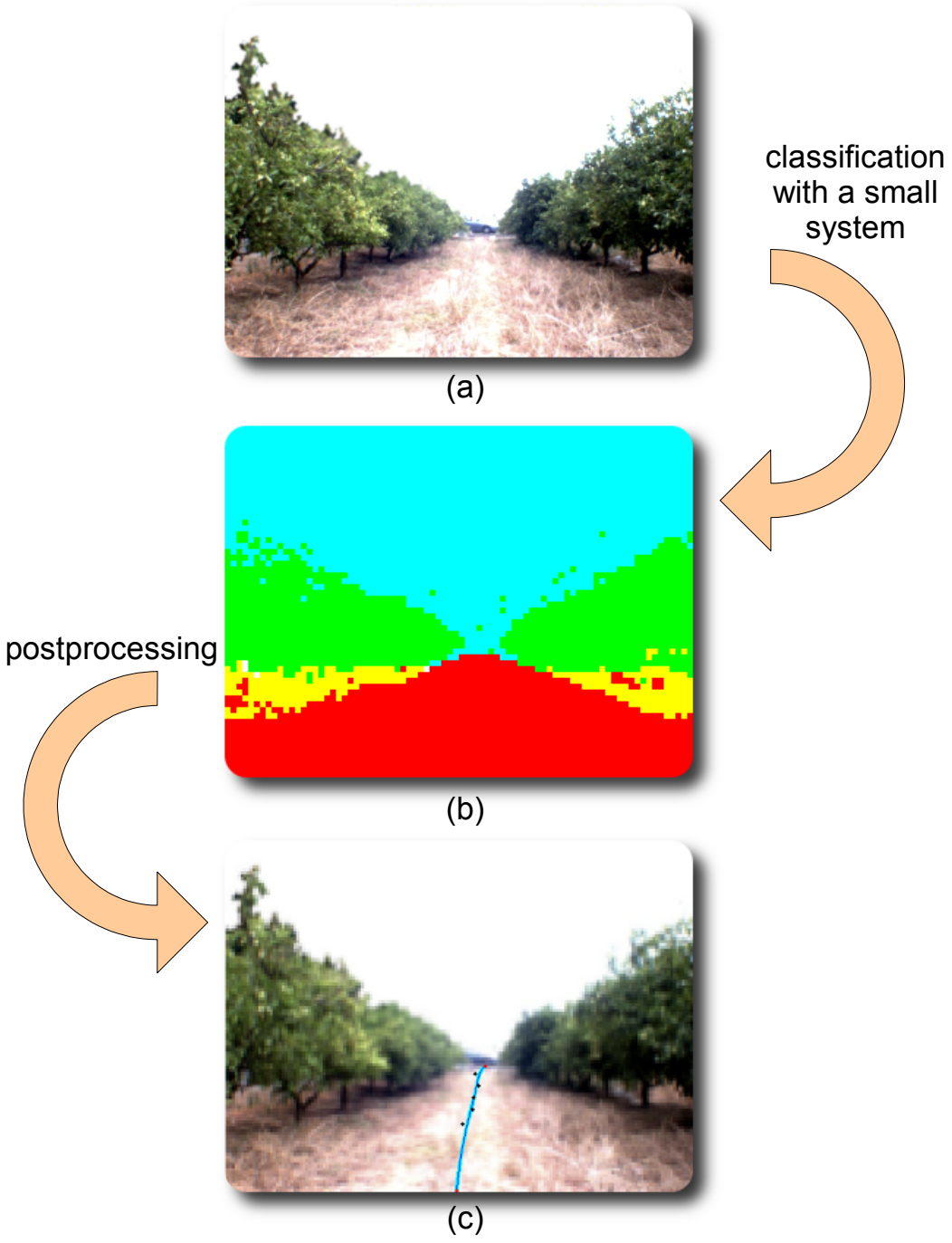


Figure 11.2: Path planner based on classification with neural networks

Moreover, there are applications in which one classifier (single neural network, ensemble or other multi-net system) is enough whereas there are applications which require some quite different classification systems that are consulted by an human expert. The following figure shows a possible application in which some different systems can be applied to solve the classification problem "*Diagnostic of breast cancer based on mammography*". In this hypothetical application, the diagnostic is given by the doctor considering his own experience, the medical history of the patient and the predictions provided by diverse classifiers based on neural networks.

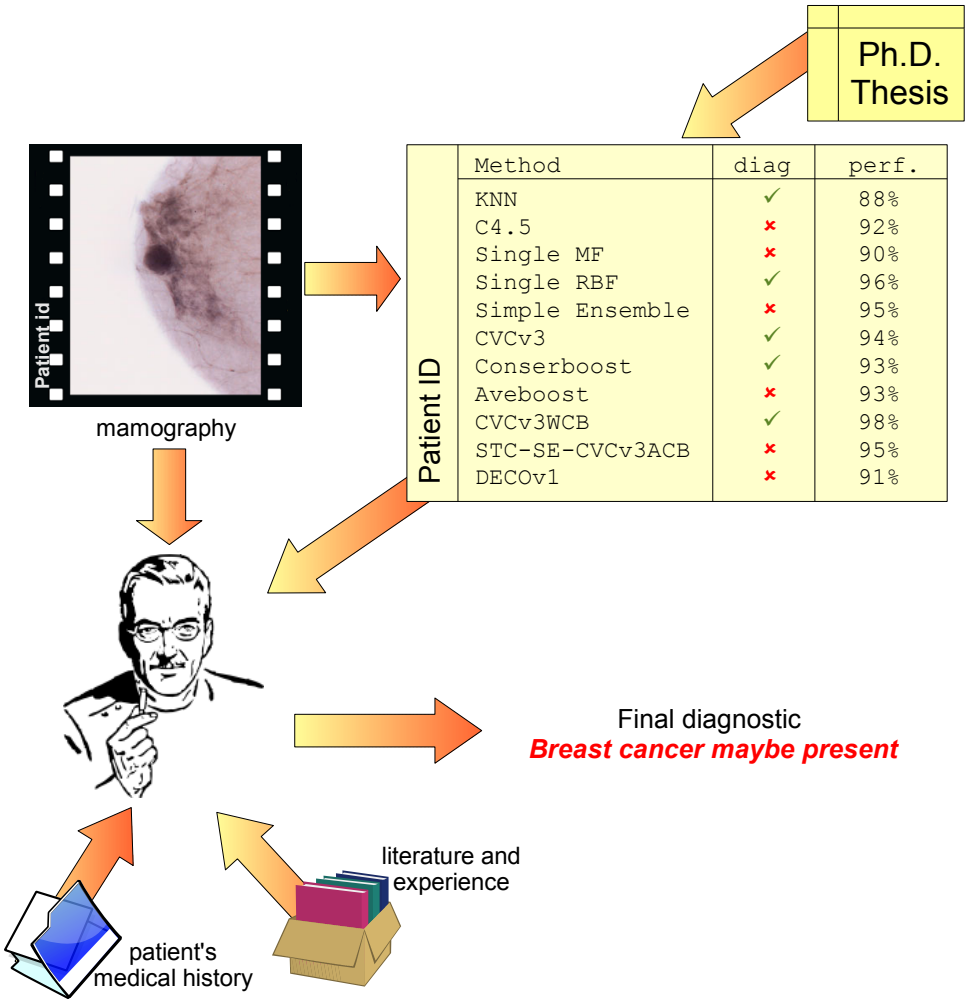


Figure 11.3: Breast Cancer diagnostic

In the previous figure, a final diagnostic of breast cancer is shown. The information provided in this thesis can be used to select some *different* approaches and classifiers with good performance: *DECOv1*, *CVCv3WCB*, *Stacked-SE-CVCv3ACB*, *Single RBF* among other alternatives. The prediction done by these advanced classifiers can be used by the specialist in order to provide the final diagnostic.

11.3 Publications

In this section, the most important publications related with the results introduced in this thesis are shown.

The results have been published in the most important conferences related to neural networks such as *International Work-Conference on Artificial Neural Networks* (*IWANN*), *International Joint Conference on Artificial Neural Networks* (*IJCNN*), *World Congress on Computational Intelligence* (*WCCI*) and *International Conference on Neuro-Information Processing* (*ICONIP*). All of them are on the top 60 conferences related to *Artificial Intelligence, Machine Learning, Robotics and Human Computer Interaction* being the *IJCNN* on the *23th* position [128].

Furthermore, *IJCNN* and *ICONIP* are cataloged to belong to group "*A*" (the most important and relevant category in this rank) according to "*The Excellence in Research for Australia Conference Ranking Exercise*" (*CORE*) [129] whereas *IWANN* and *International Conference on Artificial Neural Networks* (*ICANN*) belong to group "*B*".

11.3.1 Ensembles of *RBF* networks

Our papers which describe the research with the *RBF* network as single classifier and as an ensemble are shown here.

(RBF1) Mercedes Fernández-Redondo, Joaquín Torres-Sospedra and Carlos Hernández-Espinosa. *Training RBFs Networks: A Comparison Among Supervised and Not Supervised Algorithms*. In *Neural Information Processing*, volume 4232 of Lecture Notes in Computer Science, pages 477-486. Springer, 2006. *Proceedings of the International Conference on Neural Information Processing*.

(RBF2) Carlos Hernández-Espinosa, Mercedes Fernández-Redondo and Joaquín Torres-Sospedra. *Experiments on Ensembles of Radial Basis Functions*. In *Artificial Intelligence and Soft Computing* , volume 3070 of Lecture Notes in Computer Science, pages 197-202. Springer, 2004. *Proceedings of the International Conference on Artificial Intelligence and Soft Computing*.

(RBF3) Carlos Hernández-Espinosa, Joaquín Torres-Sospedra and Mercedes Fernández-Redondo. *Combination Methods for Ensembles of RBFs*. In *Artificial Neural Networks: Formal Models and Their Applications*, volume 3697 of Lecture Notes in Computer Science, pages 121-126. Springer, 2005. *Proceedings of the International Conference on Artificial Neural Networks*.

(RBF4) Joaquín Torres-Sospedra, Mercedes Fernández-Redondo and Carlos Hernández-Espinosa. *A comparison of combination methods for ensembles of RBF networks*. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 1137-1141. IEEE, 2005.

(RBF5) Mercedes Fernández-Redondo, Joaquín Torres-Sospedra and Carlos Hernández-Espinosa. *Combinación de Conjuntos de Redes Radial Basis Functions*. In *Actas del Simposio de Inteligencia Computacional, SICO2005*, pages 515-520. Thomson, 2005.

(RBF6) Joaquín Torres-Sospedra, Carlos Hernández-Espinosa and Mercedes Fernández-Redondo. *An Experimental Study on Training Radial Basis Functions by Gradient Descent*. In *Artificial Neural Networks in Pattern Recognition*, volume 4087 of Lecture Notes in Artificial Intelligence, pages 81-92. Springer, 2006. *Proceedings of the IAPR Workshop on Artificial Neural Networks in Pattern Recognition*.

11.3.2 Ensembles of *MF* networks: Comparisons

The papers published related to comparisons of ensembles of *MF* networks are the following ones:

(MFC1) Mercedes Fernández-Redondo, Carlos Hernández-Espinosa and Joaquín Torres-Sospedra. *Classification by Multilayer Feedforward Ensembles*. In Fuliang Yin, Jun Wang et al., editors, *Advances in Neural Networks*, volume 3173 of Lecture Notes in Computer Science, pages 852-857. Springer, 2004. *Proceedings of the International Symposium on Neural Networks*.

(MFC2) Joaquín Torres-Sospedra, Carlos Hernández-Espinosa and Mercedes Fernández-Redondo. *New results on ensembles of multilayer feedforward*. In Wlodzislaw Duch, Janusz Kacprzyk, Erkki Oja and Slawomir Zadrozny, editors, *Artificial Neural Networks: Formal Models and Their Applications*, volume 3697 of Lecture Notes in Computer Science, pages 139-144. Springer, 2005. *Proceedings of the International Conference on Artificial Neural Networks*.

(MFC3) Carlos Hernández-Espinosa, Mercedes Fernández-Redondo and Joaquín Torres-Sospedra. *Nuevos Experimentos sobre Conjuntos de RNAs*. In *Actas del Simposio de Inteligencia Computacional, SICO2005*, pages 19-25. Thomson, 2005.

(MFC4) Joaquín Torres-Sospedra, Mercedes Fernández-Redondo and Carlos Hernández-Espinosa. *Combination Methods for Ensembles of MF*. In Wlodzislaw Duch, Janusz Kacprzyk, Erkki Oja and Slawomir Zadrozny, editors, *Artificial Neural Networks: Formal Models and Their Applications*, volume 3697 of Lecture Notes in Computer Science, pages 133-138. Springer, 2005. *Proceedings of the International Conference on Artificial Neural Networks*.

(MFC5) Joaquín Torres-Sospedra, Mercedes Fernández-Redondo and Carlos Hernández-Espinosa. *A research on combination methods for ensembles of multilayer feedforward*. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 1125-1130. IEEE, 2005.

(MFC6) Joaquín Torres-Sospedra, Carlos Hernández-Espinosa and Mercedes Fernández-Redondo. *Combinación de Conjuntos de Redes Multilayer Feedforward*. In *Actas del Simposio de Inteligencia Computacional, SICO2005*, pages 11-18. Thomson, 2005.

11.3.3 Ensembles of *MF* networks: Boosting improvements

The papers, in which the proposed improvements of the *Boosting methods* have been published, are the following ones:

(MFBI1) Joaquín Torres-Sospedra, Carlos Hernández-Espinosa and Mercedes Fernández-Redondo. *Designing Multilayer Feedforward Ensembles with Cross Validated Boosting Algorithm*. In *Proceedings of International Joint Conference on Neural Networks, IJCNN 2006*, pages 2257-2262. IEEE, 2006.

(MFBI2) Joaquín Torres-Sospedra, Carlos Hernández-Espinosa and Mercedes Fernández-Redondo. *Mixing Aveboost and Conserboost to Improve Boosting Methods*. In *Proceedings of International Joint Conference on Neural Networks*, *IJCNN 2007*, pages 672-677. IEEE, 2007.

(MFBI3) Joaquín Torres-Sospedra, Carlos Hernández-Espinosa and Mercedes Fernández-Redondo. *Designing a Multilayer Feedforward Ensemble with the Weighted Conservative Boosting Algorithm*. In *Proceedings of International Joint Conference on Neural Networks*, *IJCNN 2007*, pages 684-689. IEEE, 2007.

(MFBI4) Joaquín Torres-Sospedra, Carlos Hernández-Espinosa and Mercedes Fernández-Redondo. *Researching on combining boosting ensembles*. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2008*, pages 2290-2295. IEEE, 2008.

11.3.4 Ensembles of *MF* networks: Reordering the training set

The proposed reordering techniques shown in chapter 7 have been published in:

(MFR1) Joaquín Torres-Sospedra. Carlos Hernández-Espinosa. and Mercedes Fernández-Redondo. *Adding Diversity in Ensembles of Neural Networks by Reordering the Training Set*. In Vera Kurková. Roman Neruda. and Jan Koutník. editors. *Artificial Neural Networks*, volume 5163 of Lecture Notes in Computer Science, pages 275-284. Springer, 2008. *Proceedings of the International Conference on Artificial Neural Networks*.

11.3.5 Ensembles of *MF* networks: Advanced models, *Stacked* and *Mixture*

The papers in which we introduce our research with *Stacked* and *Mixture* models as whole classifiers and ensemble combiners are:

(AM1) Joaquín Torres-Sospedra, Carlos Hernández-Espinosa and Mercedes Fernández-Redondo. *Combining MF Networks: A Comparison among Statistical Methods and Stacked Generalization*. In Friedhelm Schewenker and Simone Marinai, editors, *Artificial Neural Networks in Pattern Recognition*, volume 4087 of Lecture Notes in Artificial Intelligence, pages 210-220. Springer, 2006. *Proceedings of the IAPR Workshop on Artificial Neural Networks in Pattern Recognition*.

(AM2) Joaquín Torres-Sospedra, Carlos Hernández-Espinosa and Mercedes Fernández-Redondo. *Stacking MF Networks to Combine the Outputs Provided by RBF Networks*. In Joaquim Marques de Sá, editor, *Artificial Neural Networks*, volume 4668 of Lecture Notes in Computer Science, pages 450-459. Springer, 2007. *Proceedings of the International Conference on Artificial Neural Networks*.

(AM3) Carlos Hernández-Espinosa, Joaquín Torres-Sospedra and Mercedes Fernández-Redondo. *Researching on Multi-Net Systems Based on Stacked Generalization*. In Lionel Prevost, Simone Marinai and Friedhelm Schwenker, editors, *Artificial Neural Networks in Pattern Recognition*, volume 5064 of Lecture Notes in Artificial Intelligence, pages 193-204. Springer, 2008. *Proceedings of the IAPR Workshop on Artificial Neural Networks in Pattern Recognition*.

(AM4) Joaquín Torres-Sospedra, Carlos Hernández-Espinosa and Mercedes Fernández-Redondo. *Designing a New Multilayer Feedforward Modular Network for Classification Problems*. In *Proceedings of International Joint Conference on Neural Networks, IJCNN 2006*, pages 2263-2268. IEEE, 2006.

(AM5) Joaquín Torres-Sospedra, Carlos Hernández-Espinosa and Mercedes Fernández-Redondo. *The Mixture of Neural Networks Adapted to Multilayer Feedforward Architecture*. In De-Shuang Huang, Song Wu and Kang Li, editors, *Intelligent Computing*, volume 4113 of Lecture Notes in Computer Science, pages 488-493. Springer, 2006. *Proceedings of the International Conference on Intelligent Computing*.

(AM6) Mercedes Fernández-Redondo, Joaquín Torres-Sospedra and Carlos Hernández-Espinosa. *The Mixture of Neural Networks as Ensemble Combiner*. In Lionel Prevost, Simone Marinai and Friedhelm Schwenker, editors, *Artificial Neural Networks in Pattern Recognition*, volume 5064 of Lecture Notes in Artificial Intelligence, pages 168-179. Springer, 2008. *Proceedings of the IAPR Workshop on Artificial Neural Networks in Pattern Recognition*.

11.3.6 Other publications

In this subsection we introduce other papers we have published but their research has not been included in this thesis:

(OP1) Mercedes Fernández-Redondo. Carlos Hernández-Espinosa. and Joaquín Torres-Sospedra. *Hyperspectral Image Classification by Ensembles of Multilayer Feedforward Networks*. In *Proceedings of International Joint Conference on Neural Networks, IJCNN 2004, volume 2*, pages 1145-1149. IEEE 2004.

(OP2) Carlos Hernández-Espinosa, Mercedes Fernández-Redondo and Joaquín Torres-Sospedra. *Some Experiments on Ensembles of Neural Networks for Hyperspectral Image Classification*. In Mircea Gh. Negoita, Robert J. Howlett and Lakhmi C. Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems*, volume 3213 of Lecture Notes in Computer Science, pages 677-684. Springer, 2004. *Proceedings of the International Conference on Knowledge-Based Intelligent Information and Engineering Systems*.

11.4 Citations to our work

The citations related with the previous publications are introduced in this section. These citations involve important conferences, journals and patents. We consider that the citation of our work means that our investigation tasks have been useful for other researchers.

- Wu Di, Dai Ji, and Chi Zhongxian. *Intrusion detection based on an improved art2 neural network*. In *Proceedings of the Sixth International Conference on Parallel and Distributed Computing Applications and Technologies*, pages 234-238, IEEE Computer Society, 2005. Our paper [MFC1] has been cited.

- Vidya Manian, Luis O. Jimenez-Rodriguez, and Miguel Velez-Reyes. *A comparison of statistical and multiresolution texture features for improving hyperspectral image classification. Proceedings of SPIE - The International Society for Optical Engineering*, volume 5982, art. no. 59820I. SPIE, 2005. Our paper [OP1] has been cited.

- Guilherme Palermo Coelho and Fernando J. Von Zuben. *The influence of the pool of candidates on the performance of selection and combination techniques in ensembles*. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2006*, pages 5132-5139. IEEE, 2006. Our paper [MFC5] has been cited.

- Shivam Tripathi, V.V. Srinivas, and Ravi S. Nanjundiah. *Downscaling of precipitation for climate change scenarios: A support vector machine approach. Journal of Hydrology*, 330(3-4):621-640, Elsevier, 2006. Our paper [OP2] has been cited.

- Marjory Cristianny C. Abreu and Anne Magaly de Paula Canuto. *Evaluating the influence of the choice of the ensemble members in some fuzzy combination methods*. In *Proceedings of the International Joint Conference on Neural Networks*, pages 448-453. IEEE, 2007. . Our papers [RBF4,MFC5] have been cited.

- Marjory Cristianny C. Abreu and Anne Magaly de Paula Canuto. *An experimental study on the importance of the choice of the ensemble members in fuzzy combination methods*. In *ISDA'07:* Proceedings of the *Seventh International Conference on Intelligent Systems Design and Applications*, pages 723-728, Washington, DC, USA, 2007. IEEE Computer Society. Our papers [RBF4,MFC5] have been cited.

- Dingding Chen, Allan Zhong, John Gano, Syed Hamid, Orlando De Jesus, and Stan Stephenson. *Construction of surrogate model ensembles with sparse data. In IEEE Congress on Evolutionary Computation*, pages 244-251. IEEE, 2007. Paper [MFC5] has been cited.

- Anne Magaly de Paula Canuto and Marjory Cristianny C. Abreu. *Using fuzzy, neural and fuzzy-neural combination methods in ensembles with different levels of diversity*. In *Artificial Neural Networks - ICANN 2007, 17th International Conference, Porto, Portugal, September 9-13, 2007, Proceedings, Part I*, volume 4668 of Lecture Notes in Computer Science, pages 349-359. Springer, 2007. Our papers [RBF4,MFC5] have been cited.

- Iván Machón, Hilario López, J. Rodríguez-Iglesias, E. Marañón, and I. Vázquez. Short communication: *Simulation of a coke wastewater nitrification process using a feed-forward neuronal net. Environmental Modelling and Software*, 22(9):1382- 1387, Elsevier, 2007. Our paper [MFC2] has been cited.

- Vidya Manian and Luis O. Jimenez. *Land cover and benthic habitat classification using texture features from hyperspectral and multispectral images. Journal of Electronic Imaging*, 16(2):023011, SPIE and IS&T, 2007. Our paper [OP1] has been cited.

- Samuel Robert Reid. *Partial Ensemble Selection for Multiclass Classification. PhD Proposal*, Faculty of the Graduate School, University of Colorado, 2007. Our paper [AM1] has been cited.

- Samuel Robert Reid. *Model Combination in Multiclass Classification. Doctoral Dissertation*, Faculty of the Graduate School, University of Colorado, 2010. Our paper [AM1] has been cited.

- Lin-Tao Lv, Na Ji and Jiu-Long Zhang. *A RBF neural network model for anti-money laundering. Proceedings of the International Conference on Wavelet Analysis and Pattern Recognition, ICWAPR '08*, vol.1, pages 209-215, IEEE, 2008. Our paper [RBF1] has been cited.

- Márcio Leandro Gonçalves, Márcio Luiz De Andrade Netto, José Alfredo F. Costa, and Jurandir Zullo Junior. *An unsupervised method of classifying remotely sensed images using kohonen selforganizing maps and agglomerative hierarchical clustering methods. International Journal of Remote Sensing*, 29(11):3171-3207, Taylor & Francis, 2008. Our paper [OP2] has been cited.

- Dingding Chen, Allan Zhong, Syed Hamid, and Stanley Stephenson. Neural network based surrogate model construction methods and applications thereof. Patent record available from the US Patent Office, 2008. Paper [MFC5] has been cited.

- Vidya Manian and Miguel Velez-Reyes. *Support vector classification of land cover and benthic habitat from hyperspectral images. International Journal of High Speed Electronics and Systems*, 18(2):337-348, World Scientific, June 2008. Our paper [OP1] has been cited.

- Chris Matthews and Esther Scheurmann. *Ensembles of classifiers in arrears management.* In Bhanu Prasad, editor, *Soft Computing Applications in Business*, volume 230 of Studies in Fuzziness and Soft Computing, pages 1-18. Springer, 2008. Our paper [MFC4] has been cited.

- Shuyan Chen, Wei Wang, and Henk van Zuylen. *Construct support vector machine ensemble to detect traffic incident. Expert Systems with Applications*, 36(8):10976-10986, Elsevier, 2009. Our papers [RBF4,MFC5] have been cited.

- Davide Roverso. *System and method for empirical ensemble-based virtual sensing.* Patent record available from the World Intellectual Property Organization, 2009. Our paper [AM1] has been cited.

- Alan R. Hilal and Otman Basir. *Combination of enhanced adaboosting techniques for the characterization of breast cancer tumors. Proceedings of the International Conference on Future BioMedical Information Engineering*, FBIE 2009, 2009, 568-571. Our paper [MFBI4] has been cited.

11.5 Future work

The research done in this thesis can be complemented with the following three opened lines that has been considered for future work:

- Continuing the research in ensembles.
- Other researches related to ensembles of neural networks.
- Study and development of real optimized applications.

11.5.1 Continuing the research in ensembles

As has been commented before, we do not consider the research in designing ensembles as finished. In this opened line, we will use the knowledge acquired in this thesis in order to design more robust ensembles. With design we mean the procedure to train the different networks and the technique applied to fuse all the information provided by the individual networks.

For instance, we are nowadays working in optimized versions of *Decorrelated* and *Cooperative Ensemble Leaning System* and the first results seem to be good. Furthermore, we do not discard to use other different alternatives proposed recently in the literature with the purpose of designing better ensembles of neural networks.

We think that this line is the natural continuation or future work of the research done in this thesis. However, we will not focus only on it and we will explore new alternatives to ensembles and other optimization procedures.

11.5.2 Other researches related to ensembles of neural networks

This thesis has been focused on ensembles of neural networks and other models based on *Multiple Classifier Systems*. However, we have focused only on selecting the "optimal" training parameters for the network architecture (number of hidden nodes, adaptation step among others) and the most appropriate specific parameters for the different ensemble alternatives and combiners (i.e the λ value of *Decorrelated*, α in *Weighted-Conservative Boosting* or γ in *Dinamically Averaged Networks*). Fortunately, there are some other alternatives in order to improve the performance of the neural networks. Although some of them depend on the classification problem, the alternatives we will consider are:

- Input selection.
- Weight decay.
- Weight initialization.
- Pruning.
- False positive/negative reduction.
- Advanced combination of classifiers.

Input selection is used to denote the process of selecting those inputs which are most relevant in describing the output. In complex classification problems there are a huge quantity of information which has to be processed. For instance, each pattern of database *aritm* has 277 input elements. Another important example, the input vector used in the combination networks of *Stacked combiners* is high when the ensemble size is also high (i.e. 40 expert networks). A priori, we do not know the importance of each individual input and there can be features which may not be useful for classification. Those useless inputs could decrease the performance of the network. For these reasons, they should be removed and those inputs which are retained after this process can be considered the most important features which the neural network can process. We consider that this methodology can be applied in two levels, to the independent networks (optimizing each network of an ensemble) and directly to the whole ensemble (optimizing the performance of the ensemble by selecting the most appropriate inputs and only use them in the individual networks). In [19] a deep analysis of *input selection* methods in *MF* networks was introduced, among all the alternatives they concluded that the analysis of trained *MF* networks provided the best results, these alternatives were cataloged to belong to the group "*ARNMF*" in that reference. Moreover, a technique based on the analysis of the training set also provided good results.

The best alternatives to perform *input selection* according to reference [19] are shown in the following list. They are ranked in descending order according to their performance so the first element can be considered the best choice.

- UTA [130].
- GD-DS [131].
- TEKA[132, 133].
- BL2 [134].
- DEV [135].
- TEKB [132, 133].
- DER2 [136].
- MAO [137].
- CLO [138].
- BL1 [134].
- BOW [139].
- YOU [140].
- BA (with β set to 0.5) [141].

All the methods shown in the previous list are described in their references and they are also described and analyzed in [19]. Moreover, there are some recent methods which can be also considered and they are introduced in [142, 143].

Weight Decay was introduced by Paul J. Werbos [144]. With this procedure, the weights of the links are decreased while they are trained with *Backpropagation*. This methodology can be applied along with input selection [145].

Although the *classical weight initialization* procedure has been the only alternative used in this thesis. Some advanced techniques will be considered for future work. In chapter 7, we saw that the performance of *Simple Ensemble* could increase if the interval of weight initialization was also increased. The best mean PER for the normal *Simple Ensemble* is 25.84% but we can achieve a 28.14% if we use a higher interval. The initial configuration of a network depends on how the weights are initialized and it seems that the diversity degree among all the networks of a basic ensemble also depends on this procedure. For this reason, we think that considering other alternatives to set the initial weight values (starting points of the minimization) should also be tested with ensembles. In this case, we can apply each alternative to the ensemble in order to select the most important one and use it to establish the initial configuration of all the networks. Furthermore, we can also evaluate all these techniques on the individual networks and use the best performing ones to generate the individual networks in a way in which the different networks of an ensemble could use different alternatives to establish the initial configuration. We think that the diversity of an ensemble can be increased if we use this last alternative. Concretely, the initialization techniques described in [146, 147, 148] provided the best performance as reported in [19]. However, there are more alternatives in the literature which can also be tested.

Pruning is one of the most researched areas and this approach involves the removal of connections and hidden units based on the value of the weights. With this procedure the less important weights and/or hidden neurons can be removed while the accuracy of the networks can be increased. Moreover, the final architecture of the network tends to be less complex because it can contain less elements. In this way, the optimized networks can provide better performance and the computational cost of the prediction can be reduced. There are some interesting pruning alternatives, i.e. the techniques described in [149, 150, 151], that could be applied to the best ensembles generated in this thesis. *Prunning* can also be used in two levels, optimizing the individual networks or applying directly to the ensemble in order to eliminate the same link and/or hidden unit in all the networks.

False positive/negative reduction can be introduced to the problems in which a *false positive* is much worse than a *false negative* or vice versa. There are some applications in which not only the performance is important. In these problems, there are external factors, such as economical or psychological, which should be seriously considered. For instance, the negative diagnosis of a severe decease such as cancer when the patient is ill can have physical consequences which are not allowed. The existing illness is underestimated and does not receive attention. As it was shown in [152, 153, 154, 155] there are some techniques that can be successfully applied to determined problems. This new criteria (false positive is more important than a false negative or vice versa) can also be introduced to the ensemble model.

Advanced combination of classifiers will be focused on properly combining a set of diverse ensembles and other advanced models. In this thesis, combining has been focused only on fusing the information provided by an ensemble of neural networks. However, we consider that the use of a set of heterogeneous advanced classifiers such as different ensembles of *MF* networks with high performance (*Decorrelated*, best *Boosting* variants, *CVCv3*, our *CVCv3ACB* and *CVC3WCB*, among others), advanced *Stacked systems* (*Stacked G&O*, *Stacked-Inverboost-SN*, *Stacked-SN-CVCv3ACB*) and the best *Mixture approaches* (*Mix-MF-MF* and *Mix-SE-BN*) can be used together in order to provide a final global output. Although some of them may provide similar performance, their origins are quite different so they do not have to classify exactly in the same way and they may not agree in all the individual predictions. In this way, the final prediction is easily reached if all the different advanced systems agree but this prediction is less "clear" if there is not a consensus among all the classifiers. As Ho expressed in [45, 46], maybe we should combine effectively all the "simple" classifiers (at this stage we can consider as "simple classifier" an ensemble of neural networks and the other models analyzed in this thesis) we dispose in order to obtain a more accurate classification system.

In the advanced combining we propose, the information provided by some different classifiers, including ensembles and other *MCS*, will be considered for the final output. A superior model can be designed if the knowledge acquired by the different ensembles methods and other models are properly combined and it could provide better accuracy compared to the best ensemble methods (as an ensemble performs better than any individual network). Advanced combiners can be applied to perform this task as was suggested in [127].

11.5.3 Real optimized applications

The research shown in this thesis has been done on an heterogeneous set of databases. In the future, the knowledge acquired during the last years can be applied to generate final applications based on neural networks on a few specialized and interesting datasets. Concretely, the following three real problems have been initially considered:

- *SPAM detection on electronic mail.*
- *Diagnostic of breast cancer based on mammography.*
- *Terrain classification for navigation into orange groves.*

Although they are introduced in the following subsections, they are not fully described because they are consider for further work and some concrete parts may vary in the final design.

11.5.3.1 SPAM detection on electronic mail

In this first application we consider, each e-mail can be processed by a neural networked system in order to classify it as normal mail or *SPAM*. We consider that this "filter" can be useful because *SPAM* is a problem which had grown exponentially the last years, and today comprises nearly the 90% of all the e-mail in the world.

In figure 11.4 we show the percentage of *SPAM* e-mail in january 2010 according to the bulletin published by Karpesky Labs [156].

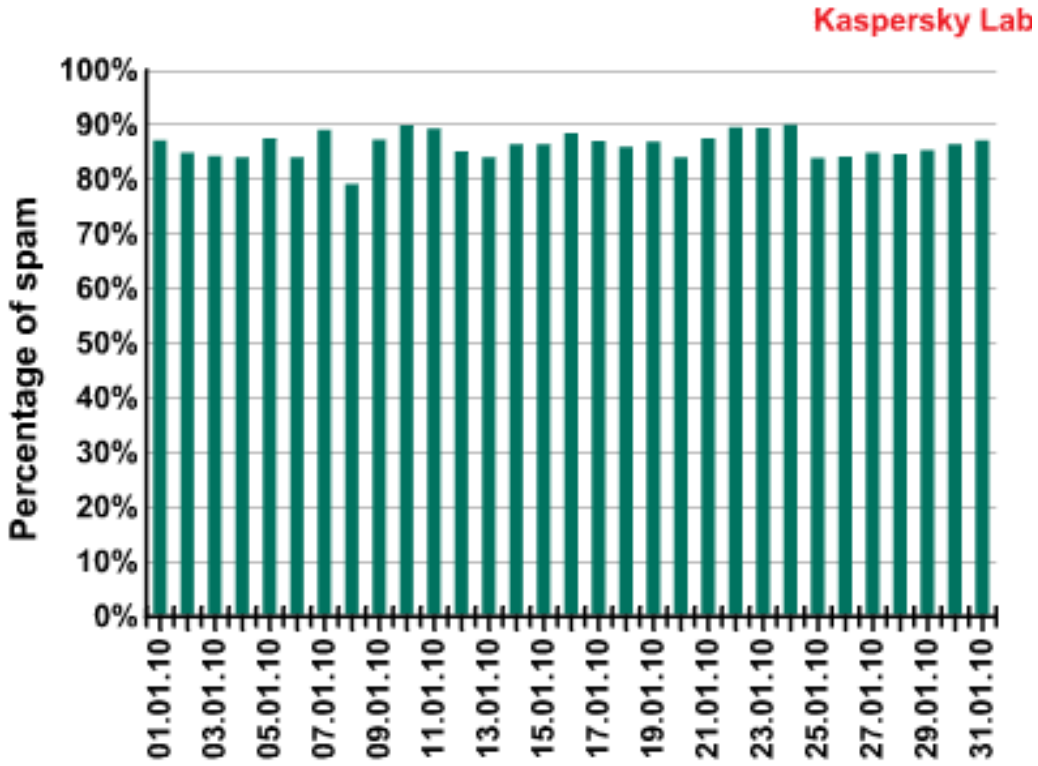


Figure 11.4: Spam percentage in email - January 2010 (Kaspersky Labs)

Moreover, the economical costs are quite high for the *Internet Service Providers*, *ISP*, and the corporations and institutions which have to buy specific anti-viruses and *SPAM* filters. For these reasons, we consider that a final application based on *SPAM* detection is quite interesting for final users (they do not receive viruses or useless mail) and for the *ISP* (they can filter conflictive mail and reduce the data traffic).

The predictions can be done by the local e-mail client (i.e. Mozilla Thunderbird), by the e-mail server or by both. Some features are extracted from the e-mail (such as the frequency of a determined word, the appearance of special characters or the use of capital words) and they can be used to perform the prediction. Although in the *UCI repository of machine learning* there is a dataset generated by *Hewlett-Packard Laboratories* which can be used, we consider that we should generate a new dataset because *SPAM* is in constant evolution and part of them is specific to the location (country or region).

Finally, we can find in the literature that there are some recent publications [157, 158, 159, 160] with proposals to detect *SPAM* in e-mail so, we think, it is an active research field.

11.5.3.2 Diagnostic of breast cancer based on mammography

In this second application, we think that an advanced classification system can be used to suggest the diagnosis of *Breast Cancer* to a specialist. In this case, a classifier based on neural networks is used to predict if a mass detected in a mammography is benign or malign as depicted in figure 11.5.

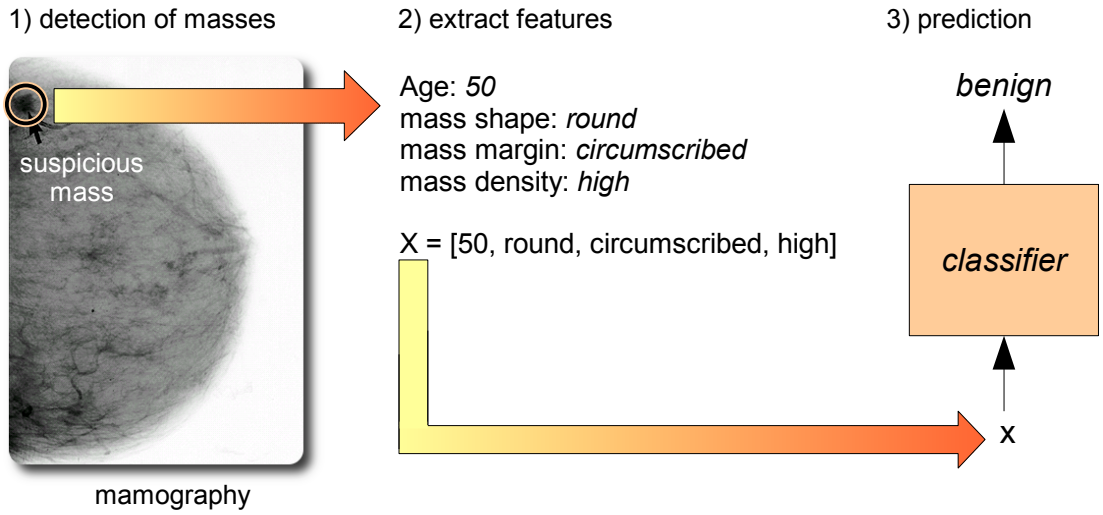


Figure 11.5: Example of the process of detecting and classifying a suspicious mass in a mammography

As we mentioned, this concrete application may be considered as a tool instead of a pure automated diagnostic application. The final decision or diagnosis is provided by the specialist which uses, among other sources of information, the prediction of one, or more, advanced classifiers. We think that a tool of these characteristics can be useful because it can have a high generalization capability and it can be considered a good "supporting" system. This aim was introduced in figure 11.2.

Moreover, we can apply the optimization procedures described in the previous subsection in order to improve the performance of the classifiers and reduce the false negatives in the diagnosis of breast cancer. As it has been mentioned, the consequences of this diagnosis in ill patients can not be allowed. However, its important to mention that the final diagnosis is always given by a specialist who also uses his own experience in order to give the final diagnosis. In conflictive cases, he has to take the last decision.

Finally, according to the bibliography [161, 162, 163, 164, 165], this kind of applications is interesting and the research on this field does not decrease. Moreover, we have recently performed a first analysis of this problem using two versions of database "*Mammographic Mass Data*" from the *UCI repository of machine learning* [70] and the results are good. In this database, the age of the patient and three features extracted directly from the mammography (mass shape, mass margin and

mass density) are used to predict if it is benign or malign. Moreover, the assessment assigned in a double-review process by physicians is also considered for the prediction. In the first study we performed, we could saw that this assessment is important in the final classification and it is a good input feature. However, it has been a first study and we have to perform a deeper analysis in order to provide optimized classification systems in which these assessments can be present or not.

This first research has been done under the supervision of *Carlos Hernández-Espinosa Ph.D.* and *Mercedes Fernández-Redondo Ph.D.* and we think it can be a good research line so we are still working on it.

11.5.3.3 Terrain classification for navigation into orange groves

In this application, we want to use a neural network based system in order to help an autonomous robot in its navigation into an orange grove. We decided to use it because a similar application was performed with neural networks in [166]. In that paper, a neural network was used for terrain classification tasks in a more general way because it was introduced for detection of any kind of off-road terrains in military applications. We think that this approach can also be used in our specific problem because our environment is more "controlled". However, it is not totally controlled because the weather and lightning conditions are important factors. These "uncontrolled" factors aimed us to use neural networks in classification because using simple color-based approaches were not enough. With "more controlled", we meant that our outdoor environments (where our robot should work) are simpler than the ones used in the reference because the basic structure of the different orange groves are similar (in the reference their robot should deal with an heterogeneous set of environments which were quite different among them).

Concretely, our proposed classifier processes a captured image in order to predict its areas as *Sky* (light blue in the image), *Soil/Land* (red), *Orange trunks* (yellow) and *Orange Crown* (green) as depicted in figure 11.6.

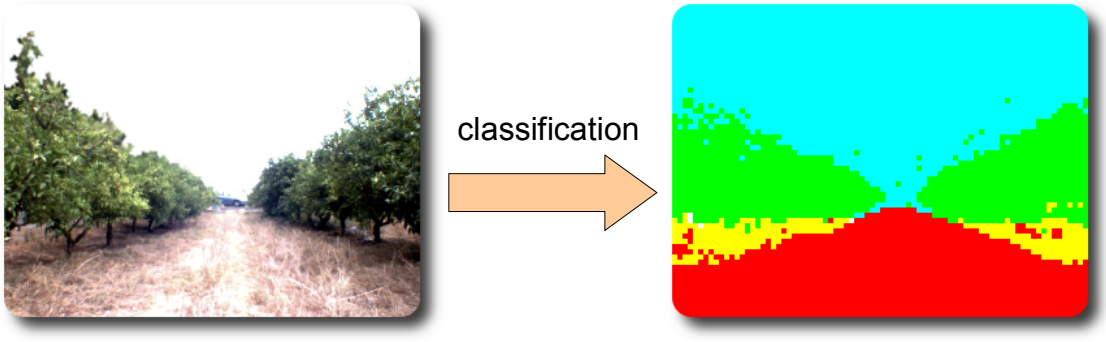


Figure 11.6: Image captured at an orange grove (left) and predicted classes (right)

However, the classification is not directly done with the original image. According to [166], the *two-level Daubechies wavelet transform* (concretely *Daub2*) is adopted for

each *HSL* channel of the captured image. *HSL* is one of the most common cylindrical-coordinate representations of points in an *RGB* color model and a pixel value depends on the *hue*, *saturation*, and *lightness*. Each wavelet provides seven sub-band images so a total of 21 sub-band images (3 channels and 7 sub-band images per channel) are used for classification. Then, some parameters (*Energy* and *Mean*) are calculated with the values of the sub-band images for each block of the original image. A block is a $N\ x\ M$ area of the original image and the parameters of the wavelet are calculated using the pixels of the sub-band images which belongs to the original area. Moreover, the relative position in the image (vertical and horizontal axises) are also used in the classification. In the original reference it has been suggested that this last information is useful because, for instance, the sky is always located at the top of the image. Finally, the input vector is generated as described in the reference. We do not describe deeper how the vector inputs are generated because this project is still in an initial stage and, maybe, the input features can vary.

Once the classification tasks are done, the predictions will be used to obtain the boundaries of the land (or soil) with the orange trunks and soil. To calculate the border lines we will use the *Hough transform*. In the example (figure 11.7), the border lines have been calculated in some different sections because the original borders are complex and, sometimes, are not given by simple lines. With the border lines (blue lines in figure 11.7 (a)), we calculate the center of the path. This path, shown as the green lines in figure 11.7 (a), along with other information obtained with *GPS*, satellite maps and other sensors will be send to the navigation module of the robot to establish the final robot path or correct its route. In figure 11.7 (b), we show an example of the final desired route (light blue line drawn on the image). To calculate this final route, we have selected one representative point for each section (denoted as black dot in figure 11.7 (b)) and we have used them as control points of a Bezier's curve.

Figure 11.7: Captured image with: (a) borders and center of the path, (b) desired path

In figure 11.8, we show two more examples of calculated desired paths in other captures from different orange groves.

Figure 11.8: Desired path from other captured images

Finally, the classification performed in this application is a part of the navigation module. Moreover, the prediction must be provided as faster as possible because the robotic system is in a constant movement. For these reasons, the classifier should be designed considering its performance, its accuracy in the border regions and the time required to process the image. The whole robotic system is run in real time so the time requirements are also important. Its important to mention that nowadays we are working in the design of this application and we have to balance the performance of the system with other external requirements such as timing costs and selection of the most appropriate vision system. I.e high resolution cameras (more than 5 mega pixel) provide better images but the economic costs of this kind of vision systems are high and the time required to process the images is also high. We have to select the most appropriate video system by balancing (economical and timing) costs and performance. This research is being supervised by *Patricio Nebot-Roglá Ph.D.* and it is supported by the Generalitat Valenciana under project GV/2010/087 and by the *Fundació Caixa Castelló - Bancaixa* under project *P1-1A2008-12*.

Appendix A

Datasets

A.1 Introduction

The datasets applied in our experiments will be reviewed in this appendix. All of the datasets described are from the *University of California Irvine: Machine Learning Repository* [70].

A.2 Arrhythmia Data Set

The *Arrhythmia Data Set*, henceforth *aritm*, distinguishes between the presence and absence of cardiac arrhythmia and classify it in one of the 16 groups.

This dataset was donated by H. Altay Guvenir. Concerning the study of H. Altay Guvenir [167]: "The aim is to distinguish between the presence and absence of cardiac arrhythmia and to classify it in one of the 16 groups. Class 01 refers to 'normal' ECG classes 02 to 15 refers to different classes of arrhythmia and class 16 refers to the rest of unclassified ones. For the time being, there exists a computer program that makes such a classification. However there are differences between the cardiolog's and the programs classification. Taking the cardiolog's as a gold standard we aim to minimize this difference by means of machine learning tools."

In our experiments we have used the 2-class version of the dataset in order to only determine the presence and absence of cardiac arrhythmia.

The most important characteristics of the database are:

Patterns used:	442	Training:	284
		Validation:	71
		Test:	87
Input parameters:	277	Categorical, Integer, Real	
Classes:	2	Binary	

A.3 Balance Scale

The *Balance Scale*, henceforth *bala*, was generated to model psychological experimental results. Each example is classified as having the balance scale tip to the right, tip to the left, or be balanced. The attributes are the left weight, the left distance, the right weight, and the right distance. The correct way to find the class is the greater of $left_{distance} \cdot left_{weight}$ and $right_{distance} \cdot right_{weight}$. If they are equal, it is balanced.

This dataset was donated by Tim Hume and it was firsly used by Robert Siegler experiments [168].

The most important characteristics of the database are:

Patterns used:	625	Training:	395
		Validation:	105
		Test:	125
Input parameters:	4	Categorical	
Classes:	3	Multiclass	

A.4 Cylinder Bands

The aim of *Cylinder Bands*, henceforth *band*, dataset is to predict delays in rotogravure printing. This delay is also known as cylinder banding. This dataset was donated by Bob Evans who firstly used it in [169].

The most important characteristics of the database are:

Patterns used:	277	{	Training:	177
			Validation:	45
			Test:	55
Input parameters:	39	Categorical, Integer, Real		
Classes:	2	Binary		

A.5 Liver Disorders Data Set

The purpose of *Liver Disorders Data Set* (*bupa*) is to predict if a patient suffers a liver disorder or disease from blood tests. This dataset was donated by Richard S. Forsyth its description can be found in [170].

Patterns used:	345	{	Training:	220
			Validation:	55
			Test:	70
Input parameters:	6	Categorical, Integer, Real		
Classes:	2	Binary		

A.6 Credit Approval Data Set

Credit Approval Data Set, henceforth *cred*, is a classification problem which concerns credit card applications. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data.

This dataset was anonymously submitted.

The most important characteristics of the database are:

Patterns used:	653	{	Training:	418
			Validation:	105
			Test:	130
Input parameters:	15	Categorical, Integer, Real		
Classes:	2	Binary		

A.7 Dermatology Data Set

The aim of *Dermatology Data Set*, henceforth *derma*, is to correctly diagnose a erythemato-squamous disease. The differential diagnosis of erythemato-squamous diseases is a real problem in dermatology. They all share the clinical features of erythema and scaling, with very little differences. The diseases in this group are psoriasis, seboreic dermatitis, lichen planus, pityriasis rosea, cronic dermatitis, and pityriasis rubra pilaris. The dataset was donated by H. Altay Guvenir.

The most important characteristics of the database are:

Patterns used:	358	{	Training:	229
			Validation:	58
			Test:	71
Input parameters:	34	Categorical, Integer		
Classes:	6	Multiclass		

A.8 Ecoli Data Set

The purpose of *Ecoli Data Set*, henceforth *ecoli*, is to predict the Cellular localization of proteins. The early versions of the dataset were generated by Kenta Nakai [171, 172] and the final version was donated by Paul Horton [173].

The most important characteristics of the database are:

Patterns used:	336	{	Training:	214
			Validation:	54
			Test:	68
Input parameters:	7	Real		
Classes:	8	Multiclass		

A.9 Solar Flare Data Set

Solar Flare Data Set, henceforth *flares*, concerns on predicting the number of solar flares of a certain class that occur in a 24 hour period. The dataset applied in the experiments is `'flare.data1'`.

The dataset was donated by Gary Bradshaw. The most important characteristics of the database are:

Patterns used:	1066	{	Training:	680
			Validation:	171
			Test:	215
Input parameters:	10	Categorical		
Classes:	2	Binary		

A.10 Glass Identification Data Set

At the scene of the crime, the glass left can be used as evidence if it is correctly identified. The aim of *Glass Identification Data Set*, henceforth *glas*, is to correctly identify a piece of glass and classify it as: building windows (float processed or not), vehicle windows (float processed or not), containers, tableware or headlamps.

This dataset was created by B. German and donated by Vina Spiehler. The most important characteristics of the database are:

Patterns used:	214	{	Training:	129
			Validation:	35
			Test:	50
Input parameters:	10	Real		
Classes:	6	Multiclass		

A.11 Heart Disease Data Set

The aim of *Heart Disease Data Set*, henceforth *hear*, is to predict if a patient suffers from a cardiac disease. The dataset was generated at the Long Beach and Cleveland Clinic Foundation and the data set was donated by David W. Aha.

The most important characteristics of the database are:

Patterns used:	297	{	Training:	190
			Validation:	48
			Test:	59
Input parameters:	13	Categorical, Integer, Real		
Classes:	2	Binary		

A.12 Image Segmentation Data Set

The *Image Segmentation Data Set*, henceforth *img*, was generated to classify a 3x3 region area from seven different outdoor images. The possible classes are: brickface, sky, foliage, cement, window, path, grass.

This dataset generated at the Vision Group of the University of Massachusetts and was donated by Carla Brodley.

The most important characteristics of the database are:

Patterns used:	2311	{	Training:	1476
			Validation:	370
			Test:	465
Input parameters:	19	Real		
Classes:	7	Multiclass		

A.13 Ionosphere Data Set

This radar data was collected by a system in Goose Bay, Labrador, see reference [174]. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts . The targets were free electrons in the ionosphere. 'Good' radar returns are those showing evidence of some type of structure in the ionosphere. 'Bad' returns are those that do not; their signals pass through the ionosphere.

The purpose of *Ionosphere Data Set*, henceforth *ionos*, is to determine if there is any protuberance in the ionosphere. This dataset was donated by Vince Sigillito. The most important characteristics of the database are:

Patterns used:	351	{	Training:	225
			Validation:	56
			Test:	70
Input parameters:	34	Integer,Real		
Classes:	2	Binary		

A.14 The MONK's problems Data Set

The *MONK's problem*, henceforth *mok*, were the basis of a first international comparison of learning algorithms. The result of this comparison is summarized in [175]. Concretely we have applied the problems MONK1 (*mok1*) and MONK2 (*mok2*).

The most important characteristics of the database are:

Patterns used:	432	{	Training:	282
			Validation:	70
			Test:	80
Input parameters:	6	Categorical		
Classes:	2	Binary		

A.15 Pima Indians Diabetes Data Set

The purpose of *Pima Indians Diabetes Data Set*, henceforth *pima*, is to determine if the patients, females at least 21 years old of Pima Indian heritage, suffer diabetes. This dataset was donated by Vince Sigillito and the original owner is the National Institute of Diabetes and Digestive and Kidney Diseases. The most important characteristics of the database are:

Patterns used:	768	{	Training:	488
			Validation:	125
			Test:	155
Input parameters:	8	Integer,Real		
Classes:	2	Binary		

A.16 Haberman's Survival Data Set

The *Haberman's Survival Data Set*, henceforth *survi*, contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer. The aim of the database is to determine if the patient survive at least 5 years or not.

This dataset was donated by Tjen-Sien Lim. The most important characteristics of the database are:

Patterns used:	306	{	Training:	196
			Validation:	49
			Test:	61
Input parameters:	3	Integer		
Classes:	2	Binary		

A.17 Congressional Voting Records Data Set

The *Congressional Voting Records Data Set*, henceforth *vote*, is related to the votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. The CQA lists nine different types of votes: voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified to nay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known (these three simplified to an unknown disposition). The purpose of the database is to determine if a congressman is democrat or republican.

The most important characteristics of the database are:

Patterns used:	435	{	Training:	282
			Validation:	70
			Test:	80
Input parameters:	16	Categorical		
Classes:	2	Binary		

A.18 Vowel Recognition (Deterding data)

The aim of the *Vowel Recognition (Deterding data)*, henceforth *vowel*, is to recognize the vowel pronunciation from the eleven steady state vowels of British English.

This database is part of the *Connectionist Bench Data Set* publicly available at the UCI repository [70]. The most important characteristics of the database are:

Patterns used:	990	{	Training:	282
			Validation:	70
			Test:	80
Input parameters:	11	Real		
Classes:	11	Multiclass		

A.19 Breast Cancer Wisconsin (Diagnostic) Data Set

The aim of *Breast Cancer Wisconsin (Diagnostic) Data Set*, henceforth *wdbc*, is to predict if a breast tumor is benign or malignant from a digitized image of a fine needle aspirate of a breast mass.

The dataset created by Dr. William H. Wolberg, W. Nick Street and Olvi L. Mangasarian was donated by Nick Street. The most important characteristics of the database are:

Patterns used:	569	{	Training:	369
			Validation:	90
			Test:	110
Input parameters:	30	Real		
Classes:	2	Binary		

Appendix B

Equations and parameters

B.1 Introduction

As we have seen in this dissertation, there are some ensemble alternatives in which the equations applied to adapt the trainable parameters are changed. These ensembles were introduced in chapter 4.

Moreover, some parameters are required to train the artificial neural networks and some multiple classifier systems. In the majority of cases, those parameters depends on the database and on the number of networks of the classification system.

In this appendix these new equations and all the parameters required to train the individual networks and the specific parameters used to generate the ensembles and other multiple classifier systems will be shown.

B.2 New equations to adapt trainable parameters

B.2.1 CELS

A specific adaptation of the trainable parameters has to be applied in *CELS* because the training algorithm used is based on the minimization of the error function and a penalty term has been added to it. This penalty is based on the network output which depends on the trainable parameters (the weights).

In the case of *MF* networks the adaptation of the weights is done as in the original *MF* network.

$$W^{net}(t+1) = W^{net}(t) - \eta \frac{\partial Error_{net}}{\partial W^{net}}(t) - \alpha \cdot \left(W^{net}(t) - W^{net}(t-1)\right) \tag{B.1}$$

For the weights between the hidden and output layer, its adjust is given by:

$$\frac{\partial Error_{net}}{\partial W^{net}}(t) = \frac{\partial Error_{net}}{\partial who^{net}_{j,k}}(t) = -\delta^{net}_{k} \cdot ho^{net}_{j} \tag{B.2}$$

The adaptation of the weights between the input and hidden layer is calculated with:

$$\frac{\partial Error}{\partial W^{net}}(t) = \frac{\partial Error}{\partial wih^{net}_{i,j}}(t) = -\left(ho^{net}_{j} \cdot \left(1 - ho^{net}_{j}\right) \cdot \sum_{h=1}^{N_{output}} \left(\delta^{net}_{h} \cdot who^{net}_{j,h}\right) \right) \cdot x_i \tag{B.3}$$

The previous equations corresponds to the original equations used to train a normal *MF* network. The difference in training a original *MF* network and the training for

the networks generated with *CELS* is introduced in the parameter δ_k^{net} which uses the following equation in order to calculate its final values.

$$\delta_k^{net} = \left((d_k - y_{k,net}) + \lambda \cdot \sum_{\substack{n=1 \\ n \neq net}}^{N_{networks}} (d_k - y_{k,n}) \right) \cdot (1 - y_{k,net}) \cdot y_{k,net} \tag{B.4}$$

B.2.2 Decorrelated

In both version of *Decorrelated* a penalty term is added during training in order to penalize a network when it is correlated with the previous trained one. This new term depends on the output provided by the network. For this reason, the adaptation of the trainable parameters (only the weights for *MF* networks) has also to be modified.

In the case of *MF* networks the adaptation of the weights is also given by equations B.2 and B.3 because they correspond to the equations used to train the original network. However, δ_k^{net} has to be modified for this two methods and its values will be calculated with equation B.5 for *DECOv1* whereas *DECOv2* uses equation B.6 to calculate them.

$$\delta_k^{net} = \begin{cases} (d_k - y_k^{net}) \cdot (1 - y_k^{net}) \cdot y_k^{net} & \text{if } net = 1 \\ ((d_k - y_k^{net}) + \lambda \cdot (d_k - y_k^{net} - 1)) \cdot (1 - y_k^{net}) \cdot y_k^{net} & \text{otherwise} \end{cases} \tag{B.5}$$

$$\delta_k^{net} = \begin{cases} (d_k - y_k^{net}) \cdot (1 - y_k^{net}) \cdot y_k^{net} & \text{if } net \text{ is odd} \\ ((d_k - y_k^{net}) + \lambda \cdot (d_k - y_k^{net} - 1)) \cdot (1 - y_k^{net}) \cdot y_k^{net} & \text{otherwise} \end{cases} \tag{B.6}$$

B.2.3 EENCL - NCL

As has been mentioned *Negative Correlation Learning* was proposed by the same authors that proposed *CELS*. In fact, both learning algorithms are similar.

In this case, the authors also added a penalty term so a specific adaptation of the trainable parameters (the weights of a *MF* network) has also to be used when the networks are generated with *NCL* in *EENCL-BG* and *EENCL-LG*.

Although the penalty introduced in *NCL* differs from the term added in *CELS*, they use the same main equations to adjust the weights. However, they also differ in how δ_k^{net} is calculated. In this learning algorithm, its values are given by the following equation:

$$\delta_k^{net} = -\left((1 - \lambda) \cdot \left(y_k^{net} - d_k\right) + \lambda \cdot (\overline{y}_k - d_k)\right) \cdot \left(1 - y_k^{net}\right) \cdot y_k^{net} \tag{B.7}$$

B.3 MF Network parameters

Table B.1 shows the training parameters of the *Multilayer Feedforward* networks. Those parameters are: Number of hidden units (N_{hidden}), Adaptation step ($Step$), Momentum rate ($Momentum$) and Maximum number of iterations of *Backpropagation* ($Epochs$).

In this case, the *MF* networks are used to train the single networks, the networks of an ensemble and the level-0 classifiers in the *Stacked Generalization*.

Table B.1: MF Network parameters

Database	N_{hidden}	$Epochs$	$Step$	$Momentum$
aritm	9	2500	0.1	0.05
bala	20	5000	0.1	0.05
band	23	5000	0.1	0.05
bupa	11	8500	0.1	0.05
cred	15	8500	0.1	0.05
derma	4	1000	0.1	0.05
ecoli	5	10000	0.1	0.05
flare	11	10000	0.6	0.05
glas	3	4000	0.1	0.05
hear	2	5000	0.1	0.05
img	14	1500	0.4	0.05
ionos	8	5000	0.1	0.05
mok1	6	3000	0.1	0.05
mok2	20	7000	0.1	0.05
pima	14	10000	0.4	0.05
survi	9	20000	0.1	0.2
vote	1	2500	0.1	0.05
vowel	15	4000	0.2	0.2
wdbc	6	4000	0.1	0.05

Those parameters can be considered almost optimal for the datasets. Moreover, the parameters have been set after a deep trial and error procedure using the validation sets in which some different network configurations are tested.

B.4 RBF Network parameters

We show the training parameters of the *Radial Basis Functions* networks in table B.2. Now, those parameters are: Number of Clusters ($N_{clusters}$), Width of Clusters ($Width$), Adaptation step ($Step$), and Maximum number of iterations ($Epochs$).

As in the case of *MF*, the parameters of the *RBF* networks are used to train the single networks and the expert networks of the multi-net systems studied.

Finally, the proposed parameters can also be considered almost optimal for the datasets. They have also been set after a deep trial and error procedure similar to the one done for the *MF* network (using different networks configurations and the validation set).

Table B.2: RBF Network parameters

Dataset	$N_{clusters}$	*Epochs*	*Step*	*Width*
aritm	50	8000	0.01	4
bala	60	6000	0.005	0.6
band	40	10000	0.01	1
bupa	40	8000	0.01	0.4
cred	30	40000	0.01	2
derma	90	18000	0.01	1.8
ecoli	70	400000	0.001	0.4
flare	40	100000	0.001	2.2
glas	110	20000	0.01	0.4
hear	20	25000	0.01	2
img	340	40000	0.001	0.4
ionos	150	8000	0.01	2.6
mok1	30	20000	0.01	0.8
mok2	45	60000	0.01	0.6
pima	60	15000	0.005	2.4
survi	10	40000	0.0001	0.4
vote	5	5000	0.01	1.8
vowel	280	18000	0.005	0.4
wdbc	60	70000	0.001	1.2

B.5 Parameters of the ensemble methods

In chapters 4 and 5, some ensemble methods were studied. Some of them requires specific parameters which are case-dependent. Usually, the value of those parameters depends on the dataset and on the number of networks of the ensemble. In this section, the parameters of the ensemble methods are shown.

All the parameters have been set after a deep trial and error procedure using the validation sets and some different values have been tested. The value with we obtain the best results, highest performance and lowest error, is set to the parameter.

B.5.1 Adaptive Training Algorithm for Ensembles

In both versions of *ATA*, *ATA-LE* and *ATA-BE*, we have to set the value for the number of iterations of the training procedure, N_{ite}. The values for this parameter are introduced in tables B.3 (*ATA-BE*) and B.4 (*ATA-LE*). Other variables and constants were introduced when the ensemble was described but they are reproduced also here.

$$P = \text{any constant (usually between 10\% to 40\%)} \tag{B.8}$$

$$\Gamma = 0.0001 \tag{B.9}$$

Table B.3: Adaptive Training Algorithm - Best Epoch (*ATA-BE*) N_{ite} values

Database	3-*net*	9-*net*	20-*net*	40-*net*
aritm	100	10	1000	500
bala	2500	2500	2500	2500
band	100	500	100	2500
bupa	1000	500	500	1000
cred	100	2500	10	10
derma	500	500	1000	2500
ecoli	2500	1000	1000	500
flare	10	2500	1000	500
glas	1000	2500	2500	2500
hear	500	100	10	10
img	2500	2500	2500	2500
ionos	1000	2500	1000	2500
mok1	500	1000	500	500
mok2	2500	2500	2500	1000
pima	100	1000	100	100
survi	1000	100	1000	1000
vote	10	10	10	10
vowel	1000	2500	2500	2500
wdbc	500	10	10	10

Table B.4: Adaptive Training Algorithm - Last Epoch (*ATA-BE*) N_{ite} values

Database	3-*net*	9-*net*	20-*net*	40-*net*
aritm	500	1000	500	500
bala	1000	2500	2500	1000
band	500	2500	100	100
bupa	500	500	500	1000
cred	10	500	500	2500
derma	500	500	500	1000
ecoli	500	1000	500	500
flare	100	100	2500	10
glas	1000	1000	2500	1000
hear	10	10	10	10
img	2500	2500	2500	2500
ionos	2500	2500	1000	500
mok1	1000	2500	500	500
mok2	500	1000	500	1000
pima	100	10	10	10
survi	1000	100	100	2500
vote	10	100	100	10
vowel	1000	2500	2500	2500
wdbc	100	10	10	100

B.5.2 Cooperative Ensemble Learning System

In *CELS*, only the weight of the penalty term in the final error equation, denoted with λ, has to be optimized.

Table B.5: Cooperative Ensemble Learning System - λ values

Database	3-*net*	9-*net*	20-*net*	40-*net*
aritm	0.5	0.25	0.25	0.1
bala	0.1	0.1	0.1	0.25
band	0.5	0.25	0.1	0.25
bupa	0.75	0.1	0.1	0.1
cred	0.1	0.75	0.5	0.5
derma	0.1	0.1	0.25	0.1
ecoli	0.5	0.1	0.1	0.25
flare	0.5	0.1	0.1	0.1
glas	0.25	0.1	0.25	0.1
hear	0.5	0.25	1.0	0.75
img	0.5	0.1	0.25	0.1
ionos	1.0	0.75	0.25	0.1
mok1	0.25	0.1	0.1	0.1
mok2	0.1	0.1	0.1	0.1
pima	0.75	0.75	0.25	0.1
survi	0.5	0.25	0.75	0.25
vote	0.5	0.75	0.1	0.1
vowel	1.0	0.1	0.1	0.1
wdbc	0.1	1.0	1.0	0.75

B.5.3 Evolutionary Ensemble with NCL

In *NCL*, the weight of the penalty (λ) has also to be optimized. Moreover, the number of iterations and generations (N_{ite} and $N_{generations}$) have also to be set.

Table B.6: Evolutionary Ensemble with NCL - λ values

Database	3-*net*	9-*net*	20-*net*	40-*net*
aritm	0.5	0.75	0.5	0.25
bala	0.75	0.75	0.5	0.25
band	0.75	0.75	0.25	0.25
bupa	0.75	0.5	0.75	0.5
cred	0.5	0.75	0.25	0.75
derma	0.5	0.75	0.5	0.5
ecoli	0.75	0.5	0.5	0.5
flare	0.75	0.25	0.75	0.5
glas	0.5	0.75	0.25	0.25
hear	0.75	0.25	0.25	0.5
img	0.25	0.5	0.25	0.75
ionos	0.25	0.25	0.5	0.5
mok1	0.5	0.75	0.75	0.75
mok2	0.75	0.25	0.5	0.75
pima	0.5	0.5	0.75	0.75
survi	0.25	0.5	0.75	0.75
vote	0.5	0.75	0.75	0.75
vowel	0.5	0.25	0.25	0.75
wdbc	0.5	0.75	0.25	0.75

Table B.7: Evolutionary Ensemble with NCL - N_{ite} values

Database	3-*net*	9-*net*	20-*net*	40-*net*
aritm	10	25	10	20
bala	5	15	5	15
band	5	15	10	25
bupa	5	25	15	25
cred	5	5	5	5
derma	15	5	15	25
ecoli	10	10	15	25
flare	5	15	5	10
glas	25	25	20	25
hear	5	10	5	10
img	10	10	15	20
ionos	25	5	25	10
mok1	15	20	25	15
mok2	10	15	25	10
pima	5	10	5	10
survi	5	15	20	15
vote	15	10	5	10
vowel	5	10	20	25
wdbc	15	15	15	10

Table B.8: Evolutionary Ensemble with NCL - $N_{generations}$ values

Database	3-*net*	9-*net*	20-*net*	40-*net*
aritm	250	500	750	750
bala	1000	750	1000	1000
band	500	90	100	250
bupa	750	750	700	250
cred	250	300	400	300
derma	300	750	250	400
ecoli	750	750	750	750
flare	1000	500	500	250
glas	1000	800	750	1000
hear	250	500	750	750
img	1000	1000	500	1000
ionos	350	300	250	500
mok1	250	100	250	200
mok2	1000	500	200	750
pima	150	200	200	300
survi	750	750	500	500
vote	500	400	500	750
vowel	750	1000	500	500
wdbc	500	500	750	500

B.5.4 Decorrelated

The variable λ (the weight of the penalty term in the final error equation) had to be optimized for each case (database and ensemble size) as was done for *CELS* and *NCL*.

Table B.9: Decorrelated v1 - λ values

Database	3-*net*	9-*net*	20-*net*	40-*net*
aritm	0.2	0.6	1.0	0.6
bala	1.0	1.0	1.0	1.0
band	1.0	1.0	1.0	1.0
bupa	0.8	0.8	0.8	0.8
cred	0.2	0.2	0.2	0.2
derma	0.2	0.2	0.8	0.2
ecoli	0.2	0.4	0.2	0.2
flare	0.4	0.2	0.2	0.6
glas	0.6	0.6	0.6	0.6
hear	0.6	0.6	0.6	0.6
img	1.0	1.0	0.8	1.0
ionos	0.2	0.6	0.2	1.0
mok1	0.6	0.6	0.6	0.6
mok2	0.4	0.4	0.4	0.4
pima	0.2	0.2	0.2	0.8
survi	0.2	0.2	0.8	0.2
vote	0.4	0.4	0.4	0.4
vowel	0.6	0.8	0.8	0.6
wdbc	0.2	0.2	0.2	0.2

Table B.10: Decorrelated v2 - λ values

Database	3-*net*	9-*net*	20-*net*	40-*net*
aritm	0.2	0.2	0.2	1
bala	0.6	0.6	0.6	0.6
band	0.6	0.6	0.6	0.6
bupa	0.2	0.2	0.2	0.2
cred	0.2	0.2	0.2	0.2
derma	0.2	0.2	0.2	0.2
ecoli	0.2	0.2	0.2	0.2
flare	0.8	0.2	0.8	0.2
glas	0.6	0.6	0.6	0.6
hear	0.6	0.6	0.6	0.6
img	0.2	1.0	1.0	0.4
ionos	0.2	0.2	0.2	0.2
mok1	0.6	0.6	0.6	0.6
mok2	0.8	0.8	0.8	0.8
pima	0.2	0.2	0.8	1
survi	0.2	0.2	0.2	0.8
vote	0.6	0.6	0.6	0.6
vowel	0.2	0.2	0.4	0.4
wdbc	0.2	0.2	0.2	0.2

B.5.5 Observational Learning Algorithm

OLA only requires to set the variance, denoted by σ, of the normal distribution used to add the error to the patterns from the virtual set.

Table B.11: Observational Learning Algorithm - σ values

Database	3-*net*	9-*net*	20-*net*	40-*net*
aritm	0.3	0.3	0.2	0.2
bala	0.5	0.5	0.6	0.6
band	0.5	0.5	0.6	0.6
bupa	0.5	0.6	0.3	0.2
cred	0.6	0.6	0.5	0.5
derma	0.2	0.6	0.6	0.6
ecoli	0.5	0.4	0.5	0.4
flare	0.2	0.5	0.3	0.3
glas	0.3	0.2	0.6	0.6
hear	0.3	0.4	0.4	0.6
img	0.6	0.6	0.6	0.5
ionos	0.6	0.3	0.6	0.6
mok1	0.2	0.2	0.6	0.6
mok2	0.6	0.6	0.4	0.3
pima	0.4	0.4	0.5	0.5
survi	0.5	0.6	0.5	0.5
vote	0.3	0.3	0.2	0.2
vowel	0.6	0.6	0.6	0.6
wdbc	0.3	0.4	0.3	0.5

B.5.6 Bagging with Noise

In this ensemble, only the variance (σ) of the normal distribution used to add the error to the training patterns had to be set.

Table B.12: *Bagging with Noise* - σ values

Database	3-*net*	9-*net*	20-*net*	40-*net*
aritm	0.1	0.1	0.1	0.1
bala	0.1	0.1	0.1	0.1
band	0.2	0.2	0.2	0.2
bupa	0.1	0.1	0.1	0.1
cred	0.7	0.7	0.7	0.7
derma	0.1	0.1	0.1	0.1
ecoli	0.1	0.1	0.1	0.1
flare	0.1	0.1	0.1	0.1
glas	0.2	0.2	0.2	0.2
hear	0.4	0.4	0.4	0.4
img	0.1	0.1	0.1	0.1
ionos	0.1	0.1	0.1	0.1
mok1	0.1	0.1	0.1	0.1
mok2	0.1	0.1	0.1	0.1
pima	0.1	0.1	0.1	0.1
survi	0.2	0.2	0.2	0.2
vote	0.1	0.1	0.1	0.1
vowel	0.1	0.1	0.1	0.1
wdbc	0.2	0.1	0.1	0.1

B.5.7 Weighted-Conservative Boosting

In the ensembles based on *WCB*, the weight (denoted by α) of the equation used to update the sampling distribution had to be optimized.

Table B.13: *Weighted Conservative Boosting* - α value

Database	3-*net*	9-*net*	20-*net*	40-*net*
aritm	1.25	0.8	0.5	1.25
bala	5	2.5	0.7	0.7
band	0.9	1.5	1.5	0.9
bupa	0.5	1	0.7	1
cred	0.3	0.5	1.25	0.6
derma	0.5	0.6	2	0.9
ecoli	0.5	0.5	0.8	1
flare	0.3	2	1.25	1.25
glas	1.25	0.8	1	1.25
hear	1	1	4	1
img	3	2	0.6	0.6
ionos	0.5	2.5	0.6	0.6
mok1	0.6	0.6	0.6	1.5
mok2	0.6	1.75	1.75	1.75
pima	1	0.7	2	1.25
survi	0.3	0.6	0.6	0.7
vote	0.6	0.01	0.1	0.1
vowel	1.75	1.75	1.75	1.25
wdbc	0.9	1.25	0.05	1.25

Table B.14: *Cross-Validated WCB* - α value

	CVCv2WCB				**CVCv3WCB**			
dataset	**3-net**	**9-net**	**20-net**	**40-net**	**3-net**	**9-net**	**20-net**	**40-net**
aritm	0.6	0.8	0.8	0.8	0.4	0.9	0.7	0.9
bala	0.4	0.5	0.4	0.4	1.25	0.7	0.5	0.4
band	0.9	1.5	1.0	0.9	1.75	0.8	0.8	1.0
bupa	0.3	0.6	0.7	1.25	0.6	0.8	0.7	0.8
cred	1.75	1.5	0.7	0.8	0.5	1.75	1.5	1.0
derma	0.1	0.5	0.5	0.7	0.1	0.6	0.3	0.6
ecoli	0.5	1.25	0.6	2.0	0.8	1.25	1.0	1.25
flare	1.5	1.0	1.0	1.0	0.3	0.7	0.8	1.0
glas	0.9	0.5	0.7	0.5	1.25	0.9	0.5	0.5
hear	1.25	1.5	0.9	1.25	2.5	1.0	0.8	0.9
img	1.25	1.0	0.6	0.8	0.9	0.9	0.6	0.5
ionos	0.4	0.7	0.6	0.8	3.0	0.6	0.6	0.6
mok1	0.3	0.05	0.3	2.5	0.1	0.6	0.6	1.75
mok2	1.25	0.5	0.8	0.9	0.9	0.9	0.9	0.9
pima	0.6	2.5	0.9	1.5	2.0	2.5	2.5	1.5
survi	0.8	0.8	1.25	1.0	0.7	0.5	4.0	0.7
vote	0.4	0.1	1.25	1.0	1.25	0.8	0.6	0.3
vowel	0.6	0.9	0.6	0.5	0.8	1.0	0.6	0.5
wdbc	1.25	1.25	0.7	0.7	5.0	0.7	0.7	0.7

B.6 Parameters of the ensemble combiners

B.6.1 Choquet Integral

There is a specific parameter in *choquet*, the desired sum of densities denoted by d_s, which has to be empirically determined as can be seen in equation 6.29. In the literature there is not any constraint associated to this parameter. After performing some experiments we noticed that the combiner did not work properly if $g^{net} \geq 1$ because λ obtained not allowed values after solving equation 6.33. Furthermore, we also noticed that the behavior of the combiner was better if the density of the best network was close to one.

Finally, after performing some tests we decided to apply equation B.10 in order to get the value of d_s. In this way, the combiner works properly and provides good results. This equation assigns the most appropriate value according to the case. Moreover, the values of the densities obtained with this equation are close to the values showed in reference [106].

$$d_s = 0.95 \cdot \frac{\sum_{net=1}^{N_{networks}} p_{net}}{\max_{net=1}^{N_{networks}} p_{net}} \tag{B.10}$$

In the optimal case, the density of the best network should have been set to $g^{net} = 1$. In the implementation of the combiner $g^{net} \leq 0.95$ as can be seen in the previous equation. The value 0.95 had to be added to get the stability of the system so because not allowed values were assigned to λ when g^{net} was higher than 0.95.

B.6.2 Feature-Based Combiners

Similarly to *choquet*, there is a specific parameter in *choquet.ddd* and in *wave.ddw*, the desired sum of densities denoted by d_s^{reg}, which has to be empirically determined. In these methods, the desired sum depends on the densities and on the region of the input data space. The value of this parameter is given by:

$$d_s^{reg} = 0.95 \cdot \frac{\sum_{net=1}^{N_{networks}} p_{reg}^{net}}{\max_{net=1}^{N_{networks}} p_{reg}^{net}} \tag{B.11}$$

Moreover, some parameters related to the algorithm *FSCL* have to been set:

$$N_{regions} = 5 \tag{B.12}$$

$$N_{ite} = 500 \tag{B.13}$$

$$\beta = 0.1 \tag{B.14}$$

$$\gamma = 0.05 \tag{B.15}$$

$$c_{ite} = 1 - 0.9 \cdot \frac{ite}{N_{ite}} \tag{B.16}$$

B.6.3 BADD Defuzzyfication Strategy

In *BADD*, the parameters related to the combiner and the parameters of the basic algorithm used to generate the reference points have to been set as follows:

$$P = 2 \quad \text{(B.17)} \qquad N_{ite} = 250 \quad \text{(B.19)}$$

$$\delta = 3 \quad \text{(B.18)} \qquad c_{ite} = 1 - 0.9 \cdot \frac{ite}{N_{ite}} \quad \text{(B.20)}$$

B.6.4 Combination by Zimmermann's Operator

There are three parameters which requires a initial value in the combiner based on the *Zimmermann's Compensatory Operator*. Those parameters are a, b and d. After performing some experiments, it has been noticed that the combiner runs better when:

$$a \in [-0.05, 0.05] \quad \text{(B.21)} \qquad b \in [-0.05, 0.05] \quad \text{(B.22)} \qquad d_{net} = perf_{net}^{val} \quad \text{(B.23)}$$

Where $perf_{net}^{val}$ is the performance of the network *net* on the validation set.

Moreover, *step* and N_{ite} are the specific parameters related to the gradient descent method applied to the optimization procedure of the combiner.

$$step = 0.1 \quad \text{(B.24)} \qquad N_{ite} = 200 \quad \text{(B.25)}$$

B.6.5 Dynamically Averaged Networks v2

In *Dynamically Averaged Networks v2* there is only one specific parameter that required to be set γ, equation 6.6, which has been set to value 0.05.

B.7 Parameters of *Stacked Generalization*

All the parameters related to the original implementations and proposed combiners based on *Stacked Generalization* are introduced in this section. These parameters correspond to the training parameters of the combination network.

Concretely, tables B.15 to B.16 show the parameters of the original *stacked* implementations. Moreover, tables B.17 to B.24 show the parameters of the *stacked* combiners when they are applied to ensembles of *MF* networks. Finally, tables B.25 to B.32 show the parameters of the *stacked* combiners when they are applied to *RBF* ensembles.

In all these tables, the parameters of the *MF* network are introduced when they are applied as combination network (level-1). For the expert networks (level-0), we have used the training parameters specified for the single *MF* and *RBF* network introduced in tables B.1 and B.2.

Table B.15: Parameters of *Original Stacked - Ghorbani & Owrangh's* implementation

	3-expert				9-expert				20-expert				40-expert			
dataset	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**
aritm	0.40	0.10	5	6000	0.40	0.10	29	4500	0.40	0.20	16	500	0.05	0.05	2	500
bala	0.10	0.05	29	1500	0.10	0.20	28	1500	0.20	0.05	16	500	0.05	0.20	5	7500
band	0.20	0.05	24	7500	0.20	0.20	4	7500	0.40	0.10	2	500	0.40	0.01	9	4000
bupa	0.40	0.20	2	7500	0.40	0.20	7	2500	0.40	0.01	27	6000	0.40	0.01	14	2500
cred	0.10	0.10	18	500	0.10	0.20	28	500	0.40	0.01	30	500	0.40	0.20	30	500
derma	0.10	0.05	25	1500	0.10	0.01	30	1500	0.10	0.05	5	1750	0.40	0.20	6	500
ecoli	0.20	0.10	9	7500	0.20	0.10	12	4000	0.40	0.01	30	1500	0.40	0.05	4	1500
flare	0.40	0.20	2	1500	0.40	0.20	15	7500	0.10	0.05	2	500	0.40	0.05	28	4000
glas	0.40	0.20	29	7500	0.40	0.10	23	7500	0.40	0.20	28	7500	0.40	0.20	30	7500
hear	0.20	0.05	9	7500	0.20	0.20	29	500	0.40	0.10	6	3000	0.05	0.05	2	1750
img	0.10	0.05	8	6500	0.10	0.10	13	3000	0.05	0.20	3	4000	0.20	0.05	4	3000
ionos	0.40	0.10	3	500	0.40	0.01	25	500	0.40	0.20	29	500	0.40	0.20	30	1500
mok1	0.40	0.20	29	7500	0.40	0.20	30	7500	0.40	0.20	30	7500	0.40	0.20	28	7500
mok2	0.40	0.10	2	1750	0.40	0.05	23	500	0.05	0.01	7	500	0.40	0.20	27	500
pima	0.20	0.05	28	500	0.20	0.10	28	500	0.20	0.20	10	7500	0.40	0.10	7	7500
survi	0.10	0.05	30	4500	0.10	0.20	11	1750	0.40	0.05	2	500	0.40	0.05	26	7500
vote	0.05	0.01	8	2500	0.05	0.20	5	4000	0.40	0.10	3	2500	0.40	0.10	3	1500
vowel	0.10	0.20	14	7500	0.10	0.10	6	3000	0.05	0.05	6	3000	0.10	0.05	8	3000
wdbc	0.20	0.01	25	500	0.20	0.10	29	500	0.40	0.20	4	5000	0.20	0.10	30	500

Table B.16: Parameters of *Original Stacked - Ting & Witten's* implementation

	3-expert				9-expert				20-expert				40-expert			
dataset	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**
aritm	0.05	0.20	4	2500	0.05	0.20	22	4000	0.40	0.20	3	1500	0.40	0.01	20	3000
bala	0.10	0.20	3	500	0.10	0.05	6	4000	0.05	0.01	10	3000	0.40	0.10	2	2500
band	0.40	0.20	2	1500	0.40	0.20	9	1500	0.40	0.20	26	1500	0.20	0.20	28	1500
bupa	0.20	0.20	30	500	0.20	0.20	15	500	0.40	0.05	22	3000	0.40	0.01	3	2500
cred	0.40	0.10	2	500	0.40	0.10	2	500	0.40	0.01	9	500	0.40	0.01	9	500
derma	0.10	0.20	15	4500	0.10	0.10	12	7500	0.40	0.20	5	7500	0.05	0.05	11	7500
ecoli	0.05	0.10	3	7500	0.05	0.05	15	1500	0.40	0.10	21	1500	0.40	0.01	7	1500
flare	0.10	0.20	20	7500	0.10	0.20	23	6500	0.05	0.01	23	4500	0.05	0.05	26	4000
glas	0.40	0.20	3	7500	0.40	0.20	28	7500	0.40	0.20	16	7500	0.40	0.20	29	7500
hear	0.05	0.20	26	500	0.05	0.10	30	500	0.40	0.01	6	4000	0.40	0.20	25	6500
img	0.40	0.05	10	6500	0.40	0.20	7	1750	0.10	0.05	10	4000	0.20	0.20	3	2500
ionos	0.05	0.05	30	500	0.05	0.20	10	500	0.40	0.20	17	6000	0.40	0.01	29	6500
mok1	0.40	0.20	29	7500	0.40	0.20	30	7500	0.40	0.20	29	7500	0.40	0.20	30	7500
mok2	0.40	0.20	30	500	0.40	0.01	7	2500	0.40	0.10	9	3000	0.10	0.05	5	7500
pima	0.10	0.20	6	2500	0.10	0.05	26	1500	0.10	0.20	10	1500	0.40	0.20	2	500
survi	0.40	0.20	24	500	0.40	0.01	26	7500	0.40	0.05	6	7500	0.40	0.10	10	7500
vote	0.40	0.20	26	7500	0.40	0.20	17	6500	0.40	0.20	20	7500	0.40	0.05	27	6500
vowel	0.10	0.20	8	6000	0.10	0.10	8	1500	0.10	0.20	10	3000	0.40	0.20	10	7500
wdbc	0.40	0.05	26	5000	0.40	0.01	9	4000	0.40	0.10	6	1750	0.40	0.20	11	1500

Table B.17: Parameters of the combiner *STC* on *MFSE* experts

	3-expert				9-expert				20-expert				40-expert			
dataset	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**
aritm	0.10	0.05	3	10000	0.10	0.05	20	500	0.10	0.05	1	100	0.10	0.05	5	100
bala	0.10	0.05	5	100	0.10	0.05	5	100	0.10	0.05	5	100	0.10	0.05	5	100
band	0.10	0.05	5	100	0.10	0.05	5	100	0.10	0.05	5	100	0.10	0.05	5	100
bupa	0.10	0.05	5	100	0.10	0.05	5	100	0.10	0.05	5	100	0.10	0.05	5	100
cred	0.10	0.05	5	100	0.10	0.05	5	100	0.10	0.05	5	100	0.10	0.05	5	100
derma	0.10	0.05	5	10000	0.10	0.05	5	10000	0.10	0.05	5	2500	0.10	0.05	5	1000
ecoli	0.10	0.05	10	2500	0.10	0.05	10	2500	0.10	0.05	7	2500	0.10	0.05	5	500
flare	0.10	0.05	1	2500	0.10	0.05	1	2500	0.10	0.05	5	1500	0.10	0.05	6	2500
glas	0.10	0.05	6	2500	0.10	0.05	6	2500	0.10	0.05	6	2500	0.10	0.05	6	2500
hear	0.10	0.05	6	2500	0.10	0.05	6	2500	0.10	0.05	6	2500	0.10	0.05	6	2500
img	0.10	0.05	8	2500	0.10	0.05	6	2500	0.10	0.05	7	2500	0.10	0.05	11	7500
ionos	0.10	0.05	7	10000	0.10	0.05	1	10000	0.10	0.05	5	10000	0.10	0.05	5	10000
mok1	0.10	0.05	5	10000	0.10	0.05	5	10000	0.10	0.05	5	10000	0.10	0.05	5	10000
mok2	0.10	0.05	5	10000	0.10	0.05	5	10000	0.10	0.05	5	10000	0.10	0.05	5	10000
pima	0.10	0.05	5	500	0.10	0.05	5	500	0.10	0.05	7	250	0.10	0.05	10	5000
survi	0.10	0.05	7	250	0.10	0.05	1	250	0.10	0.05	25	250	0.10	0.05	25	250
vote	0.10	0.05	25	250	0.10	0.05	25	250	0.10	0.05	25	250	0.10	0.05	25	250
vowel	0.10	0.05	19	10000	0.10	0.05	6	7500	0.10	0.05	20	500	0.10	0.05	10	5000
wdbc	0.10	0.05	1	250	0.10	0.05	1	250	0.10	0.05	1	250	0.10	0.05	1	500

Table B.18: Parameters of the combiner *STCP* on *MFSE* experts

	3-expert				9-expert				20-expert				40-expert			
dataset	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**
aritm	0.10	0.05	4	2500	0.10	0.05	6	1500	0.10	0.05	17	1500	0.10	0.05	5	1500
bala	0.10	0.05	5	1500	0.10	0.05	5	1500	0.10	0.05	5	1500	0.10	0.05	5	1500
band	0.10	0.05	5	1500	0.10	0.05	5	1500	0.10	0.05	5	1500	0.10	0.05	5	1500
bupa	0.10	0.05	5	1500	0.10	0.05	5	1500	0.10	0.05	5	1500	0.10	0.05	5	1500
cred	0.10	0.05	5	1500	0.10	0.05	5	1500	0.10	0.05	5	1500	0.10	0.05	5	1500
derma	0.10	0.05	10	2500	0.10	0.05	5	7500	0.10	0.05	6	5000	0.10	0.05	5	5000
ecoli	0.10	0.05	10	2500	0.10	0.05	5	2500	0.10	0.05	10	2500	0.10	0.05	10	2500
flare	0.10	0.05	1	2500	0.10	0.05	5	5000	0.10	0.05	25	4000	0.10	0.05	24	7500
glas	0.10	0.05	24	7500	0.10	0.05	24	7500	0.10	0.05	24	7500	0.10	0.05	24	7500
hear	0.10	0.05	24	7500	0.10	0.05	24	7500	0.10	0.05	24	7500	0.10	0.05	24	7500
img	0.10	0.05	10	5000	0.10	0.05	7	5000	0.10	0.05	7	5000	0.10	0.05	5	5000
ionos	0.10	0.05	1	10000	0.10	0.05	1	10000	0.10	0.05	4	10000	0.10	0.05	5	10000
mok1	0.10	0.05	5	10000	0.10	0.05	5	10000	0.10	0.05	5	10000	0.10	0.05	5	10000
mok2	0.10	0.05	5	10000	0.10	0.05	5	10000	0.10	0.05	5	10000	0.10	0.05	5	10000
pima	0.10	0.05	10	250	0.10	0.05	7	250	0.10	0.05	1	250	0.10	0.05	3	500
survi	0.10	0.05	5	2500	0.10	0.05	8	750	0.10	0.05	22	1500	0.10	0.05	11	1500
vote	0.10	0.05	11	1500	0.10	0.05	11	1500	0.10	0.05	11	1500	0.10	0.05	11	1500
vowel	0.10	0.05	30	2500	0.10	0.05	13	5000	0.10	0.05	10	2500	0.10	0.05	7	5000
wdbc	0.10	0.05	5	2500	0.10	0.05	1	2500	0.10	0.05	1	2500	0.10	0.05	1	2500

Table B.19: Parameters of the combiner *STC* on *Adaboost* experts

	3-expert				9-expert				20-expert				40-expert			
dataset	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**
aritm	0.40	0.10	30	500	0.40	0.05	24	500	0.05	0.01	25	500	0.20	0.10	2	500
bala	0.40	0.20	3	1500	0.40	0.10	2	500	0.40	0.10	26	500	0.10	0.20	3	500
band	0.40	0.10	16	6000	0.40	0.20	12	500	0.40	0.05	2	500	0.05	0.01	5	500
bupa	0.05	0.01	25	1500	0.05	0.10	13	5000	0.40	0.10	17	4500	0.40	0.20	10	4000
cred	0.05	0.01	7	500	0.05	0.10	10	4000	0.40	0.10	28	4000	0.40	0.20	13	3000
derma	0.40	0.20	4	2500	0.40	0.20	3	4000	0.20	0.20	3	6000	0.10	0.10	5	7500
ecoli	0.10	0.20	3	1750	0.10	0.20	5	1500	0.40	0.20	25	500	0.40	0.10	19	1500
flare	0.10	0.05	20	7500	0.10	0.20	11	7500	0.10	0.05	16	2500	0.40	0.20	2	500
glas	0.40	0.01	4	7500	0.40	0.10	4	7500	0.40	0.20	11	7500	0.40	0.20	13	7500
hear	0.40	0.01	6	4000	0.40	0.01	5	7500	0.20	0.01	8	4500	0.40	0.10	30	4000
img	0.40	0.05	3	1500	0.40	0.01	2	4000	0.40	0.05	2	4500	0.20	0.01	2	1750
ionos	0.40	0.01	3	500	0.40	0.10	3	500	0.40	0.20	2	500	0.05	0.10	23	500
mok1	0.40	0.20	29	7500	0.40	0.20	27	7500	0.40	0.20	30	7500	0.40	0.20	30	7500
mok2	0.10	0.01	2	500	0.10	0.05	2	500	0.10	0.20	26	500	0.20	0.20	27	500
pima	0.05	0.10	28	500	0.05	0.10	7	500	0.05	0.20	9	500	0.05	0.01	10	500
survi	0.40	0.20	2	1500	0.40	0.20	6	7500	0.40	0.20	2	1500	0.40	0.05	3	1500
vote	0.20	0.20	6	500	0.20	0.01	29	500	0.10	0.01	22	500	0.05	0.10	24	500
vowel	0.10	0.05	14	1500	0.10	0.10	24	1500	0.10	0.05	6	6500	0.10	0.05	11	7500
wdbc	0.40	0.20	29	500	0.40	0.20	11	3000	0.05	0.20	15	4500	0.05	0.01	28	500

Table B.20: Parameters of the combiner *STCP* on *Adaboost* experts

	3-expert				9-expert				20-expert				40-expert			
dataset	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**
aritm	0.40	0.05	15	1750	0.40	0.20	14	500	0.40	0.20	5	500	0.20	0.20	24	500
bala	0.40	0.01	4	7500	0.40	0.01	3	1500	0.40	0.05	30	500	0.10	0.10	2	1500
band	0.40	0.05	11	1750	0.40	0.20	14	500	0.40	0.01	3	500	0.05	0.20	2	500
bupa	0.05	0.01	10	7500	0.05	0.20	25	5000	0.40	0.20	25	6000	0.40	0.20	15	6500
cred	0.40	0.20	29	1500	0.40	0.10	14	500	0.40	0.10	2	500	0.05	0.01	13	500
derma	0.40	0.20	3	7500	0.40	0.05	3	7500	0.05	0.10	3	7500	0.10	0.01	3	2500
ecoli	0.20	0.05	4	1750	0.20	0.10	3	6500	0.40	0.20	26	500	0.40	0.01	16	1500
flare	0.10	0.01	2	7500	0.10	0.10	30	500	0.20	0.20	30	500	0.40	0.05	2	500
glas	0.40	0.10	5	7500	0.40	0.10	3	7500	0.40	0.20	14	7500	0.40	0.20	17	7500
hear	0.20	0.20	2	3000	0.20	0.05	19	500	0.40	0.01	30	2500	0.20	0.20	2	7500
img	0.40	0.20	5	500	0.40	0.05	3	1500	0.40	0.05	2	2500	0.05	0.05	3	7500
ionos	0.05	0.05	29	500	0.05	0.05	5	500	0.40	0.05	2	500	0.05	0.01	19	500
mok1	0.40	0.20	27	7500	0.40	0.20	28	7500	0.40	0.20	30	7500	0.40	0.20	28	7500
mok2	0.05	0.05	2	500	0.05	0.20	5	500	0.05	0.10	3	500	0.10	0.01	27	500
pima	0.05	0.20	30	500	0.05	0.10	13	500	0.05	0.20	14	500	0.05	0.05	4	500
survi	0.40	0.05	9	3000	0.40	0.20	2	1500	0.40	0.01	8	7500	0.40	0.20	2	1500
vote	0.20	0.05	27	500	0.20	0.05	30	500	0.10	0.05	25	500	0.05	0.20	28	500
vowel	0.20	0.20	4	2500	0.20	0.01	11	7500	0.05	0.05	4	7500	0.05	0.20	12	7500
wdbc	0.40	0.20	29	500	0.40	0.20	14	7500	0.10	0.10	13	2500	0.05	0.10	30	500

Table B.21: Parameters of the combiner *STC* on *Conservative* experts

	3-expert				9-expert				20-expert				40-expert			
dataset	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**
aritm	0.05	0.10	3	1500	0.05	0.20	18	6000	0.20	0.20	9	500	0.40	0.01	23	500
bala	0.05	0.01	2	7500	0.05	0.05	28	500	0.40	0.01	3	500	0.10	0.20	3	1500
band	0.20	0.05	22	6500	0.20	0.10	9	500	0.40	0.20	27	1500	0.05	0.10	16	500
bupa	0.40	0.05	21	3000	0.40	0.20	15	3000	0.40	0.05	3	500	0.40	0.20	10	4000
cred	0.05	0.05	30	500	0.05	0.05	25	2500	0.20	0.20	2	500	0.05	0.05	18	500
derma	0.10	0.20	2	7500	0.10	0.10	2	7500	0.05	0.20	4	1750	0.05	0.20	5	7500
ecoli	0.40	0.01	7	1750	0.40	0.20	8	1500	0.40	0.20	29	500	0.40	0.20	30	500
flare	0.10	0.10	23	7500	0.10	0.05	7	6000	0.40	0.05	14	3000	0.05	0.01	5	1500
glas	0.40	0.20	28	7500	0.40	0.01	12	7500	0.40	0.05	11	7500	0.40	0.20	18	7500
hear	0.40	0.05	30	5000	0.40	0.05	14	2500	0.10	0.20	4	500	0.40	0.01	3	2500
img	0.05	0.10	15	500	0.05	0.01	4	1500	0.40	0.10	6	500	0.40	0.10	4	500
ionos	0.40	0.20	2	500	0.40	0.10	27	500	0.05	0.20	18	500	0.05	0.05	19	500
mok1	0.40	0.20	29	7500	0.40	0.20	27	7500	0.40	0.20	30	7500	0.40	0.20	30	7500
mok2	0.40	0.20	23	7500	0.40	0.20	24	500	0.05	0.20	2	500	0.40	0.20	16	500
pima	0.40	0.05	5	500	0.40	0.20	7	500	0.05	0.01	29	500	0.20	0.05	4	500
survi	0.40	0.20	3	500	0.40	0.20	10	500	0.05	0.01	10	5000	0.05	0.05	10	6000
vote	0.40	0.10	19	7500	0.40	0.10	3	3000	0.20	0.20	6	500	0.05	0.10	30	500
vowel	0.10	0.10	8	1500	0.10	0.20	4	1750	0.05	0.05	10	7500	0.40	0.20	10	7500
wdbc	0.05	0.05	27	500	0.05	0.05	29	500	0.10	0.05	29	500	0.05	0.10	28	500

Table B.22: Parameters of the combiner *STCP* on *Conservative* experts

	3-expert				9-expert				20-expert				40-expert			
dataset	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**
aritm	0.20	0.05	6	6000	0.20	0.20	29	500	0.05	0.10	30	500	0.20	0.10	23	500
bala	0.20	0.20	3	1500	0.20	0.01	2	4000	0.05	0.10	3	1500	0.40	0.10	4	500
band	0.40	0.20	3	1500	0.40	0.01	16	2500	0.40	0.20	19	500	0.05	0.05	3	500
bupa	0.40	0.20	30	3000	0.40	0.01	7	4000	0.40	0.05	8	1500	0.40	0.20	3	4500
cred	0.40	0.10	25	500	0.40	0.05	12	500	0.05	0.05	26	500	0.05	0.20	29	500
derma	0.40	0.10	3	1500	0.40	0.05	2	5000	0.40	0.20	2	7500	0.20	0.10	2	4000
ecoli	0.20	0.20	6	1500	0.20	0.20	12	1500	0.40	0.20	30	500	0.40	0.20	25	500
flare	0.05	0.20	2	7500	0.05	0.20	16	4000	0.40	0.01	29	2500	0.40	0.05	30	3000
glas	0.40	0.20	20	7500	0.40	0.20	10	7500	0.40	0.10	13	7500	0.40	0.01	16	7500
hear	0.40	0.10	2	6500	0.40	0.05	3	1750	0.40	0.20	2	500	0.40	0.10	2	3000
img	0.40	0.20	2	2500	0.40	0.05	3	1500	0.05	0.20	3	1500	0.40	0.20	2	5000
ionos	0.05	0.01	27	500	0.05	0.01	14	500	0.20	0.01	16	500	0.05	0.05	8	500
mok1	0.40	0.20	27	7500	0.40	0.20	28	7500	0.40	0.20	30	7500	0.40	0.20	28	7500
mok2	0.20	0.01	26	3000	0.20	0.20	22	500	0.10	0.01	30	500	0.40	0.10	21	500
pima	0.40	0.01	8	500	0.40	0.20	8	500	0.20	0.20	17	500	0.20	0.05	9	500
survi	0.05	0.01	28	5000	0.05	0.20	26	500	0.40	0.01	6	6500	0.40	0.20	27	6000
vote	0.05	0.20	3	5000	0.05	0.20	20	500	0.05	0.01	28	500	0.05	0.01	29	500
vowel	0.05	0.20	8	1750	0.05	0.10	4	7500	0.10	0.05	13	7500	0.40	0.20	11	7500
wdbc	0.05	0.01	29	500	0.05	0.05	29	500	0.10	0.05	30	500	0.05	0.01	25	500

Table B.23: Parameters of the combiner *STC* on *Inverse* experts

	3-expert				9-expert				20-expert				40-expert			
dataset	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**
aritm	0.05	0.10	8	500	0.05	0.05	6	500	0.40	0.20	9	5000	0.40	0.05	23	7500
bala	0.05	0.01	2	7500	0.05	0.10	16	4000	0.40	0.20	27	500	0.20	0.20	4	1500
band	0.20	0.10	5	500	0.20	0.20	7	1500	0.40	0.10	30	4000	0.40	0.20	25	4000
bupa	0.05	0.10	4	1750	0.05	0.05	3	4000	0.20	0.20	4	1500	0.10	0.10	3	1750
cred	0.40	0.20	30	500	0.40	0.01	4	7500	0.20	0.20	4	1500	0.40	0.05	3	500
derma	0.20	0.10	2	7500	0.20	0.20	29	7500	0.40	0.20	27	7500	0.40	0.01	3	7500
ecoli	0.40	0.20	28	6500	0.40	0.20	6	1500	0.20	0.01	4	2500	0.40	0.05	4	1500
flare	0.40	0.20	2	7500	0.40	0.01	8	7500	0.20	0.01	2	7500	0.20	0.20	23	7500
glas	0.40	0.10	3	7500	0.40	0.05	3	7500	0.40	0.10	3	7500	0.40	0.20	4	7500
hear	0.40	0.20	14	4000	0.40	0.01	3	4500	0.40	0.10	5	5000	0.40	0.20	29	5000
img	0.10	0.01	13	2500	0.10	0.20	7	4000	0.40	0.10	3	3000	0.40	0.05	5	7500
ionos	0.40	0.10	13	5000	0.40	0.05	13	7500	0.20	0.10	2	3000	0.20	0.20	2	3000
mok1	0.40	0.20	29	7500	0.40	0.20	27	7500	0.40	0.20	30	7500	0.40	0.20	30	7500
mok2	0.05	0.10	7	3000	0.05	0.20	12	6500	0.40	0.20	13	7500	0.40	0.01	28	4500
pima	0.40	0.20	28	1500	0.40	0.20	7	500	0.20	0.01	7	500	0.10	0.10	3	500
survi	0.20	0.20	5	500	0.20	0.10	12	7500	0.40	0.20	27	7500	0.40	0.20	26	7500
vote	0.20	0.01	29	500	0.20	0.05	30	500	0.20	0.20	18	500	0.05	0.01	30	500
vowel	0.05	0.01	7	1500	0.05	0.10	5	1500	0.40	0.01	9	1500	0.05	0.05	13	4000
wdbc	0.40	0.05	11	7500	0.40	0.20	3	1500	0.40	0.05	2	1750	0.40	0.05	2	1750

Table B.24: Parameters of the combiner *STCP* on *Inverse* experts

	3-expert				9-expert				20-expert				40-expert			
dataset	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**
aritm	0.40	0.20	16	1750	0.40	0.10	18	4000	0.20	0.20	13	1500	0.40	0.10	18	1500
bala	0.40	0.20	27	4000	0.40	0.01	29	1750	0.40	0.20	10	3000	0.40	0.01	12	4500
band	0.40	0.20	7	1500	0.40	0.10	2	1500	0.40	0.05	5	4000	0.40	0.10	6	5000
bupa	0.40	0.10	23	6000	0.40	0.05	17	4500	0.40	0.05	2	1500	0.40	0.20	29	4500
cred	0.40	0.20	29	500	0.40	0.10	28	500	0.40	0.20	24	500	0.40	0.10	27	7500
derma	0.10	0.10	4	4000	0.10	0.20	3	7500	0.40	0.10	3	7500	0.40	0.10	5	7500
ecoli	0.40	0.20	7	4000	0.40	0.20	3	4000	0.40	0.05	6	1500	0.40	0.10	7	4000
flare	0.10	0.05	2	7500	0.10	0.20	2	7500	0.40	0.05	23	6000	0.20	0.10	21	7500
glas	0.40	0.20	3	7500	0.40	0.05	3	7500	0.40	0.05	3	7500	0.40	0.01	4	7500
hear	0.10	0.10	3	7500	0.10	0.20	4	1500	0.20	0.20	4	1500	0.40	0.20	7	6500
img	0.40	0.05	14	4000	0.40	0.20	5	3000	0.40	0.01	8	1750	0.20	0.05	4	1750
ionos	0.40	0.20	2	4000	0.40	0.20	4	7500	0.40	0.05	5	4000	0.40	0.10	5	5000
mok1	0.40	0.20	27	7500	0.40	0.20	28	7500	0.40	0.20	28	7500	0.40	0.20	30	7500
mok2	0.40	0.20	22	7500	0.40	0.20	6	3000	0.40	0.10	6	4000	0.40	0.05	23	4500
pima	0.40	0.20	9	500	0.40	0.10	5	500	0.10	0.20	10	500	0.10	0.20	19	500
survi	0.10	0.20	2	7500	0.10	0.10	20	7500	0.40	0.20	17	6500	0.40	0.20	29	7500
vote	0.40	0.10	7	6500	0.40	0.01	2	4000	0.40	0.10	3	1750	0.40	0.05	3	4000
vowel	0.10	0.05	21	7500	0.10	0.05	5	1500	0.05	0.10	12	7500	0.10	0.10	7	4000
wdbc	0.10	0.05	9	500	0.10	0.01	15	500	0.05	0.05	15	500	0.40	0.20	2	1500

Table B.25: Parameters of the combiner *STC* on *RBFSE* experts

	3-expert				9-expert				20-expert				40-expert			
dataset	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**
aritm	0.40	0.10	28	500	0.40	0.20	21	500	0.20	0.05	17	500	0.40	0.05	2	500
bala	0.05	0.10	4	3000	0.05	0.20	4	750	0.05	0.01	2	2000	0.40	0.20	2	1500
band	0.10	0.01	25	750	0.10	0.10	27	750	0.20	0.05	25	750	0.05	0.10	25	750
bupa	0.05	0.01	14	1500	0.05	0.10	3	5000	0.05	0.10	27	2000	0.05	0.20	21	1500
cred	0.40	0.20	20	2500	0.40	0.10	27	2000	0.40	0.20	18	2000	0.40	0.20	30	2000
derma	0.40	0.01	6	1000	0.40	0.01	4	1000	0.20	0.01	5	1000	0.40	0.10	8	1000
ecoli	0.05	0.10	3	7500	0.05	0.20	4	3000	0.40	0.10	7	1500	0.10	0.05	4	1750
flare	0.40	0.20	27	15000	0.40	0.10	27	15000	0.05	0.20	30	15000	0.40	0.20	17	15000
glas	0.20	0.20	3	2000	0.20	0.05	4	2000	0.10	0.20	4	2000	0.40	0.01	6	2000
hear	0.05	0.05	28	1500	0.05	0.05	18	1500	0.40	0.01	25	1500	0.40	0.10	25	1500
img	0.40	0.05	8	2500	0.40	0.10	9	2500	0.40	0.10	4	5000	0.40	0.20	9	2000
ionos	0.05	0.05	30	7500	0.05	0.05	6	4000	0.40	0.10	3	4000	0.40	0.10	5	4000
mok1	0.40	0.05	2	1000	0.40	0.20	2	1000	0.40	0.20	5	1000	0.40	0.20	5	1000
mok2	0.05	0.05	24	1000	0.05	0.01	11	1000	0.40	0.20	12	1000	0.10	0.05	23	1000
pima	0.20	0.20	6	2500	0.20	0.05	11	2500	0.40	0.20	8	2500	0.20	0.20	24	4500
survi	0.40	0.20	8	2000	0.40	0.20	6	10000	0.20	0.10	23	7500	0.40	0.10	29	6000
vote	0.40	0.20	16	2000	0.40	0.01	4	1000	0.40	0.10	3	1500	0.40	0.10	26	5000
vowel	0.40	0.10	7	1500	0.40	0.01	7	1500	0.40	0.20	5	1500	0.40	0.05	4	1500
wdbc	0.05	0.10	30	1000	0.05	0.01	26	1000	0.05	0.05	28	1000	0.05	0.10	29	1000

Table B.26: Parameters of the combiner *STCP* on *RBFSE* experts

	3-expert				9-expert				20-expert				40-expert			
dataset	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**
aritm	0.40	0.10	17	5000	0.40	0.10	20	5000	0.40	0.10	15	5000	0.40	0.05	14	5000
bala	0.40	0.10	6	8000	0.40	0.20	3	2000	0.40	0.20	2	15000	0.40	0.05	2	1500
band	0.40	0.20	26	2000	0.40	0.05	29	2000	0.20	0.05	29	2000	0.10	0.01	12	2000
bupa	0.40	0.20	30	5000	0.40	0.01	28	7500	0.40	0.20	24	7500	0.40	0.20	4	7500
cred	0.40	0.20	30	4000	0.40	0.10	29	6000	0.40	0.20	5	5000	0.40	0.05	2	3000
derma	0.40	0.05	6	1500	0.40	0.05	5	1500	0.40	0.20	9	1500	0.40	0.10	5	1500
ecoli	0.40	0.20	28	1500	0.40	0.20	3	1500	0.40	0.10	3	1500	0.40	0.20	3	2000
flare	0.40	0.01	13	4000	0.40	0.20	21	4000	0.05	0.01	12	6500	0.05	0.10	18	6000
glas	0.40	0.05	3	2500	0.40	0.10	4	2500	0.10	0.01	3	2500	0.20	0.20	4	2500
hear	0.20	0.01	29	7500	0.20	0.05	15	4000	0.40	0.01	11	7500	0.40	0.01	30	4000
img	0.20	0.05	14	10000	0.20	0.10	12	5000	0.20	0.05	8	10000	0.05	0.20	6	5000
ionos	0.05	0.01	25	1500	0.05	0.01	26	1500	0.05	0.01	21	1500	0.05	0.01	24	1500
mok1	0.40	0.20	2	2500	0.40	0.20	2	2500	0.40	0.20	4	2500	0.40	0.20	5	2500
mok2	0.10	0.01	26	2000	0.10	0.05	11	2000	0.40	0.20	14	2000	0.10	0.01	24	2000
pima	0.20	0.20	9	1500	0.20	0.10	8	1500	0.10	0.05	10	1500	0.10	0.01	20	1500
survi	0.05	0.20	6	2500	0.05	0.05	5	5000	0.40	0.20	14	7500	0.40	0.10	14	7500
vote	0.40	0.10	19	3000	0.40	0.05	30	2500	0.40	0.01	21	2500	0.40	0.05	27	2500
vowel	0.05	0.20	6	7500	0.05	0.20	9	4000	0.05	0.10	7	4000	0.10	0.01	6	4500
wdbc	0.05	0.01	25	2500	0.05	0.10	27	1500	0.05	0.05	29	1500	0.05	0.01	30	1500

Table B.27: Parameters of the combiner *STC* on $RBFSE_{threshold}$ experts

	3-expert				9-expert				20-expert				40-expert			
dataset	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**
aritm	0.40	0.10	28	1000	0.40	0.20	21	1000	0.20	0.05	17	1000	0.40	0.05	2	1000
bala	0.10	0.20	6	2000	0.10	0.10	2	2000	0.05	0.01	2	2000	0.05	0.20	4	2000
band	0.10	0.01	25	2000	0.10	0.10	27	2000	0.20	0.05	28	2000	0.05	0.10	25	2000
bupa	0.05	0.01	14	1500	0.05	0.10	4	4000	0.10	0.05	13	2000	0.05	0.10	30	2000
cred	0.40	0.20	20	2000	0.40	0.10	27	2000	0.40	0.20	2	2000	0.40	0.10	19	2000
derma	0.40	0.20	4	1000	0.40	0.01	5	1000	0.40	0.05	5	1000	0.40	0.20	10	1000
ecoli	0.40	0.20	4	1500	0.40	0.10	5	1500	0.40	0.20	3	1500	0.40	0.20	3	1500
flare	0.40	0.20	27	15000	0.40	0.10	27	15000	0.40	0.20	24	15000	0.40	0.20	17	15000
glas	0.40	0.20	6	2000	0.40	0.20	4	2000	0.05	0.05	4	2000	0.10	0.01	4	2000
hear	0.05	0.05	24	1500	0.05	0.05	18	1500	0.40	0.10	5	5000	0.40	0.01	24	5000
img	0.40	0.10	15	1750	0.40	0.01	11	1750	0.20	0.10	12	1750	0.40	0.20	10	1750
ionos	0.05	0.05	30	7500	0.05	0.20	5	5000	0.40	0.05	3	5000	0.20	0.10	3	5000
mok1	0.40	0.05	2	1000	0.40	0.20	2	1000	0.40	0.20	5	1000	0.40	0.20	5	1000
mok2	0.05	0.05	24	1000	0.05	0.10	10	1000	0.40	0.20	12	1000	0.20	0.01	23	1000
pima	0.20	0.20	6	1500	0.20	0.05	11	1500	0.05	0.20	4	4000	0.10	0.20	7	6000
survi	0.40	0.20	8	2000	0.40	0.05	12	7500	0.40	0.20	17	6500	0.40	0.20	9	7500
vote	0.40	0.20	16	3000	0.40	0.10	5	1500	0.40	0.05	3	1500	0.40	0.20	29	4000
vowel	0.10	0.20	7	2000	0.10	0.01	9	2000	0.20	0.05	7	2000	0.40	0.20	5	2000
wdbc	0.05	0.10	30	1500	0.05	0.01	26	1500	0.05	0.05	28	1500	0.05	0.10	29	1500

Table B.28: Parameters of the combiner *STCP* on $RBFSE_{threshold}$ experts

	3-expert				9-expert				20-expert				40-expert			
dataset	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**
aritm	0.40	0.05	7	2000	0.40	0.05	17	5000	0.40	0.05	28	2500	0.40	0.01	25	2500
bala	0.20	0.01	5	10000	0.20	0.05	4	2500	0.40	0.01	14	5000	0.20	0.20	2	7500
band	0.40	0.20	26	1500	0.40	0.05	29	1500	0.05	0.01	30	1500	0.10	0.20	9	1500
bupa	0.40	0.20	30	5000	0.40	0.01	24	7500	0.40	0.01	11	7500	0.40	0.20	29	7500
cred	0.40	0.05	4	6000	0.40	0.05	14	7500	0.40	0.05	10	7500	0.40	0.05	2	4000
derma	0.40	0.05	6	1500	0.40	0.01	3	1500	0.40	0.20	5	1500	0.40	0.01	5	1500
ecoli	0.40	0.20	9	2000	0.40	0.10	12	2000	0.40	0.10	5	2000	0.40	0.20	3	2000
flare	0.05	0.05	15	15000	0.05	0.20	21	5000	0.05	0.01	12	10000	0.05	0.10	18	10000
glas	0.40	0.10	4	1500	0.40	0.10	4	3000	0.40	0.20	4	1500	0.40	0.10	4	1500
hear	0.40	0.01	7	1500	0.40	0.20	18	1500	0.05	0.01	24	1500	0.40	0.05	28	1500
img	0.40	0.10	11	5000	0.40	0.01	9	5000	0.20	0.01	6	5000	0.40	0.20	8	5000
ionos	0.05	0.01	25	1500	0.05	0.01	26	1500	0.05	0.01	21	1500	0.05	0.01	24	1500
mok1	0.40	0.20	2	1500	0.40	0.20	2	1500	0.40	0.20	4	1500	0.40	0.20	5	1500
mok2	0.20	0.10	17	2000	0.20	0.20	10	2000	0.40	0.01	11	2000	0.40	0.20	14	2000
pima	0.20	0.20	9	2500	0.20	0.01	5	3000	0.20	0.20	5	6500	0.40	0.10	11	7500
survi	0.20	0.20	9	10000	0.20	0.05	5	10000	0.40	0.10	26	10000	0.40	0.01	19	10000
vote	0.40	0.10	28	2000	0.40	0.05	30	2000	0.40	0.20	2	3000	0.40	0.20	2	2000
vowel	0.10	0.01	13	2500	0.10	0.01	5	2500	0.20	0.01	8	2500	0.20	0.20	6	2500
wdbc	0.05	0.01	25	2000	0.05	0.10	27	2000	0.05	0.05	29	2000	0.05	0.01	30	2500

Table B.29: Parameters of the combiner *STC* on $RBFSE_{min-max}$ experts

	3-expert				9-expert				20-expert				40-expert			
dataset	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**
aritm	0.05	0.20	9	1500	0.05	0.05	6	1750	0.40	0.05	15	3000	0.20	0.01	17	6000
bala	0.20	0.05	2	1500	0.20	0.05	5	2000	0.40	0.20	3	2000	0.40	0.01	2	2000
band	0.40	0.20	2	1000	0.40	0.05	9	1000	0.05	0.05	30	1000	0.20	0.10	19	1000
bupa	0.40	0.05	17	6000	0.40	0.05	21	5000	0.10	0.20	13	5000	0.05	0.05	8	5000
cred	0.20	0.20	2	5000	0.20	0.10	7	2500	0.40	0.20	10	7500	0.40	0.20	12	6000
derma	0.40	0.10	7	1500	0.40	0.01	5	1500	0.40	0.20	9	1500	0.40	0.20	8	1500
ecoli	0.40	0.20	4	2000	0.40	0.10	7	2000	0.40	0.05	7	2000	0.40	0.20	3	5000
flare	0.05	0.20	24	10000	0.05	0.10	26	10000	0.05	0.10	26	7500	0.05	0.05	27	7500
glas	0.40	0.01	5	1500	0.40	0.01	6	1500	0.05	0.05	4	1500	0.40	0.20	5	1500
hear	0.20	0.01	7	7500	0.20	0.20	10	6000	0.40	0.05	17	6000	0.40	0.20	24	6000
img	0.20	0.20	5	3000	0.20	0.10	11	2000	0.40	0.10	8	2000	0.05	0.10	7	2000
ionos	0.05	0.20	6	1500	0.05	0.01	4	1500	0.20	0.10	2	1500	0.40	0.01	4	3000
mok1	0.40	0.20	24	1500	0.40	0.20	29	1500	0.40	0.20	27	1500	0.40	0.20	30	1500
mok2	0.20	0.01	29	2500	0.20	0.05	7	2500	0.40	0.05	6	2500	0.40	0.10	10	2500
pima	0.40	0.05	6	4000	0.40	0.20	25	4500	0.10	0.20	13	5000	0.05	0.20	16	10000
survi	0.40	0.20	4	2000	0.40	0.20	18	4000	0.10	0.20	14	6500	0.40	0.20	12	7500
vote	0.40	0.01	9	4000	0.40	0.20	11	4000	0.20	0.05	18	4000	0.40	0.20	18	5000
vowel	0.10	0.20	7	1500	0.10	0.05	14	2000	0.40	0.20	5	2000	0.05	0.05	7	4000
wdbc	0.20	0.05	14	2000	0.20	0.20	9	4000	0.40	0.05	3	2000	0.40	0.20	2	2000

Table B.30: Parameters of the combiner *STCP* on $RBFSE_{min-max}$ experts

	3-expert				9-expert				20-expert				40-expert			
dataset	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**
aritm	0.20	0.05	2	4000	0.20	0.01	28	8000	0.40	0.01	26	4000	0.20	0.01	21	4000
bala	0.40	0.20	3	5000	0.40	0.20	3	3000	0.40	0.01	2	6500	0.40	0.10	5	3000
band	0.05	0.10	7	1500	0.05	0.01	2	1500	0.10	0.01	28	1500	0.20	0.01	27	1500
bupa	0.40	0.05	23	5000	0.40	0.05	23	5000	0.05	0.05	9	5000	0.10	0.20	27	7500
cred	0.40	0.10	26	6000	0.40	0.01	28	8000	0.10	0.20	10	8000	0.40	0.20	27	8000
derma	0.40	0.01	6	1500	0.40	0.05	5	1500	0.40	0.20	9	1500	0.40	0.10	7	1500
ecoli	0.20	0.10	14	2500	0.20	0.01	3	2500	0.40	0.10	3	5000	0.40	0.20	3	5000
flare	0.40	0.20	11	5000	0.40	0.05	27	5000	0.40	0.20	24	5000	0.10	0.10	24	5000
glas	0.20	0.05	4	2000	0.20	0.10	3	3000	0.40	0.01	4	2000	0.40	0.10	4	2000
hear	0.40	0.05	25	5000	0.40	0.05	25	5000	0.40	0.20	16	4500	0.40	0.05	28	4500
img	0.10	0.05	6	2500	0.10	0.20	6	5000	0.40	0.20	7	4500	0.05	0.10	12	4000
ionos	0.40	0.20	6	5000	0.40	0.20	6	5000	0.20	0.01	2	2500	0.40	0.01	3	2500
mok1	0.40	0.20	29	1500	0.40	0.20	28	1500	0.40	0.20	30	1500	0.40	0.20	29	1500
mok2	0.10	0.05	30	2000	0.10	0.01	8	2000	0.40	0.01	6	2500	0.40	0.10	2	2000
pima	0.40	0.20	10	5000	0.40	0.05	22	4000	0.05	0.10	15	4000	0.10	0.01	26	6500
survi	0.10	0.20	21	10000	0.10	0.05	2	2500	0.20	0.01	29	6000	0.40	0.20	13	6000
vote	0.40	0.01	9	4500	0.40	0.20	13	3000	0.40	0.01	2	2500	0.10	0.01	20	2000
vowel	0.40	0.05	8	2500	0.40	0.10	11	2500	0.20	0.01	8	2500	0.10	0.10	12	2500
wdbc	0.40	0.10	2	2500	0.40	0.01	2	5000	0.40	0.10	3	5000	0.20	0.05	3	5000

Table B.31: Parameters of the combiner *STC* on $RBFSE_{sum}$ experts

	3-expert				9-expert				20-expert				40-expert			
dataset	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**
aritm	0.05	0.01	9	7500	0.05	0.10	20	6000	0.20	0.01	24	5000	0.40	0.05	2	3000
bala	0.05	0.10	4	3000	0.05	0.10	2	1000	0.05	0.01	2	1500	0.05	0.20	4	1500
band	0.05	0.10	26	3500	0.05	0.05	30	1500	0.05	0.10	30	1500	0.10	0.10	30	1500
bupa	0.40	0.01	2	2000	0.40	0.01	20	5000	0.10	0.05	11	1500	0.10	0.20	3	7500
cred	0.40	0.20	18	2500	0.40	0.05	2	2500	0.40	0.20	18	2500	0.40	0.20	30	2500
derma	0.20	0.10	7	2000	0.20	0.10	3	2000	0.20	0.10	3	2000	0.20	0.10	5	2000
ecoli	0.40	0.20	7	2500	0.40	0.01	5	3000	0.10	0.01	5	4000	0.40	0.20	5	4000
flare	0.40	0.20	2	3000	0.40	0.20	6	10000	0.20	0.01	11	5000	0.20	0.01	19	8000
glas	0.40	0.10	3	2000	0.40	0.05	4	2500	0.20	0.10	3	2500	0.40	0.01	7	2500
hear	0.10	0.20	2	3500	0.10	0.10	22	2500	0.40	0.10	6	2500	0.40	0.05	16	2500
img	0.20	0.01	12	7500	0.20	0.10	6	7500	0.40	0.20	9	7500	0.40	0.01	10	7500
ionos	0.40	0.20	2	12000	0.40	0.10	6	6000	0.40	0.10	4	10000	0.10	0.10	3	10000
mok1	0.40	0.20	2	1000	0.40	0.20	4	1000	0.40	0.20	5	1000	0.40	0.20	5	1000
mok2	0.05	0.05	30	1500	0.05	0.10	10	1500	0.40	0.10	9	1500	0.40	0.20	10	1500
pima	0.20	0.20	6	10000	0.20	0.10	5	5000	0.20	0.01	8	10000	0.20	0.10	8	10000
survi	0.10	0.20	3	2500	0.10	0.01	3	2500	0.40	0.10	14	7500	0.40	0.01	24	8000
vote	0.40	0.01	9	2500	0.40	0.20	25	1500	0.40	0.20	28	1500	0.40	0.20	12	2500
vowel	0.10	0.20	7	2500	0.10	0.10	5	2500	0.20	0.05	7	2500	0.40	0.20	5	2500
wdbc	0.05	0.10	24	1500	0.05	0.01	29	1500	0.05	0.01	29	1500	0.05	0.01	25	1500

Table B.32: Parameters of the combiner *STCP* on $RBFSE_{sum}$ experts

	3-expert				9-expert				20-expert				40-expert			
dataset	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**	**step**	**mom**	**Nh**	**Nite**
aritm	0.40	0.10	17	10000	0.40	0.10	5	9000	0.40	0.20	27	8000	0.40	0.20	18	6000
bala	0.40	0.01	8	7500	0.40	0.05	2	7500	0.20	0.05	3	7500	0.20	0.01	2	7500
band	0.05	0.10	23	1500	0.05	0.10	29	1500	0.10	0.05	30	1500	0.10	0.10	28	1500
bupa	0.40	0.10	6	7500	0.40	0.20	5	7500	0.05	0.10	29	7500	0.40	0.01	24	7500
cred	0.40	0.20	8	1000	0.40	0.10	9	8000	0.20	0.20	8	7500	0.40	0.20	11	5000
derma	0.20	0.10	3	3000	0.20	0.01	6	2500	0.40	0.01	5	2500	0.40	0.10	7	2500
ecoli	0.40	0.10	7	2500	0.40	0.20	7	2500	0.40	0.05	7	5000	0.40	0.20	3	5000
flare	0.05	0.20	2	10000	0.05	0.05	15	5000	0.40	0.05	22	6000	0.05	0.20	29	5000
glas	0.40	0.01	7	1500	0.40	0.05	6	1500	0.20	0.05	4	1500	0.10	0.10	4	2500
hear	0.40	0.01	2	7500	0.40	0.05	7	2500	0.10	0.05	4	4000	0.40	0.05	15	5000
img	0.20	0.05	4	7500	0.20	0.20	7	5000	0.40	0.05	7	4000	0.40	0.20	6	4000
ionos	0.05	0.01	18	2500	0.05	0.05	17	2500	0.05	0.01	19	2500	0.05	0.01	26	2500
mok1	0.40	0.20	2	1500	0.40	0.20	3	1500	0.40	0.20	8	1500	0.40	0.20	7	1500
mok2	0.10	0.20	19	2500	0.10	0.20	10	2500	0.40	0.01	11	2500	0.40	0.01	16	2500
pima	0.20	0.20	9	3000	0.20	0.20	11	3000	0.20	0.20	5	4500	0.05	0.10	8	5500
survi	0.40	0.01	2	1500	0.40	0.05	3	2500	0.40	0.05	8	10000	0.40	0.10	6	8000
vote	0.40	0.10	30	1500	0.40	0.01	29	2500	0.40	0.20	12	2500	0.40	0.01	20	5000
vowel	0.20	0.01	7	2000	0.20	0.05	8	2000	0.10	0.10	6	2000	0.40	0.10	5	2000
wdbc	0.05	0.01	30	2500	0.05	0.01	29	2500	0.05	0.01	30	2500	0.05	0.01	29	2500

B.8 Parameters of *Mixture of Experts*

The parameters related to all the alternatives based on *Mixture of Experts* are introduced here. They correspond to the training parameters of the expert networks (denoted with "expert" and "exp") and gating network ("gating" and "gat" in the tables). The parameter $Nite$ denotes the number of iterations of the global training procedure in all the tables.

Table B.33: Parameters - *MIX-BN-BN*

Database	$Step_{expert}$	$Step_{gating}$	N_{ite}
aritm	0.1	0.1	3250
bala	0.1	0.1	6500
band	0.1	0.1	6500
bupa	0.1	0.1	11050
cred	0.1	0.1	11050
derma	0.1	0.1	1300
ecoli	0.1	0.1	13000
flare	0.6	0.1	13000
glas	0.1	0.1	5200
hear	0.1	0.1	6500
img	0.4	0.1	1950
ionos	0.1	0.1	6500
mok1	0.1	0.1	3900
mok2	0.1	0.1	9100
pima	0.4	0.1	13000
survi	0.1	0.1	26000
vote	0.1	0.1	3250
vowel	0.2	0.1	5200
wdbc	0.1	0.1	5200

Table B.34: Parameters - *MIX-MF-BN*

Database	$Nhidden_{exp}$	$Step_{exp}$	Mom_{exp}	$Step_{gat}$	N_{ite}
aritm	9	0.1	0.05	0.1	3250
bala	20	0.1	0.05	0.1	6500
band	23	0.1	0.05	0.1	6500
bupa	11	0.1	0.05	0.1	11050
cred	15	0.1	0.05	0.1	11050
derma	4	0.1	0.05	0.1	1300
ecoli	5	0.1	0.05	0.1	13000
flare	11	0.6	0.05	0.1	13000
glas	3	0.1	0.05	0.1	5200
hear	2	0.1	0.05	0.1	6500
img	14	0.4	0.05	0.1	1950
ionos	8	0.1	0.05	0.1	6500
mok1	6	0.1	0.05	0.1	3900
mok2	20	0.1	0.05	0.1	9100
pima	14	0.4	0.05	0.1	13000
survi	9	0.1	0.2	0.1	26000
vote	1	0.1	0.05	0.1	3250
vowel	15	0.2	0.2	0.1	5200
wdbc	6	0.1	0.05	0.1	5200

Table B.35: Parameters - *Mix-MF-MF*

				3-expert			9-expert			20-expert			40-expert		
dataset	$\mathbf{Nh}_{exp}$	$\mathbf{step}_{exp}$	$\mathbf{mom}_{exp}$	$\mathbf{Nh}_{gat}$	$\mathbf{step}_{gat}$	**Nite**	$\mathbf{Nh}_{gat}$	$\mathbf{step}_{gat}$	**Nite**	$\mathbf{Nh}_{gat}$	$\mathbf{step}_{gat}$	**Nite**	$\mathbf{Nh}_{gat}$	$\mathbf{step}_{gat}$	**Nite**
aritm	9	0.1	0.05	3	0.001	4100	8	0.1	4100	8	0.001	1250	19	1	1800
bala	20	0.1	0.05	7	1	6500	7	0.001	500	15	0.005	1750	2	0.1	2300
band	23	0.1	0.05	2	0.01	6500	11	0.05	1500	8	0.001	2100	8	0.001	1700
bupa	11	0.1	0.05	8	0.5	11050	6	0.1	6500	2	0.001	2500	8	1	1750
cred	15	0.1	0.05	3	0.005	11050	6	0.001	500	2	0.001	6500	2	0.005	3300
derma	4	0.1	0.05	3	0.01	1300	15	0.5	1750	6	0.5	1800	17	0.01	6500
ecoli	5	0.1	0.05	5	0.005	13000	19	0.001	2250	13	1	2800	9	0.01	4400
flare	11	0.6	0.05	10	0.001	13000	2	0.001	500	3	0.001	500	18	0.5	500
glas	3	0.1	0.05	11	0.5	5200	6	1	500	3	0.5	2900	11	1	500
hear	2	0.1	0.05	10	1	6500	2	0.01	2500	7	0.005	2200	2	1	3000
img	14	0.4	0.05	16	0.001	1950	2	0.1	500	3	0.01	500	19	1	500
ionos	8	0.1	0.05	17	0.05	6500	18	1	6500	11	0.05	6500	2	1	6500
mok1	6	0.1	0.05	7	0.05	6500	16	1	6500	20	0.01	4200	11	0.5	6500
mok2	20	0.1	0.05	19	0.5	9100	12	0.005	1500	5	0.001	2000	18	0.001	1500
pima	14	0.4	0.05	8	0.005	13000	10	0.005	500	16	0.001	2700	8	0.005	2000
survi	9	0.1	0.2	5	0.001	26000	9	0.001	6000	16	1	6500	19	0.005	6500
vote	1	0.1	0.05	20	0.001	6500	13	0.001	1750	8	0.001	6500	18	0.001	2200
vowel	15	0.2	0.2	20	0.001	5200	6	0.05	500	7	0.05	500	11	0.001	1000
wdbc	6	0.1	0.05	10	0.5	5200	8	0.001	6500	18	0.001	500	15	1	500

Table B.36: Parameters - *Mix-SE-MF*

				3-expert			9-expert			20-expert			40-expert		
dataset	$\mathbf{Nh}_{exp}$	$\mathbf{step}_{exp}$	$\mathbf{mom}_{exp}$	$\mathbf{Nh}_{gat}$	$\mathbf{step}_{gat}$	**Nite**	$\mathbf{Nh}_{gat}$	$\mathbf{step}_{gat}$	**Nite**	$\mathbf{Nh}_{gat}$	$\mathbf{step}_{gat}$	**Nite**	$\mathbf{Nh}_{gat}$	$\mathbf{step}_{gat}$	**Nite**
aritm	9	0.1	0.05	2	0.0001	4100	2	0.0001	4100	4	0.0001	1250	3	0.0001	1800
bala	20	0.1	0.05	4	0.0001	6500	8	0.0001	500	3	0.0001	1750	2	0.0001	2300
band	23	0.1	0.05	2	0.0001	6500	3	0.0001	1500	2	0.0001	2100	8	0.0001	1700
bupa	11	0.1	0.05	2	0.0001	11050	12	0.0001	6500	2	0.0001	2500	18	0.0001	1750
cred	15	0.1	0.05	2	0.0001	11050	2	0.0001	500	2	0.0001	6500	2	0.0001	3300
derma	4	0.1	0.05	2	0.0001	1300	3	0.0001	1750	4	0.0001	1800	3	0.0001	6500
ecoli	5	0.1	0.05	2	0.0001	13000	2	0.0001	2250	2	0.0001	2800	2	0.0001	4400
flare	11	0.6	0.05	2	0.0006	13000	2	0.0006	500	2	0.0006	500	2	0.0006	500
glas	3	0.1	0.05	2	0.0001	5200	2	0.0001	500	2	0.0001	2900	2	0.0001	500
hear	2	0.1	0.05	2	0.0001	6500	2	0.0001	2500	5	0.0001	2200	5	0.0001	3000
img	14	0.4	0.05	2	0.0004	1950	15	0.0004	500	2	0.0004	500	4	0.0004	500
ionos	8	0.1	0.05	8	0.0001	6500	2	0.0001	6500	4	0.0001	6500	12	0.0001	6500
mok1	6	0.1	0.05	2	0.0001	6500	2	0.0001	6500	2	0.0001	4200	2	0.0001	6500
mok2	20	0.1	0.05	4	0.0001	9100	9	0.0001	1500	2	0.0001	2000	17	0.0001	1500
pima	14	0.4	0.05	2	0.0004	13000	20	0.0004	500	13	0.0004	2700	3	0.0004	2000
survi	9	0.1	0.2	9	0.0001	26000	2	0.0001	6000	3	0.0001	6500	19	0.0001	6500
vote	1	0.1	0.05	2	0.0001	6500	2	0.0001	1750	2	0.0001	6500	2	0.0001	2200
vowel	15	0.2	0.2	6	0.0002	5200	3	0.0002	500	12	0.0002	500	3	0.0002	1000
wdbc	6	0.1	0.05	2	0.0001	5200	2	0.0001	6500	2	0.0001	500	2	0.0001	500

Table B.37: Parameters - *MIX-SE-BN*

Database	$Nhidden_{exp}$	$Step_{exp}$	Mom_{exp}	$Step_{gat}$	N_{ite}
aritm	9	0.1	0.05	0.1	3250
bala	20	0.1	0.05	0.1	6500
band	23	0.1	0.05	0.1	6500
bupa	11	0.1	0.05	0.1	11050
cred	15	0.1	0.05	0.1	11050
derma	4	0.1	0.05	0.1	1300
ecoli	5	0.1	0.05	0.1	13000
flare	11	0.6	0.05	0.1	13000
glas	3	0.1	0.05	0.1	5200
hear	2	0.1	0.05	0.1	6500
img	14	0.4	0.05	0.1	1950
ionos	8	0.1	0.05	0.1	6500
mok1	6	0.1	0.05	0.1	3900
mok2	20	0.1	0.05	0.1	9100
pima	14	0.4	0.05	0.1	13000
survi	9	0.1	0.2	0.1	26000
vote	1	0.1	0.05	0.1	3250
vowel	15	0.2	0.2	0.1	5200
wdbc	6	0.1	0.05	0.1	5200

Appendix C

Complete Results

C.1 Introduction

In this appendix, all the complete results related to the ensemble methods, ensemble combiners and other multi-net systems are shown.

C.2 Complete results of ensemble methods

Here, we show the results related to the ensembles studied and proposed.

C.2.1 Traditional ensemble methods

First of all, all the results related to the *traditional ensemble methods* are introduced. Those results corresponds to the ensembles described in chapters 4 and 5.

Concretely, tables C.2 to C.10 correspond to the ensemble alternatives based on modifying the learning procedures whereas tables C.11 to C.30 correspond to the ensemble models based on modifying the learning set. Moreover, the results of *Simple Ensemble* of *MF* networks have also be included, table C.1, for comparing its accuracy to the other ensembles.

C.2.2 Reordering on ensembles

In the second subsection, all the results related to applying the new reordering algorithms to the traditional ensembles are shown. These results corresponds to the research described in chapter 7. Concretely, thy are shown in tables C.31 to C.56

C.2.3 Improving boosting

Finally, the results of the proposed boosting variants are shown in this subsection. This results corresponds to the ensembles proposed in chapter 8.

Concretely, the results related to *Averaged-Conservative Boosting* and *Weighted-Conservative Boosting* are shown in tables C.57 to C.60 whereas the results related to the new *Cross Validated Boosting* methodology are shown in tables C.61 to C.92.

C.3 Results of combination methods

In this section, all the results related to the fourteen ensemble combiners analyzed in chapter 6 are shown.

Concretely, the results related to combining ensembles of *MF* networks are shown in tables C.93 to C.108 whereas the results related to combining ensembles of *RBF* networks are shown in tables C.109 to C.124.

C.4 Other Multi-net systems

Finally, all the results related to the *Stacked model* and the *Mixture model* are shown in this subsection. Concretely, the results related to *Stacked* are shown in tables C.125 to C.154 whereas the results of *Mixture* are in tables C.155 to C.159.

Table C.1: Performance - *Simple Ensemble* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	73.4 ± 1	73.8 ± 1.1	73.8 ± 1.1	73.8 ± 1.1
bala	96 ± 0.5	95.8 ± 0.5	95.8 ± 0.6	95.9 ± 0.5
band	73.5 ± 1.2	72.9 ± 1.5	73.8 ± 1.3	73.8 ± 1.3
bupa	72.3 ± 1.2	72.4 ± 1.1	72.3 ± 1.1	72.7 ± 1.1
cred	86.5 ± 0.7	86.4 ± 0.7	86.6 ± 0.7	86.5 ± 0.7
derma	97.2 ± 0.7	97.5 ± 0.7	97.3 ± 0.7	97.6 ± 0.7
ecoli	86.6 ± 0.8	86.9 ± 0.8	86.9 ± 0.8	86.9 ± 0.7
flare	81.8 ± 0.5	81.6 ± 0.4	81.5 ± 0.5	81.6 ± 0.5
glas	94 ± 0.8	94 ± 0.7	94 ± 0.7	94.2 ± 0.6
hear	82.9 ± 1.5	83.1 ± 1.5	83.1 ± 1.5	82.9 ± 1.5
img	96.5 ± 0.2	96.7 ± 0.3	96.7 ± 0.2	96.8 ± 0.2
ionos	91.1 ± 1.1	90.3 ± 1.1	90.4 ± 1	90.3 ± 1
mok1	98.3 ± 0.9	98.8 ± 0.8	98.3 ± 0.9	98.3 ± 0.9
mok2	88 ± 2	90.8 ± 1.8	91.1 ± 1.1	91.1 ± 1.2
pima	75.9 ± 1.2	75.9 ± 1.2	75.9 ± 1.2	75.9 ± 1.2
survi	74.3 ± 1.3	74.2 ± 1.3	74.3 ± 1.3	74.3 ± 1.3
vote	95.6 ± 0.5	95.6 ± 0.5	95.6 ± 0.5	95.6 ± 0.5
vowel	88 ± 0.9	91 ± 0.5	91.4 ± 0.8	92.2 ± 0.7
wdbc	96.9 ± 0.5	96.9 ± 0.5	96.9 ± 0.5	96.9 ± 0.5

Table C.2: Performance - *CELS* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	73.3 ± 1.1	74.1 ± 1.4	73.8 ± 1.8	75.3 ± 1.7
bala	95.8 ± 0.7	95.8 ± 0.6	96.3 ± 0.5	96.6 ± 0.4
band	71.5 ± 1.3	73.3 ± 1.1	72.4 ± 1.3	74.4 ± 1.1
bupa	68.0 ± 1.4	73.3 ± 1.5	70.6 ± 1.2	69.3 ± 1.2
cred	86.5 ± 0.7	85.8 ± 0.9	86.5 ± 0.9	86.9 ± 0.5
derma	97.2 ± 0.7	97.5 ± 0.7	97.3 ± 0.6	97.3 ± 0.7
ecoli	84.3 ± 1.0	85.7 ± 0.9	85.7 ± 0.7	87.4 ± 0.9
flare	81.5 ± 0.7	81.5 ± 0.6	81.3 ± 0.5	81.7 ± 0.5
glas	94.4 ± 1.1	95.2 ± 1.0	96.4 ± 0.8	95.6 ± 0.7
hear	81.7 ± 1.2	82.9 ± 1.6	84.2 ± 1.0	85.6 ± 1.5
img	96.8 ± 0.3	96.8 ± 0.2	96.9 ± 0.3	96.8 ± 0.2
ionos	91.3 ± 0.8	91.4 ± 1.0	92.1 ± 1.0	92.3 ± 1.0
mok1	99.4 ± 0.6	100.00 ± 0.00	99.3 ± 0.8	99.4 ± 0.6
mok2	88.6 ± 1.3	90.1 ± 1.1	88.8 ± 1.5	84.4 ± 0.9
pima	75.9 ± 1.2	75.4 ± 1.2	76.2 ± 1.1	76.3 ± 1.1
survi	73.9 ± 1.3	73.6 ± 1.1	73.9 ± 0.9	73.8 ± 1.0
vote	95.6 ± 0.6	95.1 ± 0.7	95.0 ± 0.7	94.9 ± 0.8
vowel	91.3 ± 0.7	92.9 ± 0.7	94.0 ± 0.4	94.5 ± 0.7
wdbc	96.9 ± 0.5	96.7 ± 0.5	97.3 ± 0.4	97.1 ± 0.4

Table C.3: Performance - *DECOv1* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	74.9 ± 1.3	76.1 ± 1.1	75.5 ± 1.4	75.6 ± 1.3
bala	96.3 ± 0.5	96.7 ± 0.5	96.6 ± 0.5	96.7 ± 0.4
band	73.6 ± 1.1	73.8 ± 1.3	73.6 ± 1.4	73.8 ± 1.3
bupa	72.6 ± 1.3	72.7 ± 1.0	73.1 ± 1.2	73.0 ± 1.7
cred	86.4 ± 0.7	86.4 ± 0.7	86.5 ± 0.7	86.5 ± 0.7
derma	97.2 ± 0.7	97.6 ± 0.7	97.6 ± 0.7	97.6 ± 0.7
ecoli	86.6 ± 0.6	87.2 ± 0.7	87.1 ± 0.7	87.5 ± 0.7
flare	81.7 ± 0.4	81.6 ± 0.6	81.3 ± 0.5	81.4 ± 0.5
glas	94.8 ± 0.6	95.4 ± 0.7	95.4 ± 0.7	95.2 ± 0.7
hear	83.2 ± 1.4	83.2 ± 1.4	83.1 ± 1.4	83.1 ± 1.4
img	96.7 ± 0.3	96.9 ± 0.2	96.9 ± 0.2	96.9 ± 0.2
ionos	90.9 ± 0.9	90.7 ± 1.0	91.1 ± 0.9	91.0 ± 1.0
mok1	99.4 ± 0.6	100.00 ± 0.00	99.5 ± 0.5	99.4 ± 0.6
mok2	88.9 ± 2.0	90.8 ± 1.0	91.4 ± 0.9	91.8 ± 0.7
pima	76.4 ± 1.2	76.1 ± 1.1	76.1 ± 1.0	75.9 ± 1.0
survi	74.6 ± 1.5	73.9 ± 1.3	74.1 ± 1.4	73.9 ± 1.4
vote	95.8 ± 0.6	95.4 ± 0.6	95.5 ± 0.6	95.6 ± 0.6
vowel	91.5 ± 0.6	92.8 ± 0.7	93.3 ± 0.6	93.1 ± 0.6
wdbc	97.0 ± 0.5	97.0 ± 0.5	97.0 ± 0.5	97.0 ± 0.5

Table C.4: Performance - *DECOv2* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	73.9 ± 1.0	73.9 ± 1.1	74.4 ± 1.2	74.4 ± 1.2
bala	96.9 ± 0.4	96.6 ± 0.4	95.9 ± 0.6	95.8 ± 0.6
band	75.3 ± 1.3	73.1 ± 1.3	73.3 ± 1.2	73.3 ± 1.2
bupa	72.7 ± 1.4	72.0 ± 1.1	71.9 ± 1.2	72.3 ± 1.2
cred	86.5 ± 0.7	86.6 ± 0.7	86.5 ± 0.7	86.5 ± 0.7
derma	97.6 ± 0.7	97.6 ± 0.7	97.6 ± 0.7	97.6 ± 0.7
ecoli	87.2 ± 0.9	87.8 ± 0.7	88.1 ± 0.7	88.2 ± 0.7
flare	81.6 ± 0.4	81.7 ± 0.4	81.6 ± 0.5	81.7 ± 0.4
glas	94.8 ± 0.7	95.2 ± 0.8	95.2 ± 0.7	95.4 ± 0.7
hear	83.1 ± 1.4	83.1 ± 1.4	83.2 ± 1.4	83.4 ± 1.4
img	96.7 ± 0.3	96.84 ± 0.18	96.8 ± 0.2	96.77 ± 0.20
ionos	90.6 ± 1.0	90.4 ± 0.9	90.9 ± 0.9	90.7 ± 0.9
mok1	97.6 ± 1.3	100.00 ± 0.00	99.0 ± 0.7	99.88 ± 0.13
mok2	90.8 ± 1.0	89.6 ± 1.8	90.1 ± 1.4	91.8 ± 0.8
pima	75.7 ± 1.1	76.0 ± 1.0	76.1 ± 1.0	76.2 ± 1.0
survi	74.3 ± 1.4	73.8 ± 1.3	74.3 ± 1.3	74.3 ± 1.3
vote	95.5 ± 0.6	95.5 ± 0.6	95.5 ± 0.6	95.5 ± 0.6
vowel	90.3 ± 0.4	92.6 ± 0.5	93.3 ± 0.5	93.5 ± 0.6
wdbc	97.0 ± 0.5	97.0 ± 0.5	97.0 ± 0.5	96.9 ± 0.4

Table C.5: Performance - *EVOL* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	65.4 ± 1.4	65.9 ± 1.9	65.9 ± 1.9	59 ± 2
bala	62 ± 5	53 ± 5	62 ± 4	62 ± 4
band	65.3 ± 1.2	65.3 ± 1.2	65.3 ± 1.2	65.3 ± 1.2
bupa	59.7 ± 0.6	59.7 ± 0.6	59.9 ± 0.7	59.7 ± 0.6
cred	64 ± 4	57 ± 3	57 ± 3	57 ± 3
derma	57 ± 5	54 ± 6	47 ± 5	41 ± 7
ecoli	57 ± 5	57 ± 5	55 ± 4	52 ± 6
flare	80.7 ± 0.7	80.6 ± 0.8	81.2 ± 0.5	81.2 ± 0.5
glas	77 ± 4	59 ± 4	52 ± 6	37 ± 3
hear	63 ± 3	65 ± 4	60 ± 3	58 ± 4
img	77 ± 5	67 ± 4	63 ± 5	63 ± 4
ionos	83.4 ± 1.9	77 ± 3	66.1 ± 1.2	64.1 ± 1.3
mok1	69 ± 5	50.8 ± 1.2	50.9 ± 1.2	50.9 ± 1.2
mok2	66 ± 1.3	66.5 ± 1	66.5 ± 1	66.5 ± 1
pima	66.3 ± 1.2	66.1 ± 0.7	65.2 ± 0.9	65.9 ± 1
survi	74.3 ± 0.6	74.8 ± 0.7	74.8 ± 0.7	74.8 ± 0.7
vote	92.1 ± 1.9	81 ± 6	63 ± 3	63 ± 3
vowel	77.5 ± 1.7	61 ± 4	60 ± 3	54 ± 3
wdbc	94.4 ± 0.9	87.2 ± 1.6	78 ± 3	77.6 ± 1.9

Table C.6: Performance - *OLA* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	74.7 ± 1.4	72.5 ± 1	72.5 ± 1.1	75.1 ± 1.1
bala	90.2 ± 1	89.2 ± 1	90.2 ± 0.8	90.3 ± 0.8
band	68 ± 2	68.5 ± 1.7	70 ± 1.4	71.4 ± 1.6
bupa	69 ± 2	69.3 ± 1.4	69.5 ± 1.6	67.8 ± 1.2
cred	84.4 ± 0.9	84.9 ± 0.9	85.2 ± 0.8	85.1 ± 0.8
derma	91.4 ± 1.5	86.7 ± 1.7	87 ± 1.4	87.7 ± 1.6
ecoli	82.4 ± 1.4	83.5 ± 1.3	84.3 ± 1.2	84.9 ± 1.3
flare	81.1 ± 0.4	80.8 ± 0.4	80.7 ± 0.4	80.7 ± 0.4
glas	77 ± 2	60 ± 4	77.4 ± 1.4	78 ± 1.8
hear	87.9 ± 1.5	66.6 ± 1.5	64.9 ± 1.2	67.3 ± 1.3
img	95.6 ± 0.3	96.1 ± 0.2	96.4 ± 0.2	96.3 ± 0.2
ionos	90.7 ± 1.4	90.9 ± 1.7	69.4 ± 1.2	69.4 ± 1.4
mok1	99.9 ± 0.2	94 ± 3	88 ± 5	84 ± 6
mok2	71.5 ± 1.9	70.6 ± 1.3	68 ± 1.3	67.6 ± 1
pima	69.2 ± 1.6	73.8 ± 0.8	74.2 ± 1.1	74.4 ± 0.7
survi	75.2 ± 0.9	74.8 ± 0.8	74.1 ± 0.7	74.8 ± 0.8
vote	87.9 ± 1.5	60.6 ± 0.9	60.4 ± 0.9	60.4 ± 0.9
vowel	83.2 ± 1.1	88.1 ± 0.8	88.7 ± 0.8	88.6 ± 1
wdbc	94.2 ± 0.7	95.5 ± 0.6	95.3 ± 0.6	95.7 ± 0.6

Table C.7: Performance - *ATA-LE* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	74.7 ± 1.4	74.7 ± 1.3	74.3 ± 1.4	73.8 ± 1.6
bala	93.9 ± 0.8	95.4 ± 0.7	95.2 ± 0.6	95.1 ± 0.7
band	76 ± 1	74 ± 1.3	73.3 ± 1.1	73.5 ± 1.2
bupa	71 ± 2	72.1 ± 1.2	72.4 ± 1.2	71.1 ± 1
cred	86.8 ± 0.7	87.3 ± 0.8	86.9 ± 1	87.2 ± 0.6
derma	96.1 ± 1.1	97 ± 0.7	96.9 ± 0.7	96.2 ± 1
ecoli	83.5 ± 1.9	84.6 ± 1.5	85.4 ± 1.3	84.6 ± 1.5
flare	82.4 ± 0.8	82.5 ± 0.7	81.9 ± 0.8	81.9 ± 0.7
glas	89.6 ± 1.8	90 ± 0.9	91 ± 1.5	91.6 ± 1.3
hear	83.9 ± 1.5	84.1 ± 1.4	84.6 ± 1.5	83.4 ± 1.4
img	96.8 ± 0.3	96.8 ± 0.2	96.8 ± 0.3	96.9 ± 0.18
ionos	90.1 ± 0.7	90 ± 1.2	90.6 ± 1	88.7 ± 1.7
mok1	99.8 ± 0.3	99.5 ± 0.4	100 ± 0	100 ± 0
mok2	84 ± 1.5	88 ± 0.9	88.6 ± 1.4	88.8 ± 1.3
pima	76.8 ± 1.1	76.3 ± 1.1	76.8 ± 1.2	76.8 ± 1.2
survi	75.4 ± 1.6	73.8 ± 1.2	74.3 ± 1	74.3 ± 1.3
vote	96.5 ± 0.7	95.8 ± 0.7	95.6 ± 0.7	95.8 ± 0.7
vowel	86.6 ± 1.1	88.6 ± 0.8	87.9 ± 1.2	88 ± 0.9
wdbc	97.2 ± 0.4	97.3 ± 0.4	97.3 ± 0.4	97.4 ± 0.3

Table C.8: Performance - *ATA-BE* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	75.5 ± 1.4	74.9 ± 1.9	75.8 ± 1.5	76 ± 1.6
bala	94.3 ± 0.8	94.8 ± 0.7	94.7 ± 0.8	95 ± 0.7
band	75.8 ± 0.9	76.4 ± 1.3	73.1 ± 1.1	73.8 ± 1.3
bupa	71.3 ± 1.9	71.6 ± 1.2	72 ± 1.3	72.6 ± 1.9
cred	87.1 ± 0.8	86.9 ± 0.8	86.6 ± 0.7	86.7 ± 0.7
derma	97 ± 0.5	96.8 ± 0.6	97 ± 1	96.3 ± 0.8
ecoli	83.8 ± 1.9	85.7 ± 0.9	85.2 ± 1.3	85.6 ± 1.4
flare	82.2 ± 0.5	83 ± 0.6	82.3 ± 0.5	82.4 ± 0.7
glas	88.2 ± 1.9	89.4 ± 1.5	90.8 ± 1.1	90.2 ± 1
hear	84.2 ± 1.1	84.1 ± 1.3	84.6 ± 1.7	83.6 ± 1.4
img	96.9 ± 0.2	96.8 ± 0.3	96.8 ± 0.3	96.8 ± 0.3
ionos	89 ± 1	91 ± 0.9	90 ± 1	88.9 ± 1.2
mok1	99.3 ± 0.6	100 ± 0	100 ± 0	100 ± 0
mok2	83.1 ± 1.1	89 ± 1	89 ± 1.2	89.4 ± 1.4
pima	75.8 ± 0.9	76.5 ± 1.1	76.2 ± 1.2	76.1 ± 1.2
survi	74.9 ± 0.9	74.6 ± 1	73.9 ± 0.7	74.4 ± 0.9
vote	96.4 ± 0.7	96.1 ± 0.7	96.1 ± 0.7	96.1 ± 0.6
vowel	86.2 ± 1	89.1 ± 0.8	88 ± 1	88.4 ± 0.9
wdbc	97.2 ± 0.4	97.2 ± 0.4	97.3 ± 0.4	97.4 ± 0.3

Table C.9: Performance - *EENCL-LG* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	70.9 ± 2.0	70.9 ± 1.6	72.9 ± 0.9	70.7 ± 1.9
bala	95.8 ± 0.6	95.7 ± 0.8	94.9 ± 0.5	95.5 ± 0.5
band	73.1 ± 1.3	72.5 ± 1.8	72.5 ± 1.9	69.3 ± 1.9
bupa	72.1 ± 1.6	71.3 ± 1.5	72.3 ± 1.4	71.7 ± 1.6
cred	86.1 ± 0.8	86.9 ± 0.5	87.0 ± 1.0	87.2 ± 0.8
derma	96.8 ± 0.9	95.9 ± 0.9	95.1 ± 1.1	95.8 ± 1.1
ecoli	86.6 ± 1.2	85.9 ± 0.9	87.2 ± 0.7	86.5 ± 0.8
flare	81.4 ± 0.8	81.3 ± 0.8	82.0 ± 0.8	82.1 ± 0.7
glas	92.8 ± 1.6	92.8 ± 1.0	92.4 ± 1.8	93.0 ± 1.0
hear	82.9 ± 1.3	79.5 ± 1.1	82.2 ± 1.8	81.5 ± 1.2
img	96.3 ± 0.2	96.6 ± 0.3	96.9 ± 0.3	97.0 ± 0.2
ionos	93.0 ± 1.0	92.7 ± 1.1	92.3 ± 1.1	92.9 ± 0.8
mok1	99.1 ± 0.9	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	84.6 ± 0.8	82.9 ± 2.0	85.1 ± 1.9	82.1 ± 1.9
pima	74.7 ± 1.0	74.0 ± 0.4	75.2 ± 0.8	74.3 ± 0.8
survi	73.9 ± 1.2	72.6 ± 1.0	72.5 ± 1.5	72.3 ± 1.2
vote	94.5 ± 0.5	95.4 ± 0.6	95.8 ± 0.5	95.9 ± 0.6
vowel	87.2 ± 0.8	88.6 ± 0.6	88.2 ± 0.9	89.9 ± 1.0
wdbc	96.2 ± 0.4	95.9 ± 0.5	95.8 ± 0.4	96.5 ± 0.8

Table C.10: Performance - *EENCL-BG* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	74.5 ± 1.3	74.8 ± 1.2	73.4 ± 1.6	74.1 ± 1.2
bala	95.8 ± 0.6	95.7 ± 0.7	94.9 ± 0.5	95.8 ± 0.5
band	74.2 ± 1.4	73.6 ± 1.9	74.2 ± 1.9	73.5 ± 1.4
bupa	71.7 ± 1.3	72.9 ± 1.4	71.7 ± 1.8	72.9 ± 1.1
cred	86.5 ± 0.8	87.2 ± 0.7	86.3 ± 0.7	86.7 ± 0.6
derma	97.2 ± 0.8	96.1 ± 0.9	96.2 ± 0.9	96.1 ± 1.1
ecoli	86.6 ± 1.1	85.9 ± 1.2	87.6 ± 1.0	88.1 ± 0.7
flare	81.9 ± 0.5	81.7 ± 0.5	81.4 ± 0.6	82.1 ± 0.5
glas	92.2 ± 1.3	92.4 ± 1.1	91.2 ± 1.6	92.6 ± 0.9
hear	83.7 ± 1.4	83.2 ± 1.1	83.1 ± 1.4	82.7 ± 0.9
img	96.0 ± 0.2	96.3 ± 0.2	96.6 ± 0.3	96.8 ± 0.3
ionos	93.7 ± 0.9	93.4 ± 0.9	92.3 ± 1.0	93.7 ± 0.7
mok1	99.1 ± 0.9	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	85.5 ± 0.9	82 ± 2	83 ± 2	82 ± 2
pima	75.3 ± 1.0	75.6 ± 1.1	76.2 ± 1.3	76.5 ± 1.2
survi	73.9 ± 0.8	73.8 ± 1.3	74.1 ± 1.0	75.2 ± 1.1
vote	95.5 ± 0.9	96.1 ± 0.7	95.8 ± 0.5	95.9 ± 0.6
vowel	87.4 ± 0.7	88.6 ± 0.7	88.3 ± 0.9	89.6 ± 0.8
wdbc	96.4 ± 0.5	96.4 ± 0.2	96.5 ± 0.4	96.8 ± 0.6

Table C.11: Performance - *Bagging* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	74.7 ± 1.6	75.9 ± 1.7	75.6 ± 1.6	74.7 ± 1.5
bala	95.7 ± 0.7	95.6 ± 0.6	95.5 ± 0.7	95.7 ± 0.6
band	72.2 ± 1.8	74.0 ± 1.9	74.7 ± 1.1	74.4 ± 1.1
bupa	72.1 ± 1.5	72.6 ± 1.3	73.3 ± 1.2	72.7 ± 1.4
cred	87.7 ± 0.8	87.4 ± 0.7	86.9 ± 0.6	86.9 ± 0.5
derma	97.5 ± 0.6	97.7 ± 0.6	97.6 ± 0.6	97.6 ± 0.6
ecoli	86.3 ± 1.1	87.2 ± 1.0	87.1 ± 1.0	86.9 ± 1.1
flare	81.9 ± 0.6	82.4 ± 0.6	82.3 ± 0.5	82.1 ± 0.6
glas	94.2 ± 0.9	95.6 ± 0.8	95.4 ± 0.8	94.8 ± 0.9
hear	82.5 ± 1.1	84.1 ± 1.2	84.2 ± 1.5	84.2 ± 1.4
img	96.6 ± 0.3	96.7 ± 0.3	97.0 ± 0.3	97.1 ± 0.3
ionos	90.7 ± 0.9	90.1 ± 1.1	89.6 ± 1.1	90.0 ± 1.1
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	87.8 ± 1.2	89.5 ± 1.2	91.4 ± 0.8	90.8 ± 1.0
pima	76.9 ± 0.8	76.6 ± 0.9	77.0 ± 1.0	77.0 ± 1.1
survi	74.3 ± 1.1	74.4 ± 1.5	74.6 ± 1.7	74.3 ± 1.3
vote	95.8 ± 0.7	96.3 ± 0.6	96.0 ± 0.6	96.1 ± 0.7
vowel	87.4 ± 0.7	90.8 ± 0.7	91.3 ± 0.6	91.3 ± 0.8
wdbc	96.9 ± 0.4	97.3 ± 0.4	97.5 ± 0.4	97.5 ± 0.4

Table C.12: Performance - *BagNoise* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	75.5 ± 1.1	76.0 ± 1.1	76.0 ± 1.0	75.7 ± 1.3
bala	91.1 ± 0.8	91.0 ± 0.7	90.8 ± 0.7	90.7 ± 0.7
band	75.6 ± 1.6	74.9 ± 0.9	74.9 ± 1.3	75.1 ± 1.4
bupa	65.6 ± 1.7	65.1 ± 1.7	62.9 ± 2.0	65 ± 2
cred	87.2 ± 0.5	87.4 ± 0.7	86.7 ± 0.7	87.1 ± 0.9
derma	97.6 ± 0.7	97.0 ± 0.7	97.3 ± 0.6	97.5 ± 0.6
ecoli	87.5 ± 1.0	87.2 ± 0.8	87.4 ± 0.8	87.5 ± 0.8
flare	82.2 ± 0.4	82.5 ± 0.5	82.1 ± 0.5	82.1 ± 0.6
glas	81.0 ± 1.2	81.6 ± 1.4	81.2 ± 1.4	81.6 ± 1.3
hear	83.9 ± 1.3	83.2 ± 1.2	83.7 ± 1.4	83.7 ± 1.3
img	93.4 ± 0.4	93.4 ± 0.3	93.3 ± 0.3	93.4 ± 0.3
ionos	92.4 ± 0.9	93.3 ± 0.6	92.7 ± 0.6	93.0 ± 0.6
mok1	99.3 ± 0.8	99.3 ± 0.8	99.3 ± 0.8	99.6 ± 0.4
mok2	89.1 ± 1.7	90.9 ± 1.7	91.3 ± 1.5	92.3 ± 1.6
pima	76.2 ± 1.0	75.9 ± 0.9	76.3 ± 0.8	76.5 ± 0.9
survi	74.6 ± 0.7	74.8 ± 0.7	74.6 ± 0.7	74.6 ± 0.7
vote	96.4 ± 0.6	96.5 ± 0.6	96.8 ± 0.5	96.5 ± 0.6
vowel	84.4 ± 1.0	85.7 ± 0.9	86.7 ± 0.7	86.5 ± 0.8
wdbc	96.3 ± 0.6	95.9 ± 0.5	96.1 ± 0.5	95.9 ± 0.5

Table C.13: Performance - *CVCv1* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	74.0 ± 1.0	74.8 ± 1.3	74.3 ± 1.6	73.4 ± 1.9
bala	94.5 ± 0.6	95.2 ± 0.6	95.3 ± 0.6	95.2 ± 0.6
band	71.3 ± 1.6	75.3 ± 1.4	72.5 ± 1.3	74.4 ± 1.5
bupa	71.3 ± 1.7	72.6 ± 1.2	72.1 ± 1.2	73.0 ± 1.3
cred	87.0 ± 0.6	86.9 ± 0.7	86.3 ± 0.6	86.9 ± 0.7
derma	97.3 ± 0.7	97.6 ± 0.6	97.3 ± 0.6	97.3 ± 0.6
ecoli	86.8 ± 0.8	87.1 ± 1.0	86.5 ± 1.0	86.8 ± 0.9
flare	82.7 ± 0.5	81.9 ± 0.6	81.7 ± 0.7	81.7 ± 0.7
glas	93.4 ± 1.2	94.4 ± 0.9	94.0 ± 1.0	94.6 ± 0.8
hear	84.2 ± 1.2	82.7 ± 1.0	83.7 ± 1.3	83.7 ± 1.5
img	96.41 ± 0.20	96.6 ± 0.2	96.8 ± 0.3	96.6 ± 0.2
ionos	87.7 ± 1.3	89.6 ± 1.2	89.6 ± 1.3	88.3 ± 1.0
mok1	97.4 ± 1.4	98.6 ± 0.9	99.3 ± 0.8	99.3 ± 0.8
mok2	81.0 ± 1.9	90.3 ± 0.9	91.4 ± 0.9	91.8 ± 0.9
pima	76.1 ± 1.1	77.0 ± 1.1	76.2 ± 1.3	76.6 ± 1.0
survi	74.1 ± 1.4	75.2 ± 1.5	73.8 ± 0.9	74.6 ± 1.0
vote	96.1 ± 0.6	96.3 ± 0.5	95.3 ± 0.8	96.3 ± 0.6
vowel	88.0 ± 1.0	90.9 ± 0.7	91.9 ± 0.5	92.2 ± 0.8
wdbc	97.4 ± 0.3	96.8 ± 0.5	97.4 ± 0.4	96.8 ± 0.5

Table C.14: Performance - *CVCv2* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	76.1 ± 1.6	75.4 ± 1.5	74.3 ± 1.2	77.0 ± 0.8
bala	95.9 ± 0.8	96.3 ± 0.7	96.3 ± 0.5	95.8 ± 0.7
band	74.2 ± 1.4	74.5 ± 1.2	75.3 ± 1.7	74.9 ± 1.3
bupa	72.6 ± 1.3	73.1 ± 1.4	72.7 ± 1.3	72.3 ± 1.5
cred	86.8 ± 0.5	86.8 ± 0.7	86.0 ± 0.8	86.0 ± 0.8
derma	98.0 ± 0.3	97.3 ± 0.5	97.5 ± 0.6	97.2 ± 0.6
ecoli	86.8 ± 0.9	86.6 ± 1.0	86.6 ± 1.1	86.3 ± 0.9
flare	82.5 ± 0.6	82.1 ± 0.5	81.8 ± 0.4	82.2 ± 0.5
glas	94.0 ± 1.1	95.0 ± 1.1	95.4 ± 0.9	95.4 ± 0.8
hear	84.2 ± 1.5	83.9 ± 1.3	84.1 ± 1.2	82.7 ± 1.2
img	96.9 ± 0.3	97.0 ± 0.3	97.03 ± 0.17	96.7 ± 0.3
ionos	89.7 ± 1.4	90.4 ± 1.3	91.0 ± 0.9	92.0 ± 1.0
mok1	98.5 ± 1.0	99.8 ± 0.3	100.00 ± 0.00	100.00 ± 0.00
mok2	88.5 ± 1.1	94 ± 2	94.9 ± 1.5	93.5 ± 1.7
pima	76.8 ± 1.0	76.8 ± 1.1	76.7 ± 0.8	76.1 ± 0.9
survi	74.1 ± 1.2	73.0 ± 1.0	73.6 ± 1.0	73.4 ± 1.2
vote	96.3 ± 0.5	96.3 ± 0.5	95.6 ± 0.4	95.0 ± 0.7
vowel	89.7 ± 0.9	92.7 ± 0.7	93.3 ± 0.6	92.9 ± 0.7
wdbc	96.7 ± 0.3	96.8 ± 0.3	95.9 ± 0.6	96.0 ± 0.5

Table C.15: Performance - *CVCv2.5* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	75.3 ± 1.3	77.0 ± 1.3	75.7 ± 1.5	75.6 ± 0.9
bala	95.9 ± 0.6	95.8 ± 0.6	95.8 ± 0.5	96.3 ± 0.5
band	73.5 ± 1.1	74.5 ± 1.2	74.2 ± 1.3	73.1 ± 1.5
bupa	72.4 ± 1.6	74.4 ± 1.6	72.1 ± 1.2	73.1 ± 1.4
cred	85.9 ± 0.7	86.7 ± 0.7	87.0 ± 0.8	86.5 ± 0.7
derma	97.3 ± 0.5	97.7 ± 0.5	97.5 ± 0.5	97.5 ± 0.5
ecoli	86.5 ± 1.0	86.6 ± 0.9	86.6 ± 1.1	86.2 ± 1.0
flare	81.5 ± 0.7	81.5 ± 0.7	81.5 ± 0.8	81.6 ± 0.6
glas	94.8 ± 1.2	95.4 ± 1.0	94.6 ± 0.9	94.4 ± 1.1
hear	84.2 ± 1.3	83.6 ± 1.3	83.7 ± 1.5	85.3 ± 1.3
img	96.9 ± 0.3	96.8 ± 0.2	96.65 ± 0.20	97.0 ± 0.2
ionos	91.4 ± 1.0	90.4 ± 1.0	91.0 ± 1.2	90.3 ± 1.1
mok1	99.4 ± 0.6	99.6 ± 0.4	100.00 ± 0.00	99.4 ± 0.6
mok2	91.8 ± 0.7	94.6 ± 1.2	94.3 ± 1.2	94.9 ± 1.2
pima	75.8 ± 1.0	76.6 ± 1.0	77.1 ± 0.9	76.3 ± 1.2
survi	73.6 ± 1.1	74.8 ± 1.3	74.1 ± 1.2	74.3 ± 1.3
vote	95.6 ± 0.6	96.4 ± 0.6	95.8 ± 0.5	96.1 ± 0.6
vowel	89.8 ± 0.8	92.4 ± 0.8	92.6 ± 0.5	92.3 ± 0.7
wdbc	96.7 ± 0.2	96.7 ± 0.4	96.9 ± 0.4	97.1 ± 0.3

Table C.16: Performance - *CVCv3* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	75.7 ± 1.4	76.1 ± 1.1	76.2 ± 1.2	76.0 ± 1.3
bala	96.0 ± 1.0	95.6 ± 0.8	95.7 ± 0.8	95.5 ± 0.6
band	74.5 ± 1.7	73.8 ± 1.3	73.8 ± 1.3	74.9 ± 1.1
bupa	71.3 ± 1.0	73.4 ± 1.2	73.7 ± 1.3	73.3 ± 1.2
cred	86.6 ± 0.8	86.8 ± 0.9	87.2 ± 0.9	86.9 ± 0.8
derma	97.0 ± 0.6	97.3 ± 0.5	97.7 ± 0.4	97.6 ± 0.5
ecoli	86.5 ± 0.9	86.8 ± 1.0	86.5 ± 1.0	86.2 ± 1.0
flare	81.4 ± 0.7	81.8 ± 0.8	81.8 ± 0.7	82.1 ± 0.6
glas	94.4 ± 1.0	94.4 ± 0.9	94.4 ± 0.9	94.6 ± 0.9
hear	84.4 ± 1.6	84.4 ± 1.4	84.7 ± 1.4	84.4 ± 1.4
img	96.6 ± 0.2	96.9 ± 0.2	96.9 ± 0.3	97.0 ± 0.3
ionos	89.9 ± 1.2	90.6 ± 1.0	91.0 ± 0.9	91.1 ± 0.9
mok1	97.9 ± 1.1	97.4 ± 1.4	100.00 ± 0.00	99.4 ± 0.6
mok2	93.6 ± 0.8	95.8 ± 0.9	94.6 ± 1.5	95.3 ± 1.3
pima	77.3 ± 1.1	76.8 ± 1.1	76.5 ± 1.1	76.6 ± 1.1
survi	73.3 ± 1.2	74.1 ± 1.3	74.4 ± 1.2	73.9 ± 1.2
vote	96.0 ± 0.4	96.1 ± 0.5	96.0 ± 0.6	96.1 ± 0.6
vowel	90.8 ± 0.7	92.9 ± 0.5	93.5 ± 0.6	93.2 ± 0.6
wdbc	96.7 ± 0.3	96.7 ± 0.3	96.8 ± 0.4	97.1 ± 0.3

Table C.17: Performance - *DP* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	71.8 ± 1.3	68.1 ± 1.1	66.2 ± 1.8	62.1 ± 1.5
bala	92.9 ± 0.8	87.5 ± 1.2	87.4 ± 1.2	87 ± 1
band	71.1 ± 1.2	65.3 ± 1.2	65.3 ± 1.2	65.3 ± 1.2
bupa	73.4 ± 1.4	64.7 ± 1.8	59.7 ± 0.9	59.1 ± 1.2
cred	86.8 ± 0.7	86.9 ± 0.7	87.2 ± 0.5	86.6 ± 0.7
derma	97.3 ± 0.5	93.5 ± 1.1	89.2 ± 1.9	83.5 ± 1.8
ecoli	86.3 ± 1.2	84 ± 1.4	79.9 ± 1.9	76.3 ± 1.2
flare	81.7 ± 0.6	81.9 ± 0.6	81.4 ± 0.5	81.3 ± 0.5
glas	88.2 ± 1.4	79.6 ± 1.5	78 ± 2	63 ± 3
hear	84.2 ± 1.4	83.2 ± 1.4	82.7 ± 1.6	82.5 ± 1.3
img	85.9 ± 1.5	79.6 ± 1.8	68.6 ± 1.9	63.4 ± 1.3
ionos	95.5 ± 0.3	94.06 ± 0.18	91.8 ± 0.4	89.6 ± 0.5
mok1	97.4 ± 1.4	72 ± 2	66.4 ± 1.4	65 ± 2
mok2	71.6 ± 1.4	66.5 ± 1	66.5 ± 1	66.5 ± 1
pima	75.5 ± 1	75.7 ± 1	74.8 ± 0.8	71.9 ± 1
survi	74.3 ± 1	74.1 ± 0.8	74.8 ± 0.7	74.8 ± 0.7
vote	95.9 ± 0.6	94.6 ± 0.8	92.9 ± 0.4	91 ± 0.8
vowel	81.1 ± 1	66 ± 2	48.8 ± 1.8	36.4 ± 1.3
wdbc	97.2 ± 0.3	96.6 ± 0.6	94.8 ± 0.8	94.1 ± 0.7

Table C.18: Performance - *DPR* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	73 ± 2	69.7 ± 1.4	64.8 ± 1.4	59.5 ± 0.8
bala	93.3 ± 1	87.5 ± 1.1	87.4 ± 1.1	87 ± 0.9
band	70.2 ± 1.7	64.7 ± 1	65.3 ± 1.2	65.3 ± 1.2
bupa	73 ± 1.7	67.3 ± 1.8	60.6 ± 1.2	59.7 ± 0.8
cred	87.5 ± 0.7	87.5 ± 0.6	87.2 ± 0.5	87.2 ± 0.7
derma	97.3 ± 0.6	97.2 ± 0.8	93.8 ± 0.9	89.3 ± 0.7
ecoli	85.3 ± 1	83.1 ± 2	79.7 ± 1.7	76.6 ± 1.4
flare	81.5 ± 0.5	81.3 ± 0.6	81.4 ± 0.5	81.3 ± 0.5
glas	89 ± 1.1	81 ± 1.7	77.6 ± 1.9	65 ± 2
hear	83.6 ± 1	83.2 ± 1.5	83.1 ± 1.4	82.5 ± 1.5
img	88.1 ± 1.2	81 ± 1.6	67 ± 1.7	63.4 ± 1.3
ionos	95.8 ± 0.2	93.9 ± 0.3	92.39 ± 0.17	90.1 ± 0.5
mok1	99.8 ± 0.3	78 ± 2	67 ± 1.7	65.1 ± 1.5
mok2	67.9 ± 1.1	66.5 ± 1	66.5 ± 1	66.5 ± 1
pima	76.8 ± 1.1	76.3 ± 0.9	74.3 ± 0.8	70.7 ± 0.9
survi	74.6 ± 1	74.9 ± 1.2	74.8 ± 0.7	74.8 ± 0.7
vote	95.3 ± 0.6	94.5 ± 0.8	93.1 ± 0.4	91.3 ± 0.8
vowel	81.1 ± 0.9	65.8 ± 1.7	48.1 ± 1.2	35.3 ± 1
wdbc	97.4 ± 0.2	96.6 ± 0.6	95.4 ± 0.8	94 ± 0.7

Table C.19: Performance - *OP* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	72.8 ± 1.1	71.4 ± 1.2	64 ± 1.3	62.4 ± 1.6
bala	93.6 ± 0.6	90.4 ± 0.8	87.2 ± 1.1	87 ± 1.1
band	70 ± 2	67.1 ± 1.2	67.1 ± 1.2	64.6 ± 0.9
bupa	72.6 ± 1.3	68 ± 2	63 ± 2	57.1 ± 1.9
cred	86.3 ± 0.5	86.5 ± 0.6	86.3 ± 0.7	87.2 ± 0.5
derma	96.8 ± 0.5	93.5 ± 1.3	92.1 ± 1	79.2 ± 1.4
ecoli	85.7 ± 1.2	84.1 ± 1.8	78.8 ± 1.4	74 ± 3
flare	81.7 ± 0.5	81.6 ± 0.5	81.9 ± 0.5	81.4 ± 0.4
glas	91.6 ± 1.4	83.4 ± 1.6	75 ± 2	59 ± 4
hear	84.4 ± 1.7	82.9 ± 1.4	77.3 ± 2	78.6 ± 1.8
img	85.4 ± 1.1	83 ± 1.3	78 ± 2	69 ± 2
ionos	96.1 ± 0.3	94.5 ± 0.3	91.4 ± 0.5	90.7 ± 0.5
mok1	99.3 ± 0.5	94 ± 2	68.8 ± 1.9	62 ± 3
mok2	78.5 ± 1.7	67.1 ± 1.6	66.5 ± 1	66.5 ± 1
pima	75.9 ± 1.2	74.9 ± 1.2	75.1 ± 0.9	71.2 ± 0.9
survi	73.8 ± 0.8	74.4 ± 1.4	74.1 ± 0.9	73.8 ± 1.2
vote	96 ± 0.5	95 ± 0.6	93.5 ± 0.6	91 ± 0.5
vowel	83 ± 1	68.9 ± 1.2	51.2 ± 1.1	33.2 ± 1.6
wdbc	97 ± 0.3	96.3 ± 0.5	95.3 ± 0.7	94.5 ± 0.8

Table C.20: Performance - *OPR* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	72.4 ± 1.3	72 ± 1.3	66.4 ± 1.6	62.9 ± 1.4
bala	94.5 ± 0.6	91.2 ± 1	87.3 ± 1.4	86.8 ± 0.9
band	71.8 ± 1.5	69.1 ± 1.1	66.6 ± 1.8	67.1 ± 1
bupa	72.9 ± 1.4	69.1 ± 1.6	62.3 ± 2	59 ± 2
cred	86.3 ± 0.5	87 ± 0.8	86.1 ± 0.9	86.3 ± 0.8
derma	96.8 ± 0.5	94.1 ± 1.1	92.5 ± 1.2	82 ± 2
ecoli	85.4 ± 1	85.7 ± 1.5	81.2 ± 1.2	76.9 ± 1.8
flare	81.4 ± 0.5	81.3 ± 0.6	81.9 ± 0.5	81.4 ± 0.4
glas	93 ± 1.3	86.2 ± 1.6	78 ± 3	70 ± 2
hear	84.1 ± 1.4	84.4 ± 1.6	78.8 ± 0.9	74.6 ± 2
img	88.3 ± 1	83.3 ± 1.5	80.6 ± 1.8	71 ± 3
ionos	95.9 ± 0.3	94.8 ± 0.4	92.4 ± 0.7	90.9 ± 0.6
mok1	97.4 ± 1.5	99.8 ± 0.3	73 ± 2	59 ± 3
mok2	81.4 ± 1.3	70.4 ± 1.3	66.5 ± 1	66.6 ± 1
pima	75.7 ± 1.1	75.6 ± 1.2	74.9 ± 0.6	70.1 ± 0.8
survi	74.3 ± 1.2	74.1 ± 1.3	73.6 ± 0.9	74.3 ± 1
vote	95.9 ± 0.6	95.6 ± 0.6	94.9 ± 0.8	93.8 ± 0.7
vowel	85.5 ± 0.8	72.1 ± 1	59.6 ± 0.9	42 ± 2
wdbc	96.7 ± 0.5	96.7 ± 0.6	95 ± 0.8	94 ± 0.5

Table C.21: Performance - *Boosting 3* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	74.4 ± 1.2	–	–	–
bala	94.8 ± 0.7	–	–	–
band	73.6 ± 1.3	–	–	–
bupa	70.7 ± 1.2	–	–	–
cred	86.5 ± 0.5	–	–	–
derma	97.3 ± 0.6	–	–	–
ecoli	86.8 ± 0.6	–	–	–
flare	81.7 ± 0.4	–	–	–
glas	92.8 ± 1.1	–	–	–
hear	81.7 ± 1.4	–	–	–
img	95 ± 0.4	–	–	–
ionos	88.9 ± 1.4	–	–	–
mok1	98.5 ± 1.4	–	–	–
mok2	87 ± 2	–	–	–
pima	75.7 ± 0.7	–	–	–
survi	74.1 ± 1	–	–	–
vote	94.9 ± 0.6	–	–	–
vowel	85.7 ± 0.7	–	–	–
wdbc	97 ± 0.4	–	–	–

Table C.22: Performance - *Adaboost* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	71.8 ± 1.8	73.2 ± 1.6	71.4 ± 1.5	73.8 ± 1.1
bala	94.5 ± 0.8	95.3 ± 0.5	96.1 ± 0.4	95.7 ± 0.5
band	70.2 ± 1.8	72.9 ± 1.7	73.5 ± 1.4	71.1 ± 1.5
bupa	70.4 ± 1.7	73.3 ± 1.4	73 ± 1.8	72.6 ± 1.4
cred	84.9 ± 1.4	84.2 ± 0.9	84.5 ± 0.8	85.1 ± 0.9
derma	98 ± 0.5	97.3 ± 0.5	97.5 ± 0.6	97.8 ± 0.5
ecoli	85.9 ± 1.2	84.7 ± 1.4	86 ± 1.3	85.7 ± 1.4
flare	81.7 ± 0.6	81.1 ± 0.7	81.1 ± 0.8	81.1 ± 0.7
glas	94 ± 1.1	95.6 ± 0.8	95.8 ± 0.9	96.2 ± 0.8
hear	80.5 ± 1.8	81.2 ± 1.4	82 ± 1.9	82.2 ± 1.8
img	96.8 ± 0.2	97.3 ± 0.3	97.29 ± 0.19	97.3 ± 0.2
ionos	88.3 ± 1.3	89.4 ± 0.8	91.4 ± 0.8	91.6 ± 0.7
mok1	100 ± 0	100 ± 0	100 ± 0	100 ± 0
mok2	77 ± 2	79 ± 2	81 ± 2	83 ± 2
pima	75.7 ± 1	75.5 ± 0.9	74.8 ± 1	73.3 ± 1
survi	75.4 ± 1.6	74.3 ± 1.4	74.3 ± 1.5	73 ± 2
vote	95.6 ± 0.9	96.3 ± 0.7	95.9 ± 0.7	96 ± 0.7
vowel	88.4 ± 0.9	94.8 ± 0.7	96.1 ± 0.7	97 ± 0.6
wdbc	95.7 ± 0.6	95.7 ± 0.7	96.3 ± 0.5	96.7 ± 0.9

Table C.23: Performance - *Aveboost* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	73.4 ± 1.3	75.2 ± 1.0	75.5 ± 1.1	76.3 ± 1.0
bala	95.8 ± 0.4	96.1 ± 0.6	96.2 ± 0.4	96.5 ± 0.5
band	72.4 ± 1.6	74.4 ± 1.4	75.3 ± 1.3	75.6 ± 1.5
bupa	69.4 ± 1.7	71.9 ± 1.3	71.9 ± 1.2	72.9 ± 1.4
cred	85.9 ± 0.7	86.4 ± 0.4	86.6 ± 0.8	85.5 ± 0.9
derma	97.6 ± 0.7	97.5 ± 0.5	97.9 ± 0.5	97.2 ± 0.7
ecoli	85.3 ± 1.0	86.5 ± 1.2	86.2 ± 1.2	86.0 ± 1.1
flare	81.8 ± 0.8	82.0 ± 0.7	82.4 ± 0.6	80.7 ± 1.1
glas	94.2 ± 0.8	96.2 ± 1.0	96.6 ± 0.8	97.0 ± 0.8
hear	83.2 ± 1.5	84.9 ± 1.3	83.9 ± 1.4	83.6 ± 1.5
img	96.8 ± 0.2	97.1 ± 0.3	97.3 ± 0.3	97.5 ± 0.2
ionos	89.4 ± 1.2	90.4 ± 0.8	91.4 ± 1.0	91.6 ± 0.9
mok1	99.88 ± 0.13	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	80 ± 3	84.5 ± 1.8	87.9 ± 1.1	89.5 ± 1.1
pima	76.5 ± 1.1	76.6 ± 1.0	76.0 ± 1.1	76.6 ± 1.0
survi	75.1 ± 1.2	74.4 ± 1.2	74.8 ± 1.2	74.6 ± 1.1
vote	96.0 ± 0.6	96.3 ± 0.6	96.0 ± 0.7	95.8 ± 0.8
vowel	88.1 ± 1.0	93.5 ± 0.5	95.8 ± 0.6	96.4 ± 0.6
wdbc	95.6 ± 0.4	96.5 ± 0.4	95.8 ± 0.6	96.0 ± 0.5

Table C.24: Performance - *Aveboost 2* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	73.8 ± 1.6	74.9 ± 1.3	75.7 ± 1.2	75.7 ± 1.2
bala	94.5 ± 0.5	95.0 ± 0.4	95.3 ± 0.5	95.0 ± 0.5
band	70.9 ± 0.7	70.2 ± 0.8	72.2 ± 1.1	72.4 ± 1.0
bupa	70.9 ± 1.5	70.7 ± 1.5	70.6 ± 1.4	71.7 ± 1.5
cred	86.3 ± 0.7	86.8 ± 0.6	86.8 ± 0.5	86.8 ± 0.5
derma	97.3 ± 0.9	97.9 ± 0.5	97.5 ± 0.5	97.5 ± 0.5
ecoli	86.5 ± 1.1	87.1 ± 1.0	87.5 ± 1.1	87.2 ± 0.9
flare	82.6 ± 0.6	82.7 ± 0.4	82.7 ± 0.4	82.5 ± 0.4
glas	93.4 ± 1.2	94.2 ± 0.8	93.8 ± 0.8	94.2 ± 0.8
hear	82.7 ± 1.4	83.2 ± 1.5	83.6 ± 1.3	83.4 ± 1.3
img	96.4 ± 0.3	96.6 ± 0.3	96.7 ± 0.3	96.7 ± 0.2
ionos	88.9 ± 1.2	88.9 ± 1.3	88.9 ± 1.2	89.0 ± 1.1
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	80.4 ± 1.8	83.8 ± 1.1	85.1 ± 1.2	86.0 ± 1.3
pima	76.3 ± 0.8	76.6 ± 0.8	76.5 ± 1.0	76.2 ± 0.9
survi	72.6 ± 1.1	73.0 ± 1.0	72.8 ± 1.1	72.8 ± 1.1
vote	95.5 ± 0.6	95.8 ± 0.7	95.9 ± 0.6	95.9 ± 0.7
vowel	84.7 ± 1.3	86.9 ± 1.0	88.4 ± 1.0	89.0 ± 1.1
wdbc	97.1 ± 0.4	97.5 ± 0.4	97.3 ± 0.4	97.3 ± 0.4

Table C.25: Performance - *TCAv1* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	70.7 ± 1.9	72.3 ± 1.4	71.6 ± 1.8	70.6 ± 1.7
bala	90 ± 4	93.3 ± 0.9	92.9 ± 0.9	91 ± 2
band	72.2 ± 1.7	74.9 ± 1.5	71 ± 2	69 ± 2
bupa	70.9 ± 1.0	70.3 ± 1.6	71.3 ± 1.6	70.7 ± 1.9
cred	85.7 ± 0.8	83.0 ± 1.0	83.5 ± 1.1	83.4 ± 1.1
derma	96.1 ± 0.5	90 ± 4	92 ± 2	82 ± 5
ecoli	85.4 ± 1.2	85.6 ± 1.1	85.4 ± 1.5	84.0 ± 1.4
flare	81.9 ± 0.7	80.3 ± 1.0	79.7 ± 0.9	80.1 ± 1.0
glas	83 ± 5	88.8 ± 1.5	87 ± 2	87 ± 4
hear	80.3 ± 1.7	81.7 ± 2.0	82.0 ± 1.8	80.7 ± 1.9
img	94.8 ± 0.5	95.7 ± 0.3	95.7 ± 0.3	95.8 ± 0.3
ionos	87.9 ± 1.2	87.4 ± 0.8	86.1 ± 1.0	79 ± 5
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	75 ± 2	78 ± 2	77 ± 2	77 ± 2
pima	75.4 ± 0.8	75.5 ± 1.1	73.5 ± 0.9	73.7 ± 0.8
survi	73.0 ± 1.5	73.0 ± 1.3	71.3 ± 1.8	70.2 ± 1.7
vote	94.6 ± 0.7	95.1 ± 0.6	94.1 ± 0.9	92 ± 3
vowel	87.5 ± 1.1	89.2 ± 1.2	84 ± 3	79 ± 4
wdbc	91 ± 4	94.2 ± 0.7	94.4 ± 0.7	92.8 ± 1.7

Table C.26: Performance - *TCAv2* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	73.8 ± 1.9	71 ± 2	71.5 ± 1.7	71 ± 2
bala	94.3 ± 0.8	93.5 ± 0.9	91.1 ± 1.8	93.8 ± 0.5
band	72.2 ± 1.7	73.1 ± 1.8	72.2 ± 1.8	70 ± 2
bupa	70.9 ± 1.0	71.4 ± 1.9	70.9 ± 1.5	69.7 ± 1.5
cred	85.7 ± 0.8	82.8 ± 1.1	84.0 ± 1.0	84.8 ± 0.9
derma	95.9 ± 0.6	91 ± 3	88 ± 3	90 ± 3
ecoli	84.7 ± 1.4	85.6 ± 1.1	84.6 ± 1.5	85.1 ± 1.1
flare	81.1 ± 0.6	80.6 ± 0.8	80.5 ± 0.9	80.3 ± 1.1
glas	85 ± 5	84 ± 4	90 ± 2	81 ± 7
hear	80.3 ± 1.7	81.9 ± 1.5	82.7 ± 1.4	81.9 ± 1.6
img	96.3 ± 0.4	95.9 ± 0.6	94.5 ± 1.1	94.0 ± 0.8
ionos	88.6 ± 1.3	88.0 ± 1.6	87.0 ± 1.8	85 ± 2
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	76 ± 2	77.3 ± 1.9	76.8 ± 1.7	77.4 ± 1.5
pima	76.0 ± 1.2	75.2 ± 1.0	75.3 ± 1.3	75.7 ± 1.4
survi	73.8 ± 1.8	73.0 ± 1.7	74.1 ± 1.7	71.8 ± 1.7
vote	94.5 ± 0.8	95.0 ± 0.9	93.0 ± 1.5	91.4 ± 1.6
vowel	88.0 ± 0.8	89.2 ± 1.0	87.0 ± 1.0	82.7 ± 1.9
wdbc	95.0 ± 1.1	95.3 ± 0.5	94.2 ± 0.6	92.4 ± 1.4

Table C.27: Performance - *Aggreboost* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	72.3 ± 1.9	74.0 ± 1.4	74.8 ± 1.4	75.5 ± 1.2
bala	94.4 ± 0.7	95.4 ± 0.5	95.6 ± 0.5	96.1 ± 0.6
band	69 ± 2	72.5 ± 1.7	73.1 ± 2.0	73.5 ± 1.3
bupa	71 ± 2	71.7 ± 1.6	71.4 ± 1.7	72.7 ± 1.7
cred	86.5 ± 0.7	85.2 ± 1.1	85.7 ± 0.7	85.5 ± 0.8
derma	97.0 ± 0.5	96.6 ± 0.6	97.0 ± 0.6	96.6 ± 0.5
ecoli	85.7 ± 1.4	87.4 ± 1.1	87.1 ± 1.0	87.8 ± 0.9
flare	81.9 ± 0.9	81.8 ± 0.7	82.2 ± 0.5	82.0 ± 0.6
glas	93.4 ± 0.8	96.2 ± 0.8	96.2 ± 0.8	96.4 ± 0.8
hear	82.7 ± 1.4	81.9 ± 1.5	84.6 ± 1.2	83.9 ± 1.2
img	96.6 ± 0.3	96.9 ± 0.2	97.2 ± 0.3	97.3 ± 0.3
ionos	90.3 ± 0.8	91.7 ± 1.2	91.6 ± 0.9	92.3 ± 0.9
mok1	99.6 ± 0.4	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	74.6 ± 1.1	80.9 ± 1.9	85.3 ± 1.8	86.3 ± 1.1
pima	74.3 ± 1.5	74.9 ± 1.7	75.5 ± 1.3	75.7 ± 1.2
survi	74.1 ± 1.5	73.1 ± 0.9	73.8 ± 1.7	73.9 ± 1.4
vote	94.5 ± 0.9	94.6 ± 0.7	94.9 ± 0.7	95.5 ± 0.8
vowel	86.9 ± 1.2	94.6 ± 0.6	96.9 ± 0.6	97.5 ± 0.5
wdbc	96.6 ± 0.6	96.5 ± 0.6	96.8 ± 0.6	96.7 ± 0.6

Table C.28: Performance - *Conserboost* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	74.8 ± 1.3	74.7 ± 0.7	74.7 ± 0.9	75.1 ± 1.0
bala	94.7 ± 0.8	95.4 ± 0.6	95.7 ± 0.6	96.2 ± 0.7
band	71.8 ± 2.0	74.0 ± 1.1	74.5 ± 1.7	75.8 ± 1.5
bupa	72.0 ± 1.4	71.3 ± 1.4	72.9 ± 1.1	73.4 ± 1.1
cred	86.5 ± 0.6	87.1 ± 0.7	85.8 ± 0.7	86.0 ± 0.7
derma	96.9 ± 0.8	97.9 ± 0.6	97.9 ± 0.6	97.6 ± 0.7
ecoli	85.4 ± 1.2	86.2 ± 1.2	86.9 ± 1.2	87.8 ± 1.1
flare	82.1 ± 1.0	82.2 ± 0.9	82.8 ± 0.6	82.4 ± 0.6
glas	94.8 ± 0.9	95.2 ± 1.1	96.2 ± 1.1	97.0 ± 0.7
hear	83.2 ± 1.4	83.2 ± 1.2	83.1 ± 1.6	83.9 ± 0.9
img	96.5 ± 0.3	97.3 ± 0.2	97.2 ± 0.2	97.2 ± 0.2
ionos	89.4 ± 1.0	91.3 ± 0.8	92.4 ± 1.0	91.9 ± 0.8
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	77.1 ± 1.8	83 ± 2	83.8 ± 1.7	86.6 ± 1.4
pima	75.5 ± 1.2	75.9 ± 1.3	76.7 ± 1.2	76.2 ± 1.2
survi	75.6 ± 1.1	74.4 ± 1.5	72.8 ± 1.3	73.3 ± 1.5
vote	96.0 ± 0.6	95.6 ± 0.7	95.4 ± 0.8	95.5 ± 0.8
vowel	88.8 ± 1.1	94.3 ± 0.6	96.6 ± 0.6	97.3 ± 0.6
wdbc	97.0 ± 0.6	96.5 ± 0.7	96.4 ± 0.6	96.3 ± 0.5

Table C.29: Performance - *Inverseboost* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	73.9 ± 1.2	73.2 ± 1.5	68.5 ± 1.6	63.8 ± 1.9
bala	92.6 ± 0.8	91.1 ± 0.8	89.4 ± 1.0	86.1 ± 1.1
band	69.1 ± 1.5	68.5 ± 1.7	66.4 ± 1.7	66.4 ± 1.6
bupa	70.7 ± 1.4	66.3 ± 1.7	62.6 ± 1.9	61.3 ± 1.7
cred	86.8 ± 0.7	87.3 ± 0.6	87.3 ± 0.6	87.3 ± 0.6
derma	96.2 ± 1.2	94.9 ± 1.4	93.8 ± 1.4	93.8 ± 1.3
ecoli	86.2 ± 1.3	84.3 ± 1.5	82.2 ± 1.3	81.6 ± 1.4
flare	81.8 ± 0.6	81.5 ± 0.5	81.4 ± 0.4	81.3 ± 0.5
glas	90.2 ± 1.2	84.6 ± 1.2	79.2 ± 1.8	73 ± 3
hear	83.2 ± 1.4	81.7 ± 1.1	81.2 ± 1.7	80.8 ± 1.3
img	95.1 ± 0.2	94.4 ± 0.2	93.7 ± 0.2	93.2 ± 0.3
ionos	87.4 ± 0.9	84.3 ± 0.9	81.4 ± 1.7	80 ± 2
mok1	98.8 ± 0.9	96.9 ± 1.4	91.8 ± 1.9	89.0 ± 1.7
mok2	74 ± 2	69.4 ± 1.3	66.6 ± 1.4	66.4 ± 1.4
pima	75.5 ± 1.3	72.5 ± 1.1	68.1 ± 1.6	66.5 ± 1.4
survi	74.6 ± 1.2	74.1 ± 0.8	74.8 ± 0.7	74.8 ± 0.7
vote	95.6 ± 0.6	95.4 ± 0.5	95.6 ± 0.5	95.0 ± 0.4
vowel	80.6 ± 0.4	74.9 ± 0.9	69.6 ± 1.2	64.3 ± 1.3
wdbc	97.0 ± 0.5	96.9 ± 0.3	97.3 ± 0.4	97.2 ± 0.4

Table C.30: Performance - *ARCx4* of *MF* networks

Database	3-net	9-net	20-net	40-net
aritm	73.0 ± 1.4	73.3 ± 1.3	70.3 ± 1.1	71.7 ± 0.8
bala	93.8 ± 0.7	95.9 ± 0.5	96.0 ± 0.5	96.1 ± 0.5
band	71 ± 2	72.2 ± 1.3	66.2 ± 1.7	72.0 ± 1.4
bupa	69.4 ± 1.2	68.1 ± 1.7	61 ± 3	67.7 ± 1.2
cred	86.2 ± 0.8	85.0 ± 0.9	84.1 ± 1.0	84.3 ± 0.9
derma	97.0 ± 0.6	97.5 ± 0.7	97.0 ± 0.7	97.2 ± 0.7
ecoli	84.4 ± 1.3	83.2 ± 1.6	84.7 ± 1.2	85.3 ± 1.2
flare	80.6 ± 0.8	80.1 ± 0.7	74.1 ± 1.1	77.9 ± 0.9
glas	91.4 ± 0.8	95.0 ± 1.0	96.0 ± 0.8	96.2 ± 0.9
hear	84.1 ± 0.9	82.7 ± 0.9	79.8 ± 1.0	81.2 ± 1.4
img	96.4 ± 0.3	97.1 ± 0.3	97.2 ± 0.2	97.4 ± 0.2
ionos	89.0 ± 1.0	90.3 ± 0.9	91.6 ± 1.1	91.0 ± 1.0
mok1	98.9 ± 0.9	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	82.1 ± 1.5	82.6 ± 1.5	82 ± 2	87.3 ± 1.7
pima	76.3 ± 0.9	76.3 ± 0.9	69.9 ± 0.6	72.4 ± 1.0
survi	74.1 ± 1.5	73.9 ± 1.8	67.2 ± 1.7	71.3 ± 1.5
vote	96.5 ± 0.6	95.8 ± 0.4	95.3 ± 0.6	95.3 ± 0.5
vowel	84.5 ± 1.0	93.8 ± 0.6	95.6 ± 0.7	97.0 ± 0.5
wdbc	96.1 ± 0.5	96.6 ± 0.6	96.4 ± 0.6	96.5 ± 0.5

Table C.31: Performance - *SE* of *MF* networks with *Static reordering*

Database	3-net	9-net	20-net	40-net
aritm	74.7 ± 1.6	75.7 ± 1.7	75.7 ± 1.4	75.6 ± 1.6
bala	95.4 ± 0.7	95.4 ± 0.8	95.7 ± 0.8	95.8 ± 0.7
band	75.1 ± 1.3	74.5 ± 1.3	74.0 ± 1.2	74.0 ± 1.4
bupa	73.1 ± 1.2	72.3 ± 1.2	72.4 ± 1.2	72.7 ± 1.2
cred	86.6 ± 0.7	87.1 ± 0.6	87.0 ± 0.7	86.9 ± 0.6
derma	97.3 ± 0.6	97.5 ± 0.7	97.6 ± 0.7	97.6 ± 0.7
ecoli	86.9 ± 1.0	86.9 ± 0.9	87.1 ± 1.0	87.2 ± 1.0
flare	82.0 ± 0.7	82.1 ± 0.6	82.1 ± 0.6	82.0 ± 0.6
glas	92.8 ± 0.9	95.2 ± 0.7	95.4 ± 0.9	95.4 ± 0.9
hear	83.7 ± 1.2	84.2 ± 1.3	83.9 ± 1.3	84.1 ± 1.4
img	96.7 ± 0.2	96.9 ± 0.3	96.8 ± 0.3	97.0 ± 0.3
ionos	89.4 ± 1.3	89.4 ± 1.1	89.4 ± 1.2	89.6 ± 1.0
mok1	98.6 ± 0.9	98.6 ± 0.9	98.6 ± 0.9	98.6 ± 0.9
mok2	89.9 ± 1.2	91.6 ± 0.7	92.4 ± 0.8	91.6 ± 0.7
pima	76.3 ± 1.1	76.1 ± 1.0	76.6 ± 1.0	77.0 ± 1.0
survi	75.1 ± 1.3	74.3 ± 1.3	74.6 ± 1.5	74.9 ± 1.4
vote	96.0 ± 0.7	95.9 ± 0.6	96.0 ± 0.6	96.1 ± 0.6
vowel	88.6 ± 0.6	90.8 ± 0.8	91.4 ± 0.8	91.6 ± 0.8
wdbc	97.1 ± 0.4	97.4 ± 0.4	97.3 ± 0.4	97.3 ± 0.4

Table C.32: Performance - *SE* of *MF* networks with *Dynamic reordering*

Database	3-net	9-net	20-net	40-net
aritm	75.4 ± 1.8	75.3 ± 1.6	75.6 ± 1.7	75.6 ± 1.6
bala	95.9 ± 0.8	96.1 ± 0.8	95.9 ± 0.6	95.8 ± 0.7
band	75.3 ± 1.3	75.6 ± 1.2	75.6 ± 1.1	75.1 ± 1.2
bupa	72.6 ± 1.2	72.7 ± 1.1	72.7 ± 1.0	73.1 ± 1.1
cred	86.5 ± 0.6	86.5 ± 0.6	86.6 ± 0.7	86.7 ± 0.7
derma	97.5 ± 0.8	97.6 ± 0.7	97.7 ± 0.6	97.6 ± 0.7
ecoli	86.6 ± 0.8	86.0 ± 0.7	86.6 ± 0.9	86.3 ± 0.8
flare	82.2 ± 0.5	82.1 ± 0.3	82.1 ± 0.4	82.0 ± 0.4
glas	94.6 ± 1.0	94.6 ± 0.9	94.4 ± 0.8	94.4 ± 0.8
hear	83.1 ± 1.5	83.4 ± 1.6	83.2 ± 1.6	83.2 ± 1.6
img	96.9 ± 0.3	96.8 ± 0.3	97.1 ± 0.2	97.1 ± 0.2
ionos	90.6 ± 1.1	90.1 ± 1.0	90.1 ± 1.0	90.1 ± 1.0
mok1	96.6 ± 1.4	97.3 ± 1.4	98.6 ± 0.9	98.6 ± 0.9
mok2	91.0 ± 1.5	91.6 ± 1.4	91.0 ± 1.2	92.0 ± 1.1
pima	76.7 ± 1.0	76.9 ± 1.0	76.9 ± 1.0	76.8 ± 1.0
survi	74.9 ± 1.3	74.6 ± 1.4	74.8 ± 1.4	74.6 ± 1.4
vote	95.9 ± 0.6	95.6 ± 0.7	95.8 ± 0.6	95.8 ± 0.6
vowel	89.7 ± 0.7	91.5 ± 0.8	92.3 ± 0.7	92.3 ± 0.7
wdbc	97.3 ± 0.5	97.3 ± 0.5	97.3 ± 0.5	97.3 ± 0.5

Table C.33: Performance - *SE** of *MF* networks with *Static reordering*

Database	3-net	9-net	20-net	40-net
aritm	73.8 ± 1.2	76.1 ± 1.4	75.4 ± 1.4	74.9 ± 1.4
bala	95.1 ± 0.8	95.3 ± 0.8	95.4 ± 0.7	95.2 ± 0.7
band	73.6 ± 1.6	74.0 ± 0.9	74.7 ± 1.2	74.5 ± 1.0
bupa	71.4 ± 1.4	71.4 ± 1.5	71.9 ± 1.5	71.6 ± 1.2
cred	86.7 ± 0.7	87.2 ± 0.8	87.0 ± 0.6	87.3 ± 0.7
derma	97.5 ± 0.6	97.5 ± 0.6	97.3 ± 0.6	97.6 ± 0.6
ecoli	87.4 ± 0.9	87.1 ± 0.9	87.2 ± 0.8	87.2 ± 0.8
flare	81.8 ± 0.6	81.9 ± 0.6	81.9 ± 0.7	81.9 ± 0.6
glas	93.8 ± 0.8	94.0 ± 0.9	94.0 ± 0.9	94.0 ± 0.9
hear	83.2 ± 1.4	83.6 ± 1.2	83.6 ± 1.2	83.6 ± 1.2
img	96.7 ± 0.3	96.6 ± 0.3	96.7 ± 0.2	96.9 ± 0.2
ionos	90.0 ± 1.4	90.0 ± 1.1	90.7 ± 1.0	90.6 ± 1.0
mok1	97.8 ± 1.2	97.8 ± 1.2	97.8 ± 1.2	97.8 ± 1.2
mok2	91.8 ± 0.9	92.4 ± 0.6	92.0 ± 0.4	92.3 ± 0.6
pima	76.9 ± 0.8	77.2 ± 1.1	76.7 ± 1.1	76.6 ± 0.9
survi	74.8 ± 1.3	74.9 ± 1.4	74.6 ± 1.5	74.8 ± 1.4
vote	95.4 ± 0.8	95.6 ± 0.8	96.0 ± 0.6	95.9 ± 0.7
vowel	89.0 ± 0.6	89.9 ± 0.5	90.2 ± 0.5	90.6 ± 0.6
wdbc	97.2 ± 0.4	97.4 ± 0.4	97.4 ± 0.4	97.4 ± 0.3

Table C.34: Performance - *SE** of *MF* networks with *Dynamic reordering*

Database	3-net	9-net	20-net	40-net
aritm	74.1 ± 1.3	76.0 ± 1.5	76.3 ± 1.6	76.0 ± 1.6
bala	96.1 ± 0.8	96.2 ± 0.6	96.1 ± 0.7	95.9 ± 0.6
band	73.6 ± 1.1	74.0 ± 1.2	75.3 ± 1.1	74.4 ± 1.2
bupa	73.9 ± 0.9	72.7 ± 1.3	72.9 ± 1.3	72.1 ± 1.4
cred	87.4 ± 0.7	86.8 ± 0.6	86.8 ± 0.6	87.1 ± 0.6
derma	97.3 ± 0.7	97.5 ± 0.7	97.6 ± 0.7	97.6 ± 0.7
ecoli	86.5 ± 1.0	86.9 ± 1.0	86.9 ± 1.0	86.9 ± 1.0
flare	81.9 ± 0.4	82.0 ± 0.5	82.1 ± 0.6	81.9 ± 0.6
glas	95.2 ± 0.9	94.8 ± 0.9	94.8 ± 0.7	94.8 ± 0.7
hear	83.9 ± 1.4	84.2 ± 1.3	83.9 ± 1.3	83.9 ± 1.3
img	96.6 ± 0.3	96.7 ± 0.3	96.9 ± 0.3	96.9 ± 0.3
ionos	89.1 ± 1.4	89.6 ± 1.1	89.7 ± 1.1	89.4 ± 1.1
mok1	98.6 ± 1.0	99.5 ± 0.5	100.00 ± 0.00	98.6 ± 0.9
mok2	90.1 ± 1.0	91.8 ± 0.8	91.6 ± 1.0	91.9 ± 0.8
pima	76.8 ± 0.9	77.0 ± 1.0	76.6 ± 1.0	76.7 ± 1.0
survi	74.8 ± 1.5	74.8 ± 1.2	74.9 ± 1.4	74.9 ± 1.4
vote	95.6 ± 0.8	95.8 ± 0.6	95.8 ± 0.8	95.8 ± 0.8
vowel	89.0 ± 0.6	91.3 ± 0.6	91.4 ± 0.8	91.5 ± 0.8
wdbc	97.3 ± 0.4	97.3 ± 0.4	97.1 ± 0.4	97.1 ± 0.4

Table C.35: Performance - *CVCv1* of *MF* networks with *Static reordering*

Database	3-net	9-net	20-net	40-net
aritm	73.6 ± 1.4	72.9 ± 1.2	73.2 ± 1.6	74.4 ± 1.6
bala	96.1 ± 0.4	96.4 ± 0.7	96.8 ± 0.5	96.5 ± 0.6
band	74.0 ± 1.5	74.0 ± 1.1	74.9 ± 1.2	73.6 ± 1.3
bupa	72.1 ± 1.0	72.3 ± 1.4	73.6 ± 1.5	73.3 ± 1.1
cred	85.9 ± 0.8	85.5 ± 0.9	84.8 ± 0.7	85.2 ± 0.6
derma	97.6 ± 0.4	97.3 ± 0.5	97.2 ± 0.5	97.3 ± 0.5
ecoli	86.5 ± 1.0	86.9 ± 1.0	87.4 ± 1.0	86.8 ± 1.1
flare	81.7 ± 0.7	81.8 ± 0.7	81.4 ± 0.7	81.1 ± 0.6
glas	93.4 ± 0.8	95.6 ± 0.9	94.6 ± 0.9	95.4 ± 0.9
hear	82.7 ± 1.1	83.1 ± 1.4	82.2 ± 1.1	82.0 ± 1.2
img	96.9 ± 0.2	97.2 ± 0.2	97.2 ± 0.2	97.4 ± 0.3
ionos	89.6 ± 1.2	91.3 ± 1.2	91.3 ± 1.2	92.1 ± 1.1
mok1	99.4 ± 0.6	100.00 ± 0.00	98.8 ± 1.1	98.1 ± 1.3
mok2	87.9 ± 1.5	94.4 ± 1.4	95.5 ± 0.8	93.4 ± 1.6
pima	74.5 ± 1.1	74.3 ± 1.0	73.5 ± 0.9	73.7 ± 1.0
survi	74.1 ± 1.0	71.0 ± 0.8	72.3 ± 1.0	73.1 ± 1.1
vote	95.0 ± 0.6	95.0 ± 0.6	94.5 ± 0.5	94.9 ± 0.7
vowel	90.0 ± 0.6	92.8 ± 0.6	94.3 ± 0.6	94.2 ± 0.4
wdbc	96.7 ± 0.5	96.5 ± 0.4	96.4 ± 0.5	96.4 ± 0.6

Table C.36: Performance - *CVCv1* of *MF* networks with *Dynamic reordering*

Database	3-net	9-net	20-net	40-net
aritm	75.1 ± 1.2	73.0 ± 1.0	72.8 ± 1.2	73.0 ± 1.6
bala	95.4 ± 0.6	96.2 ± 0.8	96.8 ± 0.4	96.7 ± 0.6
band	73.5 ± 1.3	76.4 ± 1.1	74.2 ± 1.3	74.2 ± 0.8
bupa	72.4 ± 1.0	73.3 ± 1.6	74.0 ± 1.3	73.4 ± 1.5
cred	85.5 ± 0.8	85.7 ± 0.7	85.2 ± 0.6	84.4 ± 0.8
derma	97.9 ± 0.4	97.5 ± 0.6	97.0 ± 0.6	96.9 ± 0.5
ecoli	87.2 ± 1.0	87.5 ± 0.9	86.9 ± 1.0	87.1 ± 1.1
flare	80.8 ± 0.5	80.7 ± 0.6	80.5 ± 0.8	80.7 ± 0.7
glas	94.2 ± 0.9	95.2 ± 0.8	94.8 ± 1.0	95.4 ± 0.9
hear	84.2 ± 1.4	83.9 ± 1.1	83.2 ± 1.2	81.0 ± 1.4
img	97.1 ± 0.2	97.3 ± 0.2	97.3 ± 0.2	97.4 ± 0.2
ionos	90.9 ± 0.9	91.0 ± 1.0	91.7 ± 1.1	91.9 ± 1.2
mok1	98.1 ± 1.0	99.4 ± 0.6	98.3 ± 1.2	97.8 ± 1.2
mok2	89.0 ± 1.6	94.9 ± 1.2	95.4 ± 1.6	94.0 ± 1.6
pima	74.1 ± 1.2	74.1 ± 1.1	73.5 ± 0.6	73.5 ± 0.8
survi	73.3 ± 1.2	72.1 ± 0.8	72.1 ± 0.9	72.6 ± 0.7
vote	95.4 ± 0.5	95.1 ± 0.6	94.6 ± 0.7	94.6 ± 0.8
vowel	90.7 ± 0.8	93.4 ± 0.6	94.5 ± 0.6	94.9 ± 0.6
wdbc	97.0 ± 0.2	96.5 ± 0.4	96.1 ± 0.5	96.0 ± 0.5

Table C.37: Performance - *CVCv2* of *MF* networks with *Static reordering*

Database	3-net	9-net	20-net	40-net
aritm	74.8 ± 1.3	76.4 ± 1.6	75.4 ± 1.1	75.2 ± 1.3
bala	94.3 ± 0.7	96.5 ± 0.7	96.4 ± 0.6	96.4 ± 0.6
band	74.5 ± 1.6	73.8 ± 0.9	75.1 ± 1.2	74.9 ± 1.7
bupa	73.7 ± 1.1	73.3 ± 1.2	72.4 ± 1.3	72.7 ± 1.4
cred	86.7 ± 0.6	86.2 ± 0.7	86.4 ± 0.8	86.2 ± 0.8
derma	97.2 ± 0.6	97.5 ± 0.5	97.3 ± 0.5	97.3 ± 0.5
ecoli	87.2 ± 1.0	87.5 ± 0.7	86.9 ± 0.7	87.8 ± 1.0
flare	81.9 ± 0.6	82.1 ± 0.5	82.0 ± 0.6	81.9 ± 0.5
glas	94.0 ± 1.3	94.4 ± 1.0	94.6 ± 1.1	94.8 ± 1.0
hear	84.2 ± 1.3	83.9 ± 1.2	84.6 ± 1.3	82.2 ± 1.1
img	96.6 ± 0.2	97.0 ± 0.2	97.0 ± 0.2	97.0 ± 0.2
ionos	90.0 ± 1.0	91.0 ± 1.0	91.3 ± 1.0	91.9 ± 1.0
mok1	100.00 ± 0.00	100.00 ± 0.00	98.4 ± 1.1	100.00 ± 0.00
mok2	88.8 ± 1.2	94.6 ± 1.4	95.8 ± 0.8	95.4 ± 0.8
pima	77.0 ± 0.9	76.3 ± 1.0	76.3 ± 1.0	75.9 ± 0.9
survi	73.9 ± 1.3	73.6 ± 1.2	74.1 ± 1.4	74.1 ± 1.2
vote	96.6 ± 0.6	95.9 ± 0.5	95.6 ± 0.5	95.3 ± 0.5
vowel	89.6 ± 0.8	92.4 ± 0.8	92.9 ± 0.6	92.5 ± 0.6
wdbc	97.4 ± 0.4	96.9 ± 0.3	96.2 ± 0.6	96.0 ± 0.5

Table C.38: Performance - *CVCv2* of *MF* networks with *Dynamic reordering*

Database	3-net	9-net	20-net	40-net
aritm	76.1 ± 1.7	76.1 ± 1.2	75.7 ± 1.2	74.6 ± 1.4
bala	95.6 ± 0.6	96.3 ± 0.4	96.2 ± 0.6	96.2 ± 0.5
band	74.9 ± 1.2	75.5 ± 1.2	75.1 ± 1.0	74.2 ± 1.2
bupa	72.3 ± 1.4	74.3 ± 1.2	74.1 ± 1.1	73.9 ± 1.7
cred	86.9 ± 0.6	86.3 ± 0.8	86.1 ± 0.8	86.5 ± 0.7
derma	97.6 ± 0.5	97.6 ± 0.5	97.5 ± 0.5	97.5 ± 0.5
ecoli	86.2 ± 1.1	86.6 ± 1.2	86.9 ± 1.0	87.4 ± 0.9
flare	82.1 ± 0.6	81.9 ± 0.6	81.7 ± 0.5	81.9 ± 0.5
glas	95.2 ± 1.2	95.4 ± 1.0	95.4 ± 1.0	95.4 ± 0.8
hear	84.9 ± 1.1	84.7 ± 1.3	83.4 ± 1.2	82.0 ± 1.2
img	97.0 ± 0.3	97.35 ± 0.19	97.1 ± 0.2	97.1 ± 0.3
ionos	90.7 ± 0.9	91.7 ± 1.1	91.9 ± 1.0	91.4 ± 1.2
mok1	98.3 ± 0.9	98.9 ± 1.1	98.1 ± 1.3	100.00 ± 0.00
mok2	88.6 ± 1.1	95.8 ± 0.8	95.4 ± 1.2	93.8 ± 1.3
pima	76.7 ± 1.1	76.5 ± 1.2	76.3 ± 1.1	75.9 ± 0.9
survi	73.8 ± 1.5	74.8 ± 1.2	73.6 ± 0.8	73.8 ± 1.0
vote	96.4 ± 0.6	96.1 ± 0.4	95.0 ± 0.8	95.6 ± 0.4
vowel	89.5 ± 0.9	93.2 ± 0.6	93.3 ± 0.7	92.8 ± 0.6
wdbc	97.2 ± 0.4	97.0 ± 0.3	96.4 ± 0.5	95.9 ± 0.3

Table C.39: Performance - *CVCv2.5* of *MF* networks with *Static reordering*

Database	3-net	9-net	20-net	40-net
aritm	76.9 ± 0.9	76.4 ± 1.4	76.6 ± 1.6	76.1 ± 1.5
bala	95.9 ± 0.9	96.4 ± 0.6	96.2 ± 0.5	96.0 ± 0.5
band	73.6 ± 1.6	75.3 ± 1.3	75.1 ± 1.0	74.7 ± 1.3
bupa	73.6 ± 1.6	73.3 ± 1.3	72.9 ± 1.4	72.9 ± 1.3
cred	86.6 ± 0.7	86.5 ± 0.8	86.8 ± 0.7	87.0 ± 0.7
derma	97.5 ± 0.6	97.3 ± 0.5	97.3 ± 0.5	97.3 ± 0.5
ecoli	87.4 ± 0.9	86.8 ± 1.1	86.3 ± 0.9	86.6 ± 1.0
flare	81.4 ± 0.6	82.2 ± 0.5	82.2 ± 0.5	82.2 ± 0.5
glas	94.2 ± 0.8	94.8 ± 1.1	95.2 ± 1.0	95.0 ± 1.0
hear	83.9 ± 1.3	84.9 ± 1.2	85.1 ± 1.3	84.9 ± 1.4
img	96.77 ± 0.17	96.9 ± 0.2	97.0 ± 0.2	97.0 ± 0.2
ionos	90.4 ± 1.0	90.7 ± 1.0	91.0 ± 1.1	91.3 ± 0.9
mok1	98.9 ± 0.8	99.4 ± 0.6	99.6 ± 0.4	99.4 ± 0.6
mok2	91.8 ± 1.4	94.1 ± 1.4	94.8 ± 1.3	94.8 ± 1.4
pima	76.5 ± 1.0	76.5 ± 1.1	77.0 ± 0.9	76.9 ± 1.0
survi	74.3 ± 0.9	73.9 ± 1.0	73.4 ± 1.2	73.4 ± 1.3
vote	96.3 ± 0.6	96.1 ± 0.5	96.3 ± 0.6	96.3 ± 0.6
vowel	89.6 ± 0.7	92.1 ± 0.7	93.0 ± 0.7	93.6 ± 0.8
wdbc	96.8 ± 0.5	97.0 ± 0.4	97.1 ± 0.4	97.1 ± 0.4

Table C.40: Performance - *CVCv2.5* of *MF* networks with *Dynamic reordering*

Database	3-net	9-net	20-net	40-net
aritm	75.6 ± 1.3	76.7 ± 1.3	76.4 ± 1.3	76.6 ± 1.3
bala	96.3 ± 0.5	95.4 ± 0.6	95.6 ± 0.9	95.7 ± 0.8
band	75.8 ± 1.3	75.8 ± 0.6	74.4 ± 0.9	74.7 ± 1.1
bupa	72.4 ± 1.6	72.0 ± 1.4	73.1 ± 1.4	73.6 ± 1.2
cred	87.2 ± 0.6	86.9 ± 0.6	86.6 ± 0.6	86.6 ± 0.6
derma	97.3 ± 0.5	97.5 ± 0.4	97.7 ± 0.4	97.6 ± 0.4
ecoli	86.8 ± 1.0	86.9 ± 1.0	86.3 ± 1.0	86.3 ± 1.0
flare	82.0 ± 0.6	82.1 ± 0.5	82.0 ± 0.5	82.1 ± 0.5
glas	94.2 ± 1.0	94.8 ± 0.9	94.4 ± 1.0	94.6 ± 1.0
hear	84.4 ± 1.2	84.2 ± 1.3	84.6 ± 1.2	84.6 ± 1.2
img	96.6 ± 0.2	96.97 ± 0.20	97.1 ± 0.2	97.1 ± 0.2
ionos	91.0 ± 1.0	91.1 ± 1.0	91.1 ± 1.0	91.4 ± 1.0
mok1	98.4 ± 1.1	99.4 ± 0.6	99.4 ± 0.6	99.4 ± 0.6
mok2	92.6 ± 1.8	95.4 ± 0.9	94.6 ± 1.3	95.3 ± 1.2
pima	77.3 ± 1.0	76.8 ± 1.2	76.5 ± 1.2	76.5 ± 1.1
survi	73.9 ± 1.2	73.1 ± 1.4	73.6 ± 1.3	73.6 ± 1.3
vote	95.9 ± 0.5	95.9 ± 0.5	95.9 ± 0.5	95.9 ± 0.5
vowel	90.3 ± 0.4	92.5 ± 0.5	92.6 ± 0.5	92.7 ± 0.7
wdbc	97.1 ± 0.4	96.8 ± 0.4	97.0 ± 0.4	97.0 ± 0.4

Table C.41: Performance - *CVCv3* of *MF* networks with *Static reordering*

Database	3-net	9-net	20-net	40-net
aritm	75.2 ± 1.3	76.4 ± 1.5	76.6 ± 1.2	76.6 ± 1.4
bala	96.5 ± 0.8	96.6 ± 0.5	96.1 ± 0.7	96.2 ± 0.6
band	74.7 ± 1.6	75.1 ± 1.1	74.2 ± 1.3	74.7 ± 1.1
bupa	73.4 ± 1.6	74.0 ± 1.3	73.6 ± 1.4	73.7 ± 1.3
cred	86.7 ± 0.8	86.8 ± 0.8	86.9 ± 0.7	86.9 ± 0.6
derma	97.6 ± 0.7	97.0 ± 0.6	97.3 ± 0.5	97.5 ± 0.5
ecoli	86.9 ± 0.9	86.8 ± 0.8	87.2 ± 0.9	86.8 ± 0.8
flare	81.9 ± 0.4	82.1 ± 0.5	82.1 ± 0.5	82.4 ± 0.6
glas	94.0 ± 1.1	95.2 ± 0.8	94.4 ± 0.9	94.4 ± 0.8
hear	84.4 ± 1.5	83.9 ± 1.4	84.4 ± 1.5	84.6 ± 1.5
img	96.71 ± 0.15	96.8 ± 0.3	96.8 ± 0.3	97.0 ± 0.3
ionos	91.7 ± 0.9	91.1 ± 0.7	91.9 ± 0.8	91.4 ± 0.8
mok1	100.00 ± 0.00	99.4 ± 0.6	99.4 ± 0.6	99.4 ± 0.6
mok2	92.6 ± 1.4	94.5 ± 1.6	95.1 ± 1.3	95.0 ± 1.1
pima	76.5 ± 1.1	76.8 ± 1.1	77.0 ± 1.0	76.6 ± 1.1
survi	74.1 ± 1.2	73.9 ± 1.4	73.3 ± 1.5	73.1 ± 1.3
vote	96.3 ± 0.5	96.5 ± 0.5	96.4 ± 0.5	96.4 ± 0.5
vowel	90.0 ± 1.1	92.3 ± 0.9	93.2 ± 0.8	93.2 ± 0.8
wdbc	96.6 ± 0.4	96.7 ± 0.5	97.0 ± 0.4	96.9 ± 0.3

Table C.42: Performance - *CVCv3* of *MF* networks with *Dynamic reordering*

Database	3-net	9-net	20-net	40-net
aritm	74.5 ± 1.3	76.9 ± 1.3	77.1 ± 1.3	76.6 ± 1.1
bala	95.3 ± 0.7	95.8 ± 0.7	96.1 ± 0.6	95.9 ± 0.6
band	74.7 ± 1.0	74.5 ± 1.1	75.5 ± 0.9	75.1 ± 0.9
bupa	73.4 ± 1.7	72.9 ± 1.3	74.0 ± 1.1	72.9 ± 1.3
cred	86.9 ± 0.7	86.8 ± 0.7	86.6 ± 0.6	87.0 ± 0.6
derma	97.7 ± 0.4	97.3 ± 0.3	97.2 ± 0.4	97.2 ± 0.4
ecoli	86.0 ± 1.0	86.8 ± 1.0	86.5 ± 0.9	86.5 ± 0.9
flare	82.0 ± 0.6	82.0 ± 0.6	81.9 ± 0.5	82.2 ± 0.5
glas	94.4 ± 1.2	94.8 ± 0.9	94.8 ± 1.0	95.0 ± 1.0
hear	84.7 ± 1.4	83.9 ± 1.3	84.1 ± 1.3	84.2 ± 1.5
img	96.5 ± 0.3	97.0 ± 0.2	97.0 ± 0.2	97.05 ± 0.18
ionos	90.0 ± 1.2	90.3 ± 1.2	91.3 ± 0.9	90.7 ± 0.8
mok1	98.8 ± 0.8	99.1 ± 0.6	99.4 ± 0.6	99.4 ± 0.6
mok2	91.8 ± 1.6	93.8 ± 1.4	94.0 ± 1.4	93.5 ± 1.5
pima	76.8 ± 1.2	77.0 ± 1.1	77.1 ± 1.2	76.8 ± 1.1
survi	74.1 ± 1.3	74.6 ± 1.4	75.2 ± 1.2	74.9 ± 1.2
vote	96.3 ± 0.4	96.5 ± 0.5	96.5 ± 0.5	96.4 ± 0.5
vowel	91.0 ± 0.6	92.5 ± 0.9	92.5 ± 0.7	92.7 ± 0.6
wdbc	96.9 ± 0.3	97.0 ± 0.4	97.0 ± 0.4	96.9 ± 0.4

Table C.43: Performance - *DECOv1* of *MF* networks with *Static reordering*

Database	3-net	9-net	20-net	40-net
aritm	74.7 ± 1.4	76.3 ± 1.2	75.9 ± 1.3	75.5 ± 1.5
bala	95.8 ± 0.7	96.9 ± 0.5	96.8 ± 0.4	96.7 ± 0.4
band	73.8 ± 1.6	72.0 ± 1.6	74.2 ± 1.3	74.0 ± 1.8
bupa	71.9 ± 1.6	71.7 ± 1.5	73.0 ± 1.7	73.0 ± 1.5
cred	86.8 ± 0.8	87.2 ± 0.7	87.0 ± 0.6	87.1 ± 0.6
derma	97.5 ± 0.6	97.6 ± 0.7	97.6 ± 0.7	97.6 ± 0.7
ecoli	86.2 ± 0.9	86.5 ± 0.8	86.9 ± 0.9	87.4 ± 0.9
flare	82.6 ± 0.5	82.2 ± 0.5	82.3 ± 0.5	82.0 ± 0.6
glas	95.2 ± 0.8	95.0 ± 1.0	95.2 ± 1.0	95.0 ± 0.9
hear	83.2 ± 1.1	83.9 ± 1.4	83.9 ± 1.3	84.1 ± 1.4
img	96.6 ± 0.3	96.9 ± 0.3	96.9 ± 0.3	96.9 ± 0.3
ionos	90.1 ± 1.3	90.4 ± 1.2	90.9 ± 1.0	90.7 ± 1.0
mok1	97.4 ± 1.4	99.3 ± 0.8	99.3 ± 0.8	99.3 ± 0.8
mok2	88.6 ± 1.6	90.6 ± 0.7	91.8 ± 0.9	91.8 ± 0.9
pima	76.5 ± 1.0	76.7 ± 1.1	76.8 ± 1.0	76.6 ± 1.0
survi	74.3 ± 1.5	73.8 ± 1.4	74.1 ± 1.5	73.8 ± 1.6
vote	96.0 ± 0.7	95.8 ± 0.7	95.8 ± 0.7	95.6 ± 0.7
vowel	90.9 ± 0.9	92.1 ± 0.9	93.3 ± 0.7	93.4 ± 0.7
wdbc	97.1 ± 0.4	97.3 ± 0.4	97.3 ± 0.4	97.3 ± 0.4

Table C.44: Performance - *DECOv1* of *MF* networks with *Dynamic reordering*

Database	3-net	9-net	20-net	40-net
aritm	76.3 ± 1.6	76.2 ± 1.8	75.9 ± 1.7	76.3 ± 1.7
bala	96.3 ± 0.8	96.7 ± 0.5	96.6 ± 0.5	96.8 ± 0.4
band	74.2 ± 1.2	74.9 ± 1.2	74.4 ± 1.4	74.7 ± 1.5
bupa	71.6 ± 1.1	71.7 ± 1.2	72.7 ± 1.4	72.6 ± 1.5
cred	86.9 ± 0.6	86.8 ± 0.6	86.8 ± 0.6	87.1 ± 0.6
derma	97.6 ± 0.7	97.3 ± 0.6	97.3 ± 0.7	97.5 ± 0.7
ecoli	86.8 ± 1.0	86.9 ± 1.0	87.2 ± 0.9	87.2 ± 0.8
flare	81.7 ± 0.3	81.6 ± 0.4	81.8 ± 0.3	81.8 ± 0.3
glas	95.4 ± 0.7	95.4 ± 0.8	95.2 ± 0.8	95.0 ± 0.9
hear	83.7 ± 1.6	83.9 ± 1.6	84.1 ± 1.5	83.9 ± 1.4
img	96.8 ± 0.3	97.1 ± 0.3	97.0 ± 0.2	97.1 ± 0.2
ionos	89.1 ± 1.4	90.4 ± 1.1	90.4 ± 1.2	90.4 ± 1.1
mok1	97.6 ± 1.2	99.3 ± 0.8	98.6 ± 0.9	98.6 ± 0.9
mok2	90.1 ± 1.3	90.3 ± 1.8	90.1 ± 1.6	91.8 ± 1.1
pima	76.5 ± 0.9	76.8 ± 0.9	76.8 ± 0.9	76.7 ± 0.9
survi	74.3 ± 1.3	74.1 ± 1.4	74.4 ± 1.5	73.9 ± 1.4
vote	95.8 ± 0.7	95.4 ± 0.7	95.5 ± 0.6	95.5 ± 0.6
vowel	91.0 ± 0.7	92.8 ± 0.6	93.7 ± 0.5	93.7 ± 0.5
wdbc	97.2 ± 0.5	97.1 ± 0.5	97.3 ± 0.5	97.3 ± 0.5

Table C.45: Performance - *DECOv2* of *MF* networks with *Static reordering*

Database	3-net	9-net	20-net	40-net
aritm	74.6 ± 1.7	75.5 ± 1.5	75.4 ± 1.7	75.2 ± 1.5
bala	95.3 ± 0.8	95.4 ± 0.7	95.6 ± 0.6	95.6 ± 0.6
band	74.4 ± 0.9	75.3 ± 1.4	74.5 ± 1.4	74.5 ± 1.4
bupa	71.7 ± 1.3	72.0 ± 1.2	72.0 ± 1.2	72.3 ± 1.3
cred	87.2 ± 0.7	87.1 ± 0.6	87.2 ± 0.6	87.0 ± 0.6
derma	97.3 ± 0.7	97.3 ± 0.7	97.3 ± 0.7	97.3 ± 0.7
ecoli	86.3 ± 0.8	86.6 ± 0.8	87.2 ± 1.0	87.2 ± 1.1
flare	82.5 ± 0.7	81.7 ± 0.5	82.0 ± 0.5	82.0 ± 0.5
glas	94.2 ± 1.0	94.8 ± 1.0	94.8 ± 1.1	94.2 ± 1.0
hear	83.6 ± 1.4	83.4 ± 1.2	83.9 ± 1.3	83.7 ± 1.3
img	96.7 ± 0.3	96.8 ± 0.3	96.8 ± 0.3	96.9 ± 0.3
ionos	90.3 ± 1.1	90.6 ± 0.9	90.3 ± 1.0	90.1 ± 1.0
mok1	97.6 ± 1.3	98.9 ± 0.8	98.9 ± 0.8	97.6 ± 1.3
mok2	89.8 ± 1.3	90.3 ± 1.4	91.4 ± 1.1	91.6 ± 0.8
pima	76.3 ± 0.7	76.8 ± 0.8	77.1 ± 0.9	76.5 ± 0.9
survi	74.8 ± 1.0	74.3 ± 1.2	74.9 ± 1.3	74.9 ± 1.3
vote	96.0 ± 0.6	95.9 ± 0.6	95.9 ± 0.6	96.0 ± 0.7
vowel	90.5 ± 0.8	92.4 ± 0.9	93.2 ± 0.8	93.1 ± 0.7
wdbc	97.2 ± 0.4	97.2 ± 0.4	97.3 ± 0.4	97.1 ± 0.4

Table C.46: Performance - *DECOv2* of *MF* networks with *Dynamic reordering*

Database	3-net	9-net	20-net	40-net
aritm	75.5 ± 1.4	75.7 ± 1.5	75.7 ± 1.6	76.0 ± 1.6
bala	95.4 ± 0.8	95.5 ± 0.9	95.5 ± 0.9	95.5 ± 0.9
band	74.2 ± 1.2	74.4 ± 1.2	74.4 ± 1.5	74.5 ± 1.2
bupa	73.3 ± 1.0	73.6 ± 1.1	73.3 ± 1.0	73.3 ± 1.0
cred	86.5 ± 0.6	86.8 ± 0.6	86.8 ± 0.6	86.9 ± 0.6
derma	96.8 ± 0.8	96.9 ± 0.8	96.9 ± 0.8	96.8 ± 0.8
ecoli	86.6 ± 0.9	86.5 ± 0.8	86.5 ± 0.7	86.6 ± 0.8
flare	81.6 ± 0.5	81.7 ± 0.5	81.5 ± 0.4	81.9 ± 0.4
glas	93.2 ± 1.0	94.2 ± 0.8	94.2 ± 0.8	94.2 ± 0.8
hear	83.1 ± 1.7	83.6 ± 1.5	83.6 ± 1.6	83.7 ± 1.5
img	96.6 ± 0.3	96.8 ± 0.3	96.9 ± 0.3	97.0 ± 0.3
ionos	90.7 ± 0.9	90.3 ± 1.1	90.4 ± 1.1	90.4 ± 1.1
mok1	97.6 ± 1.3	97.6 ± 1.3	98.3 ± 1.3	98.3 ± 1.3
mok2	88.9 ± 1.6	92.1 ± 1.0	91.9 ± 1.3	92.1 ± 1.0
pima	76.8 ± 1.0	76.7 ± 0.9	76.9 ± 1.0	76.7 ± 1.0
survi	75.1 ± 1.2	75.2 ± 1.3	75.2 ± 1.4	75.2 ± 1.4
vote	96.0 ± 0.6	95.5 ± 0.7	95.5 ± 0.7	95.5 ± 0.7
vowel	90.8 ± 0.8	91.7 ± 0.7	91.8 ± 0.7	92.1 ± 0.7
wdbc	97.2 ± 0.5	97.1 ± 0.5	97.1 ± 0.5	97.1 ± 0.5

Table C.47: Performance - *Adaboost* of *MF* networks with *Dynamic reordering*

Database	3-net	9-net	20-net	40-net
aritm	73.2 ± 1.1	74.3 ± 1.3	73.3 ± 1.1	73.9 ± 1.5
bala	93.8 ± 0.6	96.1 ± 0.6	96.2 ± 0.6	96.2 ± 0.4
band	68.2 ± 1.7	69.8 ± 1.5	72.2 ± 1.5	72.9 ± 1.6
bupa	70.7 ± 1.8	73 ± 2	72.1 ± 1.5	72.7 ± 1.6
cred	85.5 ± 1.0	84.9 ± 0.7	84.8 ± 0.7	84.7 ± 0.9
derma	96.8 ± 1.0	97.5 ± 0.6	97.3 ± 0.5	97.3 ± 0.7
ecoli	84.6 ± 1.2	85.3 ± 1.0	84.6 ± 1.0	84.4 ± 1.0
flare	81.7 ± 0.8	80.7 ± 0.8	80.4 ± 0.8	80.1 ± 0.7
glas	96.2 ± 1.0	96.6 ± 0.8	97.0 ± 0.7	96.2 ± 0.8
hear	81.9 ± 1.2	82.7 ± 1.4	81.7 ± 1.5	81.0 ± 1.2
img	96.5 ± 0.4	97.2 ± 0.3	97.3 ± 0.3	97.3 ± 0.3
ionos	88.4 ± 1.0	90.1 ± 1.2	90.9 ± 0.7	91.4 ± 1.2
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	79.0 ± 1.2	81.1 ± 1.6	83.3 ± 1.5	83.4 ± 1.1
pima	77.0 ± 1.0	76.8 ± 0.9	75.4 ± 1.0	75.4 ± 1.0
survi	75.6 ± 1.6	74.1 ± 1.6	72.8 ± 1.5	71.3 ± 1.5
vote	95.3 ± 0.6	95.6 ± 0.6	95.6 ± 0.6	96.0 ± 0.7
vowel	89.4 ± 0.7	94.6 ± 0.7	96.4 ± 0.4	97.2 ± 0.4
wdbc	96.7 ± 0.3	96.8 ± 0.5	96.6 ± 0.5	96.5 ± 0.5

Table C.48: Performance - *Aveboost* of *MF* networks with *Dynamic reordering*

Database	3-net	9-net	20-net	40-net
aritm	76.0 ± 1.1	76.8 ± 0.8	76.6 ± 1.1	76.1 ± 1.0
bala	96.1 ± 0.6	96.3 ± 0.5	96.7 ± 0.5	96.7 ± 0.4
band	72 ± 2	74.5 ± 1.1	73.3 ± 1.6	74.5 ± 1.3
bupa	72.3 ± 1.5	73.1 ± 1.1	73.1 ± 1.6	73.4 ± 1.4
cred	86.5 ± 0.8	86.9 ± 1.0	87.1 ± 0.8	85.5 ± 0.9
derma	96.2 ± 0.7	97.2 ± 0.7	96.9 ± 0.7	97.0 ± 0.6
ecoli	86.2 ± 1.1	87.4 ± 0.9	86.8 ± 1.3	85.4 ± 1.3
flare	81.8 ± 0.6	82.4 ± 0.6	82.9 ± 0.6	82.7 ± 0.7
glas	92.8 ± 1.1	96.6 ± 0.9	97.4 ± 0.6	97.0 ± 0.7
hear	83.7 ± 1.3	82.9 ± 1.3	84.1 ± 1.5	81.7 ± 1.6
img	96.8 ± 0.3	97.4 ± 0.2	97.4 ± 0.3	97.42 ± 0.16
ionos	90.3 ± 0.7	90.1 ± 0.9	90.6 ± 0.8	91.0 ± 1.0
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	81.4 ± 1.7	85.8 ± 1.6	87.6 ± 1.2	90.0 ± 1.1
pima	76.7 ± 1.0	76.5 ± 0.9	76.4 ± 0.9	76.1 ± 1.0
survi	73.8 ± 1.0	73.8 ± 1.0	73.1 ± 1.0	73.3 ± 1.1
vote	96.3 ± 0.7	95.8 ± 0.7	96.1 ± 0.7	95.6 ± 0.7
vowel	87.0 ± 1.0	93.8 ± 0.6	96.2 ± 0.6	96.7 ± 0.6
wdbc	96.2 ± 0.5	96.1 ± 0.6	96.6 ± 0.5	96.6 ± 0.4

Table C.49: Performance - *Aggreboost* of *MF* networks with *Dynamic reordering*

Database	3-net	9-net	20-net	40-net
aritm	72.3 ± 1.9	74.0 ± 1.4	74.8 ± 1.4	75.5 ± 1.2
bala	94.4 ± 0.7	95.4 ± 0.5	95.6 ± 0.5	96.1 ± 0.6
band	69 ± 2	72.5 ± 1.7	73.1 ± 2.0	73.5 ± 1.3
bupa	71 ± 2	71.7 ± 1.6	71.4 ± 1.7	72.7 ± 1.7
cred	86.5 ± 0.7	85.2 ± 1.1	85.7 ± 0.7	85.5 ± 0.8
derma	97.0 ± 0.5	96.6 ± 0.6	97.0 ± 0.6	96.6 ± 0.5
ecoli	85.7 ± 1.4	87.4 ± 1.1	87.1 ± 1.0	87.8 ± 0.9
flare	81.9 ± 0.9	81.8 ± 0.7	82.2 ± 0.5	82.0 ± 0.6
glas	93.4 ± 0.8	96.2 ± 0.8	96.2 ± 0.8	96.4 ± 0.8
hear	82.7 ± 1.4	81.9 ± 1.5	84.6 ± 1.2	83.9 ± 1.2
img	96.6 ± 0.3	96.9 ± 0.2	97.2 ± 0.3	97.3 ± 0.3
ionos	90.3 ± 0.8	91.7 ± 1.2	91.6 ± 0.9	92.3 ± 0.9
mok1	99.6 ± 0.4	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	74.6 ± 1.1	80.9 ± 1.9	85.3 ± 1.8	86.3 ± 1.1
pima	74.3 ± 1.5	74.9 ± 1.7	75.5 ± 1.3	75.7 ± 1.2
survi	74.1 ± 1.5	73.1 ± 0.9	73.8 ± 1.7	73.9 ± 1.4
vote	94.5 ± 0.9	94.6 ± 0.7	94.9 ± 0.7	95.5 ± 0.8
vowel	86.9 ± 1.2	94.6 ± 0.6	96.9 ± 0.6	97.5 ± 0.5
wdbc	96.6 ± 0.6	96.5 ± 0.6	96.8 ± 0.6	96.7 ± 0.6

Table C.50: Performance - *Conserboost* of *MF* networks with *Dynamic reordering*

Database	3-net	9-net	20-net	40-net
aritm	74.6 ± 1.2	74.9 ± 1.1	75.7 ± 1.1	75.3 ± 1.5
bala	95.7 ± 0.5	96.3 ± 0.4	96.4 ± 0.4	96.5 ± 0.4
band	71.5 ± 1.4	73.1 ± 1.1	75.5 ± 1.2	75.1 ± 1.4
bupa	71.9 ± 1.7	72.9 ± 1.3	72.7 ± 1.2	73.3 ± 1.2
cred	86.6 ± 0.9	86.4 ± 0.8	85.3 ± 1.0	85.8 ± 0.8
derma	97.6 ± 0.7	97.3 ± 0.7	98.0 ± 0.6	97.7 ± 0.5
ecoli	83.5 ± 0.9	85.4 ± 1.0	87.4 ± 1.2	87.5 ± 1.0
flare	82.3 ± 0.4	82.2 ± 0.6	82.3 ± 0.5	82.1 ± 0.5
glas	95.4 ± 0.8	95.6 ± 0.9	96.8 ± 0.7	96.8 ± 0.6
hear	83.6 ± 1.5	83.2 ± 1.3	83.2 ± 0.9	83.4 ± 1.2
img	96.2 ± 0.3	97.2 ± 0.3	97.25 ± 0.18	97.3 ± 0.2
ionos	88.9 ± 1.0	90.4 ± 1.1	91.7 ± 0.8	91.7 ± 0.7
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	77 ± 2	84.1 ± 1.3	86.5 ± 1.5	88.5 ± 1.5
pima	76.8 ± 1.1	77.2 ± 1.2	76.9 ± 1.0	77.4 ± 0.9
survi	74.3 ± 1.3	74.3 ± 1.3	73.9 ± 1.3	74.1 ± 1.3
vote	96.4 ± 0.7	95.8 ± 0.6	96.1 ± 0.6	96.0 ± 0.7
vowel	88.5 ± 1.2	94.6 ± 0.6	96.3 ± 0.6	97.2 ± 0.5
wdbc	96.6 ± 0.4	96.6 ± 0.5	96.3 ± 0.6	96.7 ± 0.6

Table C.51: Performance - *ATA-LE* of *MF* networks with *Static reordering*

Database	3-net	9-net	20-net	40-net
aritm	72.9 ± 0.9	73.3 ± 1.8	72.5 ± 1.7	73.4 ± 1.6
bala	94.7 ± 0.9	95.7 ± 0.5	95.5 ± 0.6	94.3 ± 1.1
band	73.3 ± 1.5	72.5 ± 1.3	73.3 ± 1.1	73.6 ± 1.5
bupa	72.7 ± 1.4	71.9 ± 1.5	72.3 ± 1.4	72.7 ± 1.5
cred	86.8 ± 0.8	86.0 ± 0.9	86.8 ± 0.7	85.0 ± 0.7
derma	95.4 ± 0.8	96.9 ± 0.7	96.6 ± 0.6	96.9 ± 0.7
ecoli	86.8 ± 1.0	85.6 ± 1.8	85.4 ± 1.1	86.0 ± 1.2
flare	81.5 ± 0.7	82.0 ± 0.7	81.2 ± 0.8	81.9 ± 0.7
glas	89.2 ± 1.2	90.2 ± 1.7	91.2 ± 1.0	90.4 ± 1.5
hear	83.4 ± 1.6	83.7 ± 1.6	83.7 ± 1.5	83.4 ± 1.7
img	96.7 ± 0.3	96.9 ± 0.2	96.9 ± 0.3	97.0 ± 0.3
ionos	90.9 ± 1.1	91.1 ± 0.8	90.6 ± 1.2	90.0 ± 1.1
mok1	99.6 ± 0.4	100.00 ± 0.00	99.6 ± 0.3	100.00 ± 0.00
mok2	85.5 ± 1.5	89.4 ± 1.5	90.4 ± 0.9	89.0 ± 1.0
pima	74.6 ± 1.1	76.1 ± 1.0	75.9 ± 1.3	76.3 ± 0.9
survi	74.4 ± 1.5	76.2 ± 1.5	75.4 ± 1.9	71.1 ± 1.2
vote	96.3 ± 0.7	95.4 ± 0.7	95.3 ± 0.7	95.4 ± 0.7
vowel	87.3 ± 0.8	90.2 ± 0.6	89.8 ± 0.9	89.1 ± 0.8
wdbc	97.2 ± 0.3	97.2 ± 0.4	97.1 ± 0.4	97.1 ± 0.4

Table C.52: Performance - *ATA-LE* of *MF* networks with *Dynamic reordering*

Database	3-net	9-net	20-net	40-net
aritm	73.6 ± 1.4	73.1 ± 1.8	73.6 ± 1.8	74.4 ± 1.9
bala	94.3 ± 0.9	95.9 ± 0.8	95.6 ± 0.8	95.0 ± 1.1
band	72.4 ± 1.7	72.0 ± 1.5	72.5 ± 1.6	72.5 ± 1.9
bupa	71 ± 2	71.4 ± 1.4	71.4 ± 1.3	71.6 ± 1.6
cred	86.8 ± 0.7	85.9 ± 0.6	85.4 ± 0.7	85.3 ± 0.6
derma	96.3 ± 0.8	96.8 ± 0.8	96.2 ± 0.9	96.9 ± 0.7
ecoli	85.6 ± 1.2	85.7 ± 1.4	85.0 ± 1.4	85.0 ± 1.8
flare	82.0 ± 0.8	81.8 ± 0.8	80.9 ± 0.9	82.1 ± 0.6
glas	88 ± 2	90.8 ± 1.2	90.0 ± 1.2	88.8 ± 2.0
hear	84.6 ± 1.5	84.2 ± 1.5	83.9 ± 1.4	83.9 ± 1.4
img	96.84 ± 0.18	97.0 ± 0.2	96.8 ± 0.3	96.8 ± 0.3
ionos	89.9 ± 0.9	91.3 ± 0.9	92.1 ± 0.7	89.1 ± 0.9
mok1	99.88 ± 0.13	100.00 ± 0.00	99.4 ± 0.5	100.00 ± 0.00
mok2	83.9 ± 1.8	90.0 ± 1.0	89.6 ± 1.0	89.5 ± 0.8
pima	75.7 ± 0.9	76.0 ± 1.1	76.2 ± 0.9	76.8 ± 1.0
survi	75.1 ± 1.6	76.2 ± 1.4	75.7 ± 1.4	72.5 ± 1.9
vote	95.9 ± 0.7	95.3 ± 0.7	95.1 ± 0.8	95.3 ± 0.7
vowel	86.9 ± 0.6	89.2 ± 0.9	88.9 ± 0.7	89.5 ± 1.1
wdbc	96.5 ± 0.4	97.2 ± 0.3	97.5 ± 0.4	97.2 ± 0.4

Table C.53: Performance - *ATA-BE* of *MF* networks with *Static reordering*

Database	3-net	9-net	20-net	40-net
aritm	74.5 ± 1.8	75.4 ± 1.5	74.4 ± 1.6	74.1 ± 1.6
bala	94.6 ± 0.9	95.6 ± 0.8	95.5 ± 0.7	95.2 ± 0.7
band	71.6 ± 2.0	74.9 ± 1.2	72.0 ± 1.2	73.3 ± 1.4
bupa	72.7 ± 1.3	72.6 ± 1.2	72.0 ± 1.4	71.7 ± 1.2
cred	86.9 ± 0.8	86.5 ± 0.7	87.2 ± 0.7	87.3 ± 0.7
derma	96.2 ± 0.8	96.3 ± 0.9	96.1 ± 0.8	96.6 ± 0.9
ecoli	85.1 ± 1.2	87.2 ± 1.0	86.2 ± 1.2	85.1 ± 1.1
flare	81.7 ± 0.8	81.9 ± 0.5	82.0 ± 0.7	82.2 ± 0.7
glas	86 ± 2	89.8 ± 1.9	89.6 ± 1.2	88.0 ± 1.9
hear	84.4 ± 1.2	83.6 ± 1.3	83.7 ± 1.2	84.4 ± 1.4
img	96.6 ± 0.3	97.1 ± 0.2	97.1 ± 0.3	97.0 ± 0.2
ionos	89.6 ± 1.2	90.9 ± 1.4	90.9 ± 1.2	89.9 ± 1.1
mok1	98.6 ± 0.9	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	85.1 ± 1.4	88.4 ± 1.3	89.6 ± 0.9	90.6 ± 1.5
pima	76.6 ± 0.9	75.5 ± 1.2	77.0 ± 0.9	76.8 ± 0.9
survi	73.9 ± 1.3	75.1 ± 1.5	74.9 ± 1.3	74.8 ± 1.5
vote	96.1 ± 0.8	95.8 ± 0.7	95.6 ± 0.7	95.5 ± 0.7
vowel	88.4 ± 0.5	89.7 ± 0.5	89.1 ± 0.9	90.7 ± 1.1
wdbc	96.9 ± 0.4	97.2 ± 0.3	96.9 ± 0.4	97.1 ± 0.5

Table C.54: Performance - *ATA-BE* of *MF* networks with *Dynamic reordering*

Database	3-net	9-net	20-net	40-net
aritm	75.6 ± 1.2	75.3 ± 1.4	74.9 ± 1.2	74.8 ± 1.4
bala	95.8 ± 0.7	95.8 ± 0.8	95.4 ± 0.9	94.8 ± 0.8
band	74.5 ± 1.6	73.5 ± 1.8	76.0 ± 0.8	74.4 ± 1.2
bupa	72.6 ± 1.3	72.9 ± 1.1	73.7 ± 1.0	72.6 ± 1.2
cred	86.4 ± 0.8	86.8 ± 0.6	86.7 ± 0.5	86.5 ± 0.5
derma	96.2 ± 1.0	96.6 ± 0.8	97.2 ± 0.7	97.0 ± 0.5
ecoli	86.2 ± 0.6	86.9 ± 1.0	86.2 ± 1.2	86.2 ± 1.1
flare	81.9 ± 0.5	81.9 ± 0.5	81.7 ± 0.6	81.9 ± 0.7
glas	89.4 ± 0.9	90.4 ± 1.4	88 ± 2	90.4 ± 1.2
hear	83.9 ± 1.4	84.1 ± 1.5	84.4 ± 1.5	84.4 ± 1.3
img	96.7 ± 0.2	97.1 ± 0.3	97.1 ± 0.2	96.9 ± 0.3
ionos	88.9 ± 1.1	92.1 ± 1.0	91.1 ± 1.1	91.0 ± 1.4
mok1	98.8 ± 1.3	100.00 ± 0.00	99.88 ± 0.13	99.88 ± 0.13
mok2	81.5 ± 1.7	87.1 ± 1.5	88.8 ± 1.3	90.6 ± 0.7
pima	76.8 ± 1.1	75.9 ± 1.0	77.0 ± 1.1	77.0 ± 1.1
survi	74.4 ± 1.5	74.6 ± 1.3	75.2 ± 1.0	73.4 ± 0.9
vote	96.3 ± 0.7	96.3 ± 0.5	96.3 ± 0.6	96.0 ± 0.7
vowel	86.4 ± 0.7	89.0 ± 1.0	88.9 ± 0.9	89.3 ± 0.6
wdbc	97.1 ± 0.4	97.4 ± 0.4	97.3 ± 0.4	97.5 ± 0.3

Table C.55: Performance - *EENCL-LG* of *MF* networks with *Dynamic reordering*

Database	3-net	9-net	20-net	40-net
aritm	72.4 ± 1.7	73.0 ± 1.7	71.1 ± 1.3	72.0 ± 1.6
bala	96.2 ± 0.7	95.5 ± 0.5	94.3 ± 0.7	95.9 ± 0.5
band	74.4 ± 0.8	73.3 ± 1.3	72.7 ± 0.7	71.5 ± 1.4
bupa	69.9 ± 1.6	72.0 ± 1.4	70.9 ± 1.7	72.9 ± 1.6
cred	86.1 ± 0.8	86.2 ± 0.6	86.8 ± 1.0	86.6 ± 0.6
derma	97.0 ± 0.7	94.9 ± 0.8	96.1 ± 0.8	95.9 ± 0.8
ecoli	85.6 ± 1.0	87.5 ± 1.0	87.8 ± 1.0	86.9 ± 1.1
flare	80.0 ± 0.9	81.2 ± 1.0	81.7 ± 0.8	81.9 ± 0.9
glas	93.0 ± 1.0	91.0 ± 1.1	92.8 ± 1.0	93.4 ± 0.8
hear	80.3 ± 1.7	82.5 ± 1.5	80.3 ± 1.6	83.1 ± 1.2
img	97.0 ± 0.2	96.6 ± 0.2	97.0 ± 0.3	97.0 ± 0.2
ionos	92.3 ± 1.1	92.7 ± 0.8	92.0 ± 1.0	93.4 ± 1.2
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	86 ± 2	84.6 ± 1.6	83 ± 2	83 ± 2
pima	73.8 ± 0.8	74.1 ± 0.8	75.2 ± 1.0	74.5 ± 1.0
survi	73.9 ± 1.6	74.1 ± 1.5	73.8 ± 1.4	72.8 ± 1.0
vote	95.3 ± 0.4	94.9 ± 0.8	96.0 ± 0.6	95.0 ± 0.7
vowel	87.2 ± 0.6	88.7 ± 0.9	90.8 ± 0.9	89.9 ± 0.7
wdbc	95.8 ± 0.6	96.7 ± 0.7	96.5 ± 0.5	96.3 ± 0.7

Table C.56: Performance - *EENCL-BG* of *MF* networks with *Dynamic reordering*

Database	3-net	9-net	20-net	40-net
aritm	75.6 ± 1.7	74.9 ± 1.4	75.3 ± 1.4	74.4 ± 1.2
bala	95.8 ± 0.7	95.4 ± 0.6	94.3 ± 0.7	95.6 ± 0.5
band	74.0 ± 1.4	72.7 ± 1.3	74.4 ± 1.1	72.0 ± 1.5
bupa	72.1 ± 1.4	72.9 ± 1.2	72.4 ± 1.4	72.1 ± 1.5
cred	86.8 ± 0.7	86.5 ± 0.6	86.5 ± 0.7	86.8 ± 0.5
derma	97.2 ± 0.7	94.5 ± 0.8	96.1 ± 0.8	96.5 ± 0.9
ecoli	86.9 ± 1.1	87.4 ± 0.8	87.5 ± 0.8	87.4 ± 1.2
flare	81.7 ± 0.5	81.8 ± 0.6	81.7 ± 0.7	82.2 ± 0.7
glas	93.2 ± 1.3	91.6 ± 1.2	92.6 ± 1.4	91.8 ± 0.9
hear	83.6 ± 1.2	84.1 ± 1.4	82.4 ± 1.6	81.5 ± 1.1
img	96.5 ± 0.3	96.62 ± 0.18	97.05 ± 0.20	97.0 ± 0.2
ionos	93.4 ± 1.0	92.9 ± 1.1	92.9 ± 0.9	92.6 ± 0.8
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	85 ± 2	86.1 ± 1.5	83 ± 2	83 ± 2
pima	76.6 ± 1.0	76.5 ± 0.8	76.8 ± 1.3	76.4 ± 1.1
survi	74.1 ± 1.7	74.6 ± 1.0	74.8 ± 0.9	73.9 ± 1.0
vote	96.0 ± 0.5	96.0 ± 0.6	96.3 ± 0.5	95.8 ± 0.7
vowel	87.0 ± 1.0	88.5 ± 1.0	90.9 ± 0.8	89.6 ± 0.7
wdbc	96.5 ± 0.5	97.2 ± 0.4	96.5 ± 0.7	96.5 ± 0.8

Table C.57: Performance - *ACB* of *MF* networks and *Output Average*

Database	3-net	9-net	20-net	40-net
aritm	73.3 ± 0.5	76.2 ± 1.3	76.6 ± 1.2	77.7 ± 1.4
bala	96.0 ± 0.6	96.2 ± 0.5	96.2 ± 0.4	96.5 ± 0.6
band	71.3 ± 1.3	72.4 ± 1.5	74.9 ± 1.0	74.0 ± 1.1
bupa	73.4 ± 1.7	74.1 ± 1.4	72.6 ± 1.5	72.4 ± 1.1
cred	86.8 ± 0.9	87.2 ± 0.6	86.6 ± 0.8	86.4 ± 0.9
derma	97.9 ± 0.5	98.0 ± 0.5	97.7 ± 0.6	97.7 ± 0.6
ecoli	86.0 ± 1.0	87.5 ± 0.5	88.4 ± 0.7	86.9 ± 1.2
flare	82.0 ± 0.8	82.4 ± 0.7	82.4 ± 0.7	82.4 ± 0.7
glas	94.0 ± 1.2	95.4 ± 0.9	96.0 ± 1.0	96.6 ± 0.8
hear	82.7 ± 1.4	84.6 ± 1.3	84.2 ± 1.4	83.9 ± 1.3
img	96.3 ± 0.2	97.1 ± 0.3	97.3 ± 0.3	97.4 ± 0.3
ionos	88.6 ± 1.0	90.4 ± 1.1	90.7 ± 1.1	90.9 ± 0.9
mok1	99.1 ± 0.6	99.8 ± 0.3	100.00 ± 0.00	100.00 ± 0.00
mok2	80 ± 2	82.0 ± 1.7	85.3 ± 1.5	87.1 ± 1.5
pima	77.4 ± 0.8	75.9 ± 1.1	75.7 ± 1.0	76.1 ± 1.0
survi	75.6 ± 1.2	74.6 ± 1.3	74.8 ± 1.4	74.8 ± 1.5
vote	96.3 ± 0.7	96.0 ± 0.6	95.6 ± 0.7	95.6 ± 0.8
vowel	87.9 ± 0.7	92.8 ± 0.6	94.6 ± 0.4	96.1 ± 0.6
wdbc	96.9 ± 0.4	96.9 ± 0.6	96.7 ± 0.5	96.5 ± 0.4

Table C.58: Performance - *ACB* of *MF* networks and *Boosting Combiner*

Database	3-net	9-net	20-net	40-net
aritm	74.0 ± 1.1	76.1 ± 1.1	76.3 ± 1.4	77.4 ± 1.3
bala	95.9 ± 0.6	96.6 ± 0.4	96.0 ± 0.5	96.1 ± 0.5
band	69.8 ± 1.4	70.9 ± 1.6	73.5 ± 1.2	73.3 ± 1.8
bupa	73.7 ± 1.7	73.0 ± 1.5	72.4 ± 1.9	73.0 ± 1.4
cred	87.1 ± 0.8	87.0 ± 0.6	86.7 ± 0.7	86.5 ± 0.7
derma	97.3 ± 0.8	98.2 ± 0.5	97.9 ± 0.5	97.7 ± 0.6
ecoli	86.3 ± 0.9	87.4 ± 0.7	87.4 ± 0.5	87.8 ± 0.7
flare	82.1 ± 0.7	82.1 ± 0.6	82.7 ± 0.6	82.9 ± 0.6
glas	93.6 ± 1.2	95.2 ± 0.9	95.6 ± 0.9	96.4 ± 0.9
hear	82.9 ± 1.5	84.1 ± 1.3	84.1 ± 1.4	84.1 ± 1.4
img	96.3 ± 0.3	97.1 ± 0.3	97.2 ± 0.3	97.3 ± 0.3
ionos	88.4 ± 0.8	89.6 ± 1.1	90.4 ± 1.1	91.1 ± 0.8
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	80 ± 2	81.8 ± 1.5	86.1 ± 1.4	87.0 ± 1.4
pima	76.5 ± 1.1	76.2 ± 1.1	75.9 ± 0.9	76.5 ± 0.9
survi	74.6 ± 1.5	73.9 ± 1.5	74.9 ± 1.4	75.1 ± 1.5
vote	95.8 ± 0.6	95.6 ± 0.6	95.6 ± 0.7	95.6 ± 0.7
vowel	86.3 ± 0.8	91.3 ± 0.7	94.4 ± 0.6	95.8 ± 0.5
wdbc	96.2 ± 0.6	96.5 ± 0.5	96.5 ± 0.6	96.4 ± 0.4

Table C.59: Performance - *WCB* of *MF* networks and *Output Average*

Database	3-net	9-net	20-net	40-net
aritm	75.2 ± 1.3	74.8 ± 1.3	75.9 ± 1.5	75.3 ± 0.9
bala	96.3 ± 0.4	96.1 ± 0.3	96.5 ± 0.4	96.3 ± 0.5
band	73.3 ± 1.3	73.1 ± 1.9	75.3 ± 1.5	74.4 ± 1.4
bupa	71.1 ± 1.8	70.1 ± 1.5	73.4 ± 1.4	72.1 ± 1.4
cred	86.7 ± 0.6	86.0 ± 0.6	86.7 ± 1.0	87.2 ± 0.7
derma	97.6 ± 0.6	98.0 ± 0.5	97.5 ± 0.6	98.0 ± 0.5
ecoli	86.6 ± 0.7	87.1 ± 0.8	86.2 ± 1.1	87.2 ± 1.1
flare	82.3 ± 0.6	79.0 ± 1.1	80.7 ± 1.3	80.8 ± 1.0
glas	94.8 ± 0.9	96.8 ± 0.8	96.8 ± 0.7	96.8 ± 0.9
hear	84.6 ± 1.3	82.7 ± 1.7	82.5 ± 1.4	82.7 ± 1.2
img	96.8 ± 0.3	97.4 ± 0.3	97.1 ± 0.2	97.4 ± 0.2
ionos	90.3 ± 0.8	91.6 ± 0.8	92.1 ± 0.8	92.7 ± 0.8
mok1	98.8 ± 0.9	99.5 ± 0.5	100.00 ± 0.00	100.00 ± 0.00
mok2	79 ± 2	85.6 ± 1.0	87.8 ± 1.2	88.6 ± 1.4
pima	76.7 ± 1.0	77.5 ± 1.1	75.5 ± 1.1	76.4 ± 1.2
survi	74.4 ± 1.2	74.4 ± 1.1	74.6 ± 1.2	74.8 ± 1.3
vote	95.8 ± 0.6	96.1 ± 0.6	95.9 ± 0.6	96.1 ± 0.6
vowel	91.6 ± 0.5	95.4 ± 0.7	96.8 ± 0.5	97.4 ± 0.6
wdbc	96.0 ± 0.4	96.1 ± 0.5	97.3 ± 0.4	96.5 ± 0.7

Table C.60: Performance - *WCB* of *MF* networks and *Boosting Combiner*

Database	3-net	9-net	20-net	40-net
aritm	73.8 ± 1.2	74.9 ± 1.3	75.4 ± 1.4	74.1 ± 1.1
bala	95.9 ± 0.5	95.8 ± 0.6	96.6 ± 0.4	96.6 ± 0.3
band	72.5 ± 1.3	73.5 ± 1.9	73.5 ± 1.2	75.1 ± 1.6
bupa	70.9 ± 1.9	71.1 ± 1.5	74.0 ± 1.3	73.4 ± 1.1
cred	86.8 ± 0.6	86.8 ± 0.5	85.6 ± 1.0	87.0 ± 0.7
derma	97.5 ± 0.7	97.9 ± 0.5	97.0 ± 0.7	98.0 ± 0.5
ecoli	86.6 ± 0.6	87.1 ± 0.6	88.4 ± 0.9	88.2 ± 0.6
flare	81.8 ± 0.6	79.8 ± 1.3	82.4 ± 0.7	82.5 ± 0.7
glas	93.6 ± 0.9	96.0 ± 0.8	96.8 ± 0.7	96.0 ± 0.7
hear	84.4 ± 1.4	83.2 ± 1.4	79.8 ± 1.3	83.1 ± 1.4
img	96.8 ± 0.3	97.2 ± 0.2	97.1 ± 0.3	97.2 ± 0.2
ionos	90.1 ± 1.1	90.7 ± 1.1	92.1 ± 0.8	92.1 ± 0.7
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	79 ± 2	85.3 ± 1.2	87.4 ± 1.0	87.5 ± 1.1
pima	76.8 ± 1.1	77.4 ± 0.8	75.3 ± 0.9	76.4 ± 1.2
survi	74.6 ± 1.3	74.9 ± 1.5	74.6 ± 1.4	74.9 ± 1.4
vote	96.3 ± 0.7	96.0 ± 0.6	95.8 ± 0.7	96.0 ± 0.7
vowel	89.0 ± 0.5	94.5 ± 0.8	97.2 ± 0.5	97.1 ± 0.5
wdbc	96.4 ± 0.5	96.5 ± 0.5	97.3 ± 0.4	96.3 ± 0.5

Table C.61: Performance - *CVCv2Adaboost* of *MF* networks and *Output Average*

Database	3-net	9-net	20-net	40-net
aritm	74.4 ± 1.5	75.6 ± 1.1	72.4 ± 1.4	68.9 ± 1.8
bala	96.3 ± 0.5	96.3 ± 0.6	96.2 ± 0.6	95.4 ± 0.6
band	71.5 ± 0.9	72.7 ± 1.4	72.7 ± 1.6	66 ± 3
bupa	70 ± 2	70.9 ± 1.6	71.7 ± 1.4	69 ± 2
cred	84.8 ± 1.0	85.0 ± 0.8	86.5 ± 0.6	86.2 ± 0.8
derma	96.9 ± 0.6	97.3 ± 0.6	97.0 ± 0.4	96.8 ± 0.6
ecoli	87.6 ± 0.9	85.7 ± 0.9	82.6 ± 1.5	82.1 ± 2.0
flare	80.2 ± 0.9	79.3 ± 0.9	79.6 ± 0.8	77.3 ± 0.7
glas	92.8 ± 1.0	96.0 ± 0.9	96.0 ± 1.1	94.2 ± 2.0
hear	81.7 ± 1.3	80.5 ± 1.8	82.4 ± 1.1	78.1 ± 1.2
img	96.6 ± 0.3	97.33 ± 0.19	97.4 ± 0.2	97.4 ± 0.3
ionos	89.7 ± 0.9	91.3 ± 1.0	91.6 ± 1.2	91.4 ± 1.8
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	78 ± 2	85.8 ± 1.3	84.1 ± 2.0	77.3 ± 1.6
pima	75.6 ± 1.0	74.8 ± 1.2	73.5 ± 1.4	73.1 ± 1.1
survi	75.6 ± 1.5	71 ± 2	69.2 ± 1.4	72.3 ± 1.7
vote	95.8 ± 0.6	95.4 ± 0.5	95.9 ± 0.6	91.6 ± 1.6
vowel	89.4 ± 0.9	96.8 ± 0.3	98.0 ± 0.5	97.1 ± 0.5
wdbc	96.6 ± 0.2	96.5 ± 0.7	96.0 ± 0.5	96.1 ± 0.5

Table C.62: Performance - *CVCv2Adaboost* of *MF* networks and *Boosting Combiner*

Database	3-net	9-net	20-net	40-net
aritm	72.5 ± 1.5	73.4 ± 0.9	72.3 ± 1.4	71.0 ± 1.1
bala	95.1 ± 0.5	95.4 ± 0.5	96.2 ± 0.6	95.8 ± 0.7
band	68.9 ± 1.5	72.4 ± 1.6	71.3 ± 1.9	73.1 ± 1.4
bupa	68 ± 2	70 ± 2	69.9 ± 1.7	70.3 ± 1.1
cred	85.2 ± 0.8	86.4 ± 0.7	85.3 ± 0.7	84.9 ± 0.7
derma	96.6 ± 0.6	97.2 ± 0.6	97.0 ± 0.5	97.3 ± 0.6
ecoli	85.6 ± 1.3	83.8 ± 0.9	86.3 ± 0.8	84.3 ± 1.1
flare	82.0 ± 0.9	81.0 ± 1.1	79.7 ± 0.9	80.1 ± 0.7
glas	92.4 ± 0.9	96.6 ± 0.8	96.4 ± 1.2	96.2 ± 1.0
hear	80.7 ± 1.3	81.2 ± 1.5	82.7 ± 1.6	81.2 ± 1.5
img	96.1 ± 0.3	97.48 ± 0.16	97.46 ± 0.15	97.70 ± 0.12
ionos	89.1 ± 0.7	91.7 ± 0.6	90.7 ± 1.0	92.6 ± 0.6
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	74.8 ± 2.0	86.1 ± 1.4	87.3 ± 0.8	87.5 ± 1.2
pima	75.4 ± 0.8	75.9 ± 0.8	73.2 ± 1.3	71.9 ± 1.2
survi	73.3 ± 1.6	71.6 ± 1.4	69.0 ± 1.5	68.9 ± 1.3
vote	95.8 ± 0.5	95.5 ± 0.6	95.3 ± 0.6	95.6 ± 0.5
vowel	85.0 ± 0.8	96.4 ± 0.4	98.1 ± 0.4	97.1 ± 0.6
wdbc	96.5 ± 0.4	96.7 ± 0.5	96.4 ± 0.6	95.9 ± 0.5

Table C.63: Performance - *CVCv3Adaboost* and *Output Average*

Database	3-net	9-net	20-net	40-net
aritm	73.3 ± 1.6	72.9 ± 1.2	75.7 ± 1.4	74.4 ± 1.4
bala	96.1 ± 0.5	95.9 ± 0.4	95.8 ± 0.4	96.2 ± 0.4
band	72.4 ± 1.6	73.6 ± 1.5	73.6 ± 1.0	74.0 ± 1.2
bupa	70.4 ± 1.9	72.9 ± 1.3	71.7 ± 1.2	72 ± 2
cred	84.8 ± 0.7	84.9 ± 1.0	85.1 ± 0.7	84.5 ± 0.8
derma	97.0 ± 0.4	97.0 ± 0.5	96.9 ± 0.5	97.2 ± 0.5
ecoli	87.5 ± 1.0	85.4 ± 0.9	82.5 ± 1.4	80.7 ± 1.5
flare	81.1 ± 1.1	78.7 ± 1.4	78.0 ± 1.6	79.4 ± 1.0
glas	94.4 ± 0.9	95.8 ± 1.0	95.4 ± 1.2	95.8 ± 1.0
hear	80.3 ± 1.6	83.1 ± 1.3	82.2 ± 1.3	79 ± 2
img	96.75 ± 0.19	97.68 ± 0.15	97.74 ± 0.18	97.8 ± 0.2
ionos	90.6 ± 0.9	90.7 ± 0.9	91.9 ± 0.5	92.1 ± 0.6
mok1	99.4 ± 0.6	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	78.5 ± 1.8	81.9 ± 1.3	80.1 ± 1.6	77 ± 2
pima	75.7 ± 0.8	75.4 ± 0.8	72.3 ± 0.7	73.6 ± 0.8
survi	74.1 ± 1.2	71.5 ± 1.6	71.3 ± 1.3	73.6 ± 0.9
vote	94.3 ± 0.7	94.6 ± 0.7	95.5 ± 0.8	95.4 ± 0.7
vowel	90.9 ± 0.8	96.7 ± 0.5	97.7 ± 0.5	98.5 ± 0.4
wdbc	96.1 ± 0.3	96.6 ± 0.5	96.0 ± 0.4	96.3 ± 0.4

Table C.64: Performance - *CVCv3Adaboost* and *Boosting Combiner*

Database	3-net	9-net	20-net	40-net
aritm	72.3 ± 1.4	70.8 ± 1.5	71.4 ± 1.3	72.8 ± 1.1
bala	94.9 ± 0.6	95.4 ± 0.4	95.4 ± 0.7	95.8 ± 0.4
band	71.5 ± 2.0	72 ± 2	71.3 ± 1.8	71.8 ± 1.3
bupa	71.0 ± 1.7	71.7 ± 1.5	71.4 ± 1.3	71.1 ± 1.4
cred	85.7 ± 1.0	84.5 ± 1.1	84.8 ± 0.9	84.3 ± 1.4
derma	96.3 ± 0.7	96.9 ± 0.6	97.2 ± 0.6	97.3 ± 0.5
ecoli	84.7 ± 1.2	86.9 ± 0.8	87.2 ± 1.0	86.5 ± 0.8
flare	81.4 ± 0.9	80.8 ± 1.0	80.2 ± 1.0	80.2 ± 1.1
glas	91.0 ± 2.0	95.0 ± 1.1	95.6 ± 1.3	96.4 ± 1.0
hear	79.8 ± 1.0	81.5 ± 1.2	80.3 ± 1.4	81.2 ± 1.4
img	96.8 ± 0.2	97.4 ± 0.2	97.7 ± 0.2	97.55 ± 0.20
ionos	89.7 ± 1.0	90.6 ± 1.0	90.7 ± 0.7	92.3 ± 0.7
mok1	99.3 ± 0.8	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	80.0 ± 1.6	83.3 ± 1.7	83.9 ± 1.7	84.8 ± 1.5
pima	75.3 ± 1.3	76.2 ± 1.0	75.0 ± 1.1	75.2 ± 0.9
survi	73.0 ± 0.9	72.0 ± 0.9	71.5 ± 1.8	70.3 ± 1.7
vote	94.3 ± 0.6	95.5 ± 0.8	95.3 ± 0.7	95.0 ± 0.7
vowel	88.5 ± 0.7	95.4 ± 0.6	97.7 ± 0.4	98.3 ± 0.4
wdbc	96.0 ± 0.4	96.4 ± 0.4	96.5 ± 0.4	96.4 ± 0.5

Table C.65: Performance - *CVCv2Aveboost* and *Output Average*

Database	3-net	9-net	20-net	40-net
aritm	74.3 ± 1.1	75.7 ± 1.0	75.9 ± 0.9	74.9 ± 1.1
bala	95.4 ± 0.6	96.3 ± 0.7	96.4 ± 0.7	95.6 ± 0.7
band	69.1 ± 1.9	73.3 ± 1.4	76.0 ± 1.0	76.5 ± 1.1
bupa	71.4 ± 1.6	71.4 ± 1.9	72.4 ± 1.5	72.9 ± 1.4
cred	86.5 ± 0.6	85.5 ± 0.5	86.2 ± 0.6	86.3 ± 0.9
derma	96.6 ± 0.7	97.0 ± 0.6	97.6 ± 0.6	97.6 ± 0.6
ecoli	86.5 ± 1.0	86.6 ± 0.8	87.5 ± 1.2	85.9 ± 1.0
flare	82.0 ± 0.6	81.8 ± 0.8	81.5 ± 0.8	79.5 ± 1.0
glas	94.4 ± 1.2	96.2 ± 0.9	96.6 ± 0.8	96.6 ± 0.8
hear	84.2 ± 1.0	82.7 ± 1.5	82.5 ± 1.4	83.2 ± 1.3
img	96.88 ± 0.18	97.4 ± 0.2	97.63 ± 0.17	97.74 ± 0.18
ionos	87.9 ± 1.1	92.1 ± 0.8	92.4 ± 0.6	92.4 ± 0.7
mok1	99.88 ± 0.13	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	78.6 ± 1.8	89.5 ± 1.9	90.5 ± 1.6	89.4 ± 1.0
pima	76.8 ± 0.8	77.0 ± 0.8	75.7 ± 0.8	74.6 ± 0.8
survi	74.8 ± 1.5	73.8 ± 1.3	72.5 ± 1.1	72.3 ± 1.9
vote	96.1 ± 0.7	96.1 ± 0.7	95.8 ± 0.6	95.3 ± 0.7
vowel	88.8 ± 0.5	96.3 ± 0.6	97.3 ± 0.5	98.2 ± 0.3
wdbc	96.6 ± 0.6	96.6 ± 0.4	96.5 ± 0.5	96.1 ± 0.5

Table C.66: Performance - *CVCv2Aveboost* and *Boosting Combiner*

Database	3-net	9-net	20-net	40-net
aritm	74.0 ± 1.2	76.2 ± 0.6	75.7 ± 1.1	74.4 ± 0.9
bala	94.6 ± 0.7	96.5 ± 0.6	96.0 ± 0.6	96.0 ± 0.6
band	69.1 ± 1.7	73.1 ± 1.7	76.0 ± 1.2	75.6 ± 1.2
bupa	69.3 ± 1.6	73.1 ± 1.6	73.9 ± 1.3	73.3 ± 1.5
cred	86.0 ± 0.7	85.8 ± 0.6	86.1 ± 0.8	86.1 ± 1.1
derma	96.9 ± 0.5	97.3 ± 0.6	97.5 ± 0.7	97.3 ± 0.5
ecoli	85.7 ± 0.8	86.3 ± 0.8	86.0 ± 1.3	85.9 ± 0.9
flare	82.3 ± 0.6	82.6 ± 0.7	81.4 ± 0.8	79.5 ± 1.1
glas	92.2 ± 0.8	95.6 ± 1.0	96.4 ± 0.9	96.4 ± 0.8
hear	84.9 ± 1.0	82.4 ± 1.1	84.2 ± 1.4	82.7 ± 1.3
img	96.49 ± 0.17	97.3 ± 0.2	97.70 ± 0.16	97.74 ± 0.17
ionos	88.6 ± 1.0	91.9 ± 0.7	92.6 ± 0.7	92.1 ± 0.7
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	77.8 ± 1.3	87.9 ± 2.0	89.8 ± 1.7	89.8 ± 1.5
pima	76.8 ± 1.0	76.1 ± 0.9	76.6 ± 0.9	74.8 ± 1.0
survi	74.1 ± 1.4	74.6 ± 1.7	72.8 ± 1.3	72.8 ± 1.4
vote	96.0 ± 0.5	96.3 ± 0.7	96.1 ± 0.5	95.6 ± 0.7
vowel	85.7 ± 0.8	95.5 ± 0.7	97.4 ± 0.4	97.9 ± 0.4
wdbc	96.2 ± 0.4	96.7 ± 0.5	96.7 ± 0.5	96.5 ± 0.6

Table C.67: Performance - *CVCv3Aveboost* and *Output Average*

Database	3-net	9-net	20-net	40-net
aritm	74.5 ± 1.2	75.3 ± 1.3	76.8 ± 1.3	76.6 ± 1.0
bala	96.2 ± 0.6	97.0 ± 0.5	96.6 ± 0.5	96.5 ± 0.6
band	73.1 ± 0.8	73.6 ± 1.4	74.0 ± 1.4	75.5 ± 1.9
bupa	72.9 ± 1.6	72.0 ± 1.7	72.7 ± 1.5	73.0 ± 1.5
cred	86.7 ± 0.8	86.3 ± 0.9	85.5 ± 0.8	85.5 ± 0.9
derma	96.6 ± 0.6	96.9 ± 0.5	97.0 ± 0.6	97.2 ± 0.5
ecoli	87.2 ± 1.0	87.1 ± 1.2	87.5 ± 1.1	86.8 ± 1.1
flare	81.5 ± 0.6	81.6 ± 0.7	81.0 ± 0.8	80.2 ± 1.0
glas	94.6 ± 1.2	96.8 ± 0.8	96.6 ± 0.9	96.6 ± 0.8
hear	82.5 ± 1.7	84.4 ± 1.6	84.2 ± 1.5	84.2 ± 1.3
img	97.0 ± 0.2	97.4 ± 0.2	97.76 ± 0.18	97.83 ± 0.17
ionos	91.3 ± 0.9	91.9 ± 1.0	92.3 ± 0.9	93.0 ± 0.7
mok1	99.0 ± 0.7	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	82 ± 2	86.9 ± 1.3	88.5 ± 2.0	90.5 ± 1.6
pima	77.4 ± 1.1	76.5 ± 1.0	76.6 ± 0.9	76.3 ± 0.9
survi	73.1 ± 1.1	73.9 ± 1.7	74.6 ± 1.3	73.0 ± 1.4
vote	96.0 ± 0.6	96.3 ± 0.7	95.9 ± 0.7	96.0 ± 0.8
vowel	89.7 ± 0.9	95.8 ± 0.6	97.4 ± 0.5	97.5 ± 0.4
wdbc	96.3 ± 0.5	96.5 ± 0.4	97.0 ± 0.4	96.8 ± 0.5

Table C.68: Performance - *CVCv3Aveboost* and *Boosting Combiner*

Database	3-net	9-net	20-net	40-net
aritm	74.4 ± 1.6	74.4 ± 1.5	76.1 ± 1.2	75.9 ± 1.4
bala	95.5 ± 0.6	96.2 ± 0.6	96.4 ± 0.4	96.4 ± 0.6
band	72.5 ± 1.5	72.5 ± 0.7	72.9 ± 1.2	74.4 ± 1.4
bupa	70 ± 2	71.6 ± 1.8	71.4 ± 1.6	72.3 ± 1.6
cred	87.5 ± 0.8	86.9 ± 0.7	85.6 ± 0.9	85.3 ± 0.9
derma	96.1 ± 0.7	97.0 ± 0.6	96.9 ± 0.6	97.5 ± 0.5
ecoli	84.7 ± 1.1	86.9 ± 1.5	87.2 ± 1.1	86.9 ± 1.0
flare	81.6 ± 0.8	82.0 ± 0.8	81.4 ± 0.8	81.2 ± 0.8
glas	93.2 ± 1.1	96.2 ± 0.7	96.4 ± 0.8	96.6 ± 0.8
hear	82.7 ± 1.2	83.4 ± 1.1	83.4 ± 1.3	83.4 ± 1.0
img	96.6 ± 0.3	97.4 ± 0.2	97.70 ± 0.19	97.68 ± 0.17
ionos	90.9 ± 1.0	92.9 ± 0.9	92.4 ± 0.9	92.6 ± 0.9
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	81 ± 2	85.4 ± 1.9	88.9 ± 1.5	91.0 ± 1.1
pima	76.6 ± 1.2	76.8 ± 1.1	76.6 ± 1.1	76.4 ± 0.9
survi	72.5 ± 1.2	73.6 ± 1.4	75.4 ± 1.4	74.3 ± 1.5
vote	96.1 ± 0.8	96.1 ± 0.6	95.8 ± 0.6	95.5 ± 0.7
vowel	87.8 ± 0.9	95.2 ± 0.4	97.3 ± 0.4	97.4 ± 0.3
wdbc	95.8 ± 0.4	96.4 ± 0.3	96.8 ± 0.4	96.5 ± 0.5

Table C.69: Performance - *CVCv2Aveboost2* and *Output Average*

Database	3-net	9-net	20-net	40-net
aritm	76.8 ± 1.8	76.0 ± 1.6	76.1 ± 1.2	77.0 ± 1.6
bala	93.9 ± 0.8	95.1 ± 0.7	95.7 ± 0.6	95.3 ± 0.7
band	71.6 ± 1.5	74.9 ± 1.0	74.2 ± 1.3	74.0 ± 1.1
bupa	71.9 ± 1.7	73.0 ± 1.3	72.9 ± 1.1	73.6 ± 0.9
cred	86.9 ± 0.8	86.5 ± 0.7	87.0 ± 0.6	86.3 ± 0.6
derma	97.0 ± 0.5	97.5 ± 0.4	97.9 ± 0.3	97.7 ± 0.4
ecoli	86.5 ± 0.8	86.5 ± 1.0	86.9 ± 0.9	86.6 ± 1.0
flare	82.0 ± 0.6	81.9 ± 0.6	81.7 ± 0.6	82.0 ± 0.4
glas	93.0 ± 0.8	93.6 ± 0.9	93.8 ± 1.1	94.0 ± 1.1
hear	85.6 ± 1.4	83.4 ± 1.3	83.1 ± 1.3	83.6 ± 1.2
img	95.9 ± 0.3	96.5 ± 0.2	96.77 ± 0.15	96.90 ± 0.20
ionos	89.0 ± 1.2	89.9 ± 1.0	89.4 ± 0.9	90.1 ± 0.9
mok1	99.4 ± 0.6	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	76.4 ± 1.5	84.1 ± 1.5	86.3 ± 1.4	84.8 ± 1.9
pima	77.0 ± 1.1	76.8 ± 1.1	76.4 ± 1.3	75.8 ± 0.9
survi	73.9 ± 1.0	73.9 ± 0.9	73.1 ± 0.9	73.9 ± 0.9
vote	96.6 ± 0.6	96.3 ± 0.6	96.4 ± 0.6	96.5 ± 0.6
vowel	85.8 ± 0.9	90.7 ± 0.6	90.8 ± 0.8	91.2 ± 0.7
wdbc	97.0 ± 0.5	97.3 ± 0.4	97.5 ± 0.4	97.4 ± 0.4

Table C.70: Performance - *CVCv2Aveboost2* and *Boosting Combiner*

Database	3-net	9-net	20-net	40-net
aritm	74.9 ± 1.8	74 ± 2	76.1 ± 1.0	76.1 ± 1.7
bala	93.5 ± 0.7	94.8 ± 0.7	95.1 ± 0.7	95.4 ± 0.6
band	70.4 ± 1.1	75.5 ± 1.0	73.6 ± 1.6	74.7 ± 1.0
bupa	71.3 ± 1.7	72.6 ± 1.6	74.4 ± 1.1	72.1 ± 1.2
cred	86.2 ± 0.7	86.3 ± 0.7	86.5 ± 0.7	86.8 ± 0.6
derma	97.3 ± 0.5	97.5 ± 0.6	97.6 ± 0.4	97.6 ± 0.5
ecoli	85.7 ± 1.1	86.9 ± 1.2	86.6 ± 1.1	87.1 ± 0.9
flare	81.7 ± 0.7	81.5 ± 0.6	81.8 ± 0.5	82.3 ± 0.4
glas	90.8 ± 1.5	94.0 ± 0.8	94.2 ± 1.1	94.6 ± 0.8
hear	83.4 ± 1.3	83.1 ± 1.5	83.2 ± 1.3	83.6 ± 1.5
img	95.6 ± 0.3	96.3 ± 0.2	96.65 ± 0.15	96.8 ± 0.2
ionos	88.1 ± 1.0	90.3 ± 1.0	90.7 ± 0.9	90.7 ± 0.9
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	75.0 ± 1.5	83.5 ± 1.6	88.0 ± 1.6	85.0 ± 1.8
pima	76.1 ± 0.8	76.3 ± 1.1	75.9 ± 1.3	75.7 ± 0.8
survi	74.1 ± 1.0	74.3 ± 1.3	73.0 ± 1.0	73.8 ± 0.9
vote	96.4 ± 0.6	96.4 ± 0.6	96.1 ± 0.6	96.1 ± 0.6
vowel	81.1 ± 0.9	88.9 ± 0.6	89.1 ± 0.8	89.7 ± 0.7
wdbc	97.4 ± 0.5	97.3 ± 0.4	97.1 ± 0.4	97.4 ± 0.3

Table C.71: Performance - *CVCv3Aveboost2* and *Output Average*

Database	3-net	9-net	20-net	40-net
aritm	73.9 ± 1.8	75.3 ± 1.7	76.0 ± 1.7	76.2 ± 1.5
bala	94.6 ± 0.6	95.0 ± 0.4	95.1 ± 0.6	95.3 ± 0.7
band	70 ± 2	72.4 ± 1.1	71.8 ± 1.3	73.3 ± 1.2
bupa	73.6 ± 1.7	73.3 ± 1.4	73.4 ± 1.5	73.3 ± 1.3
cred	86.7 ± 0.7	86.8 ± 0.6	86.8 ± 0.6	86.9 ± 0.6
derma	97.7 ± 0.6	98.0 ± 0.4	97.9 ± 0.4	97.6 ± 0.4
ecoli	86.0 ± 1.2	86.5 ± 1.0	86.2 ± 1.0	86.6 ± 0.9
flare	81.8 ± 0.4	82.2 ± 0.6	82.2 ± 0.6	81.9 ± 0.5
glas	92.4 ± 1.2	93.6 ± 1.2	93.4 ± 1.2	93.2 ± 1.1
hear	84.1 ± 1.1	85.1 ± 1.7	85.4 ± 1.4	85.1 ± 1.3
img	96.7 ± 0.3	96.9 ± 0.3	96.8 ± 0.3	96.7 ± 0.2
ionos	86.9 ± 0.9	88.7 ± 1.2	89.0 ± 1.2	90.0 ± 1.0
mok1	99.4 ± 0.6	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	83 ± 2	84.0 ± 1.7	85.5 ± 1.7	86.3 ± 1.6
pima	76.6 ± 1.1	77.2 ± 1.0	77.0 ± 1.2	77.1 ± 1.0
survi	75.1 ± 1.3	74.3 ± 1.4	74.8 ± 1.7	74.4 ± 1.3
vote	95.9 ± 0.5	96.0 ± 0.5	96.3 ± 0.5	96.3 ± 0.6
vowel	87.6 ± 1.0	89.8 ± 0.6	90.9 ± 0.5	91.3 ± 0.6
wdbc	97.1 ± 0.5	97.2 ± 0.4	97.5 ± 0.4	97.3 ± 0.3

Table C.72: Performance - *CVCv3Aveboost2* and *Boosting Combiner*

Database	3-net	9-net	20-net	40-net
aritm	73.7 ± 1.8	74.6 ± 1.8	74.9 ± 1.7	75.2 ± 1.5
bala	93.8 ± 0.5	94.7 ± 0.5	94.7 ± 0.4	94.8 ± 0.7
band	67.5 ± 1.9	71.6 ± 1.8	72.0 ± 1.3	72.7 ± 1.3
bupa	72.6 ± 1.8	73.3 ± 1.6	73.6 ± 1.5	73.9 ± 1.6
cred	86.6 ± 0.6	87.0 ± 0.6	86.9 ± 0.6	86.8 ± 0.5
derma	97.7 ± 0.6	98.0 ± 0.5	98.2 ± 0.4	97.7 ± 0.4
ecoli	85.3 ± 0.9	86.5 ± 1.0	86.2 ± 1.0	86.0 ± 1.1
flare	82.1 ± 0.5	82.1 ± 0.6	82.2 ± 0.6	82.2 ± 0.6
glas	93.0 ± 1.0	93.0 ± 1.0	93.0 ± 1.2	93.8 ± 1.1
hear	84.4 ± 1.4	84.4 ± 1.6	84.7 ± 1.5	84.6 ± 1.6
img	96.0 ± 0.3	96.7 ± 0.3	96.8 ± 0.3	96.7 ± 0.3
ionos	87.4 ± 0.8	88.7 ± 1.2	88.6 ± 1.2	89.0 ± 1.2
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	81.3 ± 1.6	83.1 ± 1.5	84.8 ± 1.6	86.1 ± 1.5
pima	77.3 ± 1.1	76.5 ± 1.1	76.3 ± 1.0	76.5 ± 1.1
survi	74.1 ± 1.2	74.6 ± 1.5	74.4 ± 1.4	74.6 ± 1.5
vote	95.9 ± 0.6	95.6 ± 0.5	96.0 ± 0.4	96.0 ± 0.5
vowel	85.3 ± 1.1	88.3 ± 0.8	89.4 ± 0.7	89.7 ± 0.8
wdbc	96.9 ± 0.5	97.1 ± 0.5	97.2 ± 0.5	97.2 ± 0.4

Table C.73: Performance - *CVCv2Aggreboost* and *Output Average*

Database	3-net	9-net	20-net	40-net
aritm	73.6 ± 1.7	73.7 ± 1.0	75.2 ± 1.1	74.8 ± 1.5
bala	95.6 ± 0.6	96.0 ± 0.3	96.0 ± 0.6	96.6 ± 0.5
band	71.5 ± 0.9	75.1 ± 0.6	74.2 ± 0.9	76.4 ± 1.5
bupa	70.3 ± 1.6	70.3 ± 1.5	70.9 ± 1.3	72.0 ± 1.7
cred	84.2 ± 0.9	85.0 ± 0.9	85.3 ± 0.8	85.5 ± 0.7
derma	96.5 ± 0.5	97.0 ± 0.6	96.9 ± 0.6	97.0 ± 0.6
ecoli	86.5 ± 1.2	86.3 ± 1.1	86.2 ± 1.1	85.9 ± 0.9
flare	80.2 ± 0.9	79.0 ± 1.1	80.5 ± 1.0	79.3 ± 0.8
glas	93.6 ± 1.1	96.2 ± 0.7	96.4 ± 0.8	96.0 ± 1.0
hear	84.1 ± 1.6	82.5 ± 1.4	82.5 ± 1.2	82.4 ± 1.1
img	96.4 ± 0.3	97.33 ± 0.19	97.61 ± 0.16	97.74 ± 0.12
ionos	88.0 ± 0.7	92.1 ± 0.7	92.3 ± 0.6	93.3 ± 0.7
mok1	99.5 ± 0.3	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	78.3 ± 1.7	84.6 ± 1.6	89.3 ± 1.0	89.4 ± 1.0
pima	76.7 ± 0.9	74.6 ± 1.0	76.2 ± 0.6	74.5 ± 1.1
survi	72.5 ± 1.5	72.6 ± 0.8	72.1 ± 1.5	70.2 ± 1.2
vote	95.1 ± 0.9	95.5 ± 0.5	95.9 ± 0.6	95.8 ± 0.8
vowel	90.2 ± 0.6	96.9 ± 0.5	97.8 ± 0.5	98.3 ± 0.4
wdbc	96.9 ± 0.5	96.7 ± 0.5	97.2 ± 0.4	96.9 ± 0.4

Table C.74: Performance - *CVCv2Aggreboost* and *Boosting Combiner*

Database	3-net	9-net	20-net	40-net
aritm	73.2 ± 1.8	73.7 ± 1.0	74.0 ± 1.1	74.7 ± 1.2
bala	94.6 ± 0.5	96.1 ± 0.4	96.4 ± 0.5	96.3 ± 0.6
band	70.4 ± 1.5	73.6 ± 1.5	73.6 ± 0.6	75.6 ± 1.2
bupa	68.7 ± 1.8	69.3 ± 1.7	70.9 ± 1.4	72.9 ± 1.4
cred	84.9 ± 1.0	84.4 ± 0.8	85.0 ± 0.9	85.4 ± 0.9
derma	96.5 ± 0.5	97.2 ± 0.6	97.0 ± 0.6	96.8 ± 0.6
ecoli	86.3 ± 1.2	86.0 ± 1.0	86.9 ± 0.7	86.3 ± 0.8
flare	81.2 ± 0.9	81.7 ± 0.9	80.9 ± 0.7	81.5 ± 0.5
glas	92.6 ± 1.7	95.8 ± 0.9	96.6 ± 0.8	96.2 ± 0.8
hear	82.5 ± 1.3	83.2 ± 1.4	82.9 ± 1.0	83.6 ± 1.4
img	96.3 ± 0.3	97.3 ± 0.2	97.68 ± 0.15	97.66 ± 0.14
ionos	87.3 ± 0.6	90.9 ± 0.9	92.0 ± 0.6	93.3 ± 0.7
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	76.5 ± 1.4	84.9 ± 1.0	88.9 ± 1.1	89.1 ± 1.4
pima	76.2 ± 1.0	76.3 ± 0.9	76.5 ± 0.9	75.1 ± 1.2
survi	72.6 ± 1.3	72.5 ± 1.4	74.1 ± 1.5	69.8 ± 1.6
vote	94.6 ± 0.7	95.8 ± 0.5	95.9 ± 0.6	96.1 ± 0.7
vowel	87.0 ± 1.0	96.4 ± 0.3	97.9 ± 0.5	98.1 ± 0.5
wdbc	96.5 ± 0.6	96.4 ± 0.5	96.8 ± 0.3	96.6 ± 0.4

Table C.75: Performance - *CVCv3Aggreboost* and *Output Average*

Database	3-net	9-net	20-net	40-net
aritm	73.8 ± 1.5	74.5 ± 1.3	76.2 ± 1.1	76.7 ± 1.3
bala	95.3 ± 0.6	95.9 ± 0.4	95.8 ± 0.6	96.2 ± 0.3
band	70.2 ± 1.5	74.4 ± 1.1	74.5 ± 1.6	74.4 ± 1.1
bupa	70.0 ± 1.6	72.0 ± 1.8	72.6 ± 1.3	72.4 ± 1.5
cred	83.8 ± 1.2	84.5 ± 1.0	85.1 ± 1.0	86.6 ± 0.8
derma	97.0 ± 0.6	97.0 ± 0.6	97.7 ± 0.6	97.6 ± 0.5
ecoli	84.4 ± 1.1	85.7 ± 1.4	86.8 ± 1.2	86.9 ± 1.1
flare	78.9 ± 1.3	78.9 ± 1.0	79.1 ± 0.9	79.2 ± 1.0
glas	94.2 ± 1.0	96.6 ± 1.1	95.8 ± 1.1	95.8 ± 1.2
hear	81.5 ± 1.2	80.2 ± 1.2	83.1 ± 1.1	83.4 ± 1.2
img	96.9 ± 0.4	97.4 ± 0.2	97.44 ± 0.18	97.59 ± 0.12
ionos	89.7 ± 0.9	91.3 ± 0.8	92.3 ± 0.9	92.9 ± 0.7
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	81.5 ± 1.7	82 ± 2	84.4 ± 1.9	87.0 ± 1.6
pima	76.4 ± 1.2	74.6 ± 1.3	75.2 ± 1.1	76.3 ± 0.8
survi	73.3 ± 1.3	74.6 ± 1.2	74.6 ± 1.2	73.3 ± 1.6
vote	95.1 ± 0.8	95.9 ± 0.7	96.0 ± 0.6	95.8 ± 0.6
vowel	92.3 ± 0.4	97.0 ± 0.5	98.2 ± 0.5	98.1 ± 0.7
wdbc	95.4 ± 0.5	95.9 ± 0.4	96.5 ± 0.4	96.4 ± 0.4

Table C.76: Performance - *CVCv3Aggreboost* and *Boosting Combiner*

Database	3-net	9-net	20-net	40-net
aritm	71.1 ± 1.8	74.0 ± 1.2	74.9 ± 0.7	76.7 ± 1.2
bala	94.4 ± 0.6	95.0 ± 0.5	95.4 ± 0.5	95.9 ± 0.4
band	70.0 ± 1.6	71.8 ± 1.1	72.9 ± 1.4	74.5 ± 1.1
bupa	66.9 ± 1.7	70.0 ± 1.4	72.0 ± 1.3	73.0 ± 1.5
cred	84.5 ± 1.2	86.2 ± 0.7	85.8 ± 0.9	86.9 ± 0.7
derma	96.8 ± 0.8	97.0 ± 0.6	97.5 ± 0.5	97.2 ± 0.5
ecoli	84.3 ± 1.2	87.1 ± 1.1	87.4 ± 1.0	87.8 ± 0.8
flare	82.1 ± 0.6	81.7 ± 0.9	81.9 ± 0.8	82.4 ± 0.7
glas	93.0 ± 1.0	94.8 ± 1.1	95.8 ± 1.1	96.0 ± 1.1
hear	82.7 ± 1.2	81.9 ± 1.1	83.1 ± 1.3	83.4 ± 1.7
img	96.6 ± 0.4	97.4 ± 0.2	97.48 ± 0.18	97.42 ± 0.15
ionos	89.4 ± 1.1	90.3 ± 1.1	91.4 ± 0.8	92.1 ± 0.5
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	81 ± 2	83.5 ± 1.7	85.0 ± 1.8	87.8 ± 1.5
pima	75.4 ± 1.0	74.8 ± 1.0	75.9 ± 1.1	75.4 ± 1.1
survi	72.1 ± 1.6	73.1 ± 1.3	74.1 ± 1.3	74.4 ± 1.0
vote	95.1 ± 0.6	96.1 ± 0.6	95.9 ± 0.5	96.0 ± 0.7
vowel	88.4 ± 0.7	96.5 ± 0.3	97.8 ± 0.5	98.3 ± 0.5
wdbc	95.7 ± 0.6	96.5 ± 0.5	96.8 ± 0.5	96.7 ± 0.5

Table C.77: Performance - *CVCv2Conserboost* and *Output Average*

Database	3-net	9-net	20-net	40-net
aritm	75.4 ± 1.3	75.5 ± 0.9	74.6 ± 1.2	73.8 ± 1.0
bala	96.2 ± 0.5	96.6 ± 0.7	96.3 ± 0.5	96.1 ± 0.4
band	71.8 ± 2.0	72.7 ± 1.5	75.6 ± 1.0	75.3 ± 1.7
bupa	69 ± 2	72.4 ± 1.2	71.4 ± 1.7	72.6 ± 1.0
cred	86.2 ± 0.8	85.6 ± 0.6	87.1 ± 0.7	85.5 ± 0.5
derma	97.3 ± 0.4	97.2 ± 0.7	97.3 ± 0.6	97.3 ± 0.6
ecoli	86.5 ± 1.3	86.6 ± 0.9	87.1 ± 0.8	87.6 ± 0.8
flare	83.1 ± 0.6	81.4 ± 0.4	81.3 ± 0.5	79.9 ± 0.7
glas	94.6 ± 0.9	96.2 ± 0.9	96.8 ± 0.9	95.8 ± 1.1
hear	83.9 ± 1.0	82.7 ± 1.3	83.2 ± 1.2	82.5 ± 1.7
img	97.18 ± 0.17	97.5 ± 0.2	97.63 ± 0.16	97.83 ± 0.17
ionos	89.0 ± 0.8	92.3 ± 0.9	92.0 ± 0.8	92.4 ± 0.8
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	77.9 ± 1.9	87.5 ± 1.5	91.6 ± 1.3	90.5 ± 0.9
pima	76.8 ± 1.0	77.0 ± 0.6	75.5 ± 0.7	74.8 ± 1.0
survi	75.6 ± 1.4	75.6 ± 1.2	71.5 ± 1.1	71.5 ± 1.4
vote	95.9 ± 0.6	95.8 ± 0.6	95.9 ± 0.5	96.0 ± 0.6
vowel	90.0 ± 0.6	97.0 ± 0.3	97.8 ± 0.4	98.0 ± 0.4
wdbc	96.8 ± 0.4	96.5 ± 0.5	96.7 ± 0.4	96.8 ± 0.5

Table C.78: Performance - *CVCv2Conserboost* and *Boosting Combiner*

Database	3-net	9-net	20-net	40-net
aritm	74.3 ± 1.6	73.9 ± 1.1	74.5 ± 1.2	74.6 ± 1.3
bala	94.8 ± 0.6	95.8 ± 0.5	96.6 ± 0.6	96.6 ± 0.5
band	67.6 ± 1.5	73.1 ± 1.6	75.6 ± 1.2	74.7 ± 1.4
bupa	69.3 ± 1.9	71.7 ± 1.6	72.3 ± 1.1	73.0 ± 1.1
cred	86.3 ± 0.6	85.7 ± 0.5	85.9 ± 0.8	85.3 ± 0.5
derma	97.0 ± 0.6	97.3 ± 0.6	97.2 ± 0.7	97.0 ± 0.7
ecoli	85.9 ± 1.3	86.5 ± 0.8	87.1 ± 0.8	87.6 ± 0.7
flare	82.4 ± 0.8	81.5 ± 0.8	81.5 ± 0.9	80.7 ± 0.7
glas	93.2 ± 1.0	96.4 ± 0.7	96.2 ± 0.9	96.2 ± 0.9
hear	83.4 ± 1.5	83.6 ± 0.6	84.2 ± 1.4	83.6 ± 1.6
img	96.88 ± 0.17	97.35 ± 0.19	97.57 ± 0.17	97.68 ± 0.18
ionos	89.1 ± 0.8	92.4 ± 0.7	92.1 ± 0.8	92.4 ± 0.7
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	77 ± 2	88.4 ± 1.2	89.9 ± 1.2	90.1 ± 1.2
pima	77.0 ± 1.0	76.9 ± 0.9	76.6 ± 1.0	75.0 ± 0.9
survi	74.1 ± 1.5	74.3 ± 1.4	73.3 ± 1.6	72.1 ± 1.8
vote	95.8 ± 0.5	95.8 ± 0.6	96.1 ± 0.6	95.9 ± 0.6
vowel	86.0 ± 0.7	96.5 ± 0.3	97.8 ± 0.4	97.6 ± 0.5
wdbc	96.5 ± 0.4	96.4 ± 0.5	96.8 ± 0.5	96.6 ± 0.4

Table C.79: Performance - *CVCv3Conserboost* and *Output Average*

Database	3-net	9-net	20-net	40-net
aritm	74.1 ± 1.4	76.7 ± 1.2	76.7 ± 1.2	77.0 ± 1.3
bala	95.7 ± 0.4	96.3 ± 0.4	96.2 ± 0.5	96.2 ± 0.5
band	72.7 ± 1.6	72.5 ± 1.2	75.5 ± 1.6	74.9 ± 1.3
bupa	71.7 ± 1.4	72.7 ± 1.2	73.7 ± 1.3	74.1 ± 1.1
cred	86.8 ± 0.9	86.4 ± 0.7	85.6 ± 0.6	86.5 ± 0.5
derma	96.8 ± 0.8	97.2 ± 0.5	97.3 ± 0.6	96.9 ± 0.5
ecoli	86.5 ± 0.9	86.9 ± 0.9	87.8 ± 0.7	87.8 ± 0.9
flare	81.6 ± 0.6	80.5 ± 0.8	81.0 ± 0.6	80.9 ± 0.8
glas	94.6 ± 0.8	95.8 ± 0.7	96.2 ± 1.0	96.2 ± 1.0
hear	83.9 ± 1.6	84.6 ± 1.4	84.9 ± 1.3	84.4 ± 1.3
img	97.0 ± 0.2	97.68 ± 0.15	97.57 ± 0.15	97.74 ± 0.14
ionos	90.0 ± 1.1	92.1 ± 0.7	92.1 ± 0.6	92.6 ± 0.6
mok1	99.6 ± 0.4	99.88 ± 0.13	100.00 ± 0.00	100.00 ± 0.00
mok2	80.6 ± 1.5	88.6 ± 1.3	91.0 ± 1.5	91.1 ± 1.1
pima	77.0 ± 0.9	76.7 ± 0.9	76.8 ± 0.9	77.0 ± 1.0
survi	73.9 ± 1.5	73.1 ± 1.7	72.3 ± 1.6	73.3 ± 1.4
vote	96.5 ± 0.6	96.4 ± 0.7	95.8 ± 0.7	95.9 ± 0.7
vowel	91.2 ± 0.8	95.8 ± 0.4	97.7 ± 0.3	98.3 ± 0.3
wdbc	96.6 ± 0.4	96.6 ± 0.5	96.8 ± 0.4	96.9 ± 0.5

Table C.80: Performance - *CVCv3Conserboost* and *Boosting Combiner*

Database	3-net	9-net	20-net	40-net
aritm	74 ± 2	74.8 ± 1.4	75.7 ± 1.4	76.1 ± 1.4
bala	95.3 ± 0.5	96.2 ± 0.5	96.6 ± 0.4	96.6 ± 0.4
band	70.7 ± 1.3	71.3 ± 1.3	74.2 ± 1.5	74.5 ± 1.3
bupa	69.4 ± 1.5	73.6 ± 1.2	73.7 ± 1.6	74.0 ± 1.4
cred	86.5 ± 0.9	86.1 ± 0.8	86.6 ± 0.7	86.7 ± 0.7
derma	96.3 ± 0.9	97.5 ± 0.5	97.5 ± 0.5	97.7 ± 0.4
ecoli	85.3 ± 1.0	86.3 ± 0.8	88.5 ± 0.7	88.2 ± 0.8
flare	82.0 ± 0.8	81.6 ± 0.9	81.7 ± 0.5	82.5 ± 0.5
glas	93.0 ± 0.9	95.8 ± 0.7	95.8 ± 1.1	96.8 ± 0.7
hear	83.2 ± 1.6	84.2 ± 1.3	84.4 ± 1.2	84.9 ± 1.3
img	96.8 ± 0.2	97.57 ± 0.16	97.57 ± 0.14	97.72 ± 0.14
ionos	89.6 ± 1.0	92.1 ± 0.9	91.9 ± 0.6	92.1 ± 0.5
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	79.0 ± 1.2	86.6 ± 1.6	90.4 ± 1.4	89.5 ± 1.4
pima	75.9 ± 1.1	76.5 ± 1.0	76.8 ± 1.1	76.5 ± 0.9
survi	74.3 ± 1.1	73.9 ± 1.3	72.5 ± 1.3	73.3 ± 1.3
vote	96.0 ± 0.6	96.3 ± 0.6	95.9 ± 0.7	96.0 ± 0.6
vowel	88.4 ± 0.7	95.6 ± 0.3	97.3 ± 0.3	98.2 ± 0.3
wdbc	96.5 ± 0.3	96.5 ± 0.4	96.7 ± 0.5	96.5 ± 0.4

Table C.81: Performance - *CVCv2Inverseboost* and *Output Average*

Database	3-net	9-net	20-net	40-net
aritm	75 ± 2	74.4 ± 1.4	68.9 ± 1.4	56.6 ± 0.9
bala	92.2 ± 0.8	91.8 ± 0.8	88.5 ± 1.4	86.8 ± 1.5
band	72 ± 2	69.5 ± 0.9	65.1 ± 1.1	65.3 ± 1.2
bupa	72.1 ± 1.6	70.1 ± 1.9	63.6 ± 1.5	59.7 ± 0.6
cred	87.1 ± 0.5	87.4 ± 0.5	87.4 ± 0.5	87.8 ± 0.6
derma	96.8 ± 0.7	93.9 ± 1.3	94.4 ± 1.5	91.0 ± 1.5
ecoli	84.6 ± 1.2	84.7 ± 1.9	80.9 ± 1.5	73 ± 2
flare	81.6 ± 0.5	81.9 ± 0.6	81.3 ± 0.5	81.3 ± 0.5
glas	88 ± 2	85.2 ± 1.6	80.6 ± 1.4	77 ± 3
hear	83.1 ± 1.2	83.7 ± 1.3	81.4 ± 1.4	83.6 ± 1.6
img	95.6 ± 0.2	95.1 ± 0.3	93.6 ± 0.4	92.8 ± 0.3
ionos	87.0 ± 0.8	85.3 ± 0.9	80.4 ± 1.8	65.3 ± 1.9
mok1	99.4 ± 0.6	95 ± 3	93 ± 3	83 ± 3
mok2	74.1 ± 1.9	75 ± 2	67.9 ± 1.0	66.5 ± 1.0
pima	76.1 ± 1.0	75.5 ± 1.2	73.9 ± 0.6	65.5 ± 1.1
survi	74.3 ± 1.6	73.6 ± 0.9	74.8 ± 0.7	74.8 ± 0.7
vote	95.3 ± 0.6	95.9 ± 0.5	95.8 ± 0.6	94.9 ± 0.6
vowel	81.2 ± 0.9	80.0 ± 0.8	74.3 ± 1.0	68.2 ± 1.7
wdbc	96.7 ± 0.4	97.2 ± 0.4	96.7 ± 0.2	96.8 ± 0.4

Table C.82: Performance - *CVCv2Inverseboost* and *Boosting Combiner*

Database	3-net	9-net	20-net	40-net
aritm	74.8 ± 1.5	73.7 ± 1.9	66.9 ± 1.2	56.8 ± 1.0
bala	92.0 ± 0.8	91.8 ± 0.9	86.9 ± 1.2	86.5 ± 1.3
band	70 ± 2	67.8 ± 1.3	65.3 ± 1.2	65.3 ± 1.2
bupa	71.4 ± 1.8	69.1 ± 1.6	61.6 ± 1.3	59.7 ± 0.6
cred	87.0 ± 0.5	87.5 ± 0.5	87.2 ± 0.6	87.8 ± 0.5
derma	96.8 ± 0.7	94.1 ± 1.4	92.8 ± 1.7	87 ± 3
ecoli	84.4 ± 1.2	84.1 ± 1.7	80.7 ± 1.6	66 ± 3
flare	81.5 ± 0.6	82.1 ± 0.6	81.3 ± 0.5	81.3 ± 0.5
glas	88 ± 2	85.4 ± 1.5	78 ± 2	74 ± 3
hear	83.7 ± 1.2	83.4 ± 1.0	81.5 ± 1.6	81 ± 3
img	95.5 ± 0.3	94.7 ± 0.3	93.3 ± 0.4	92.0 ± 0.3
ionos	86.7 ± 0.5	85.1 ± 0.6	77.7 ± 1.9	65.0 ± 1.8
mok1	100.00 ± 0.00	96 ± 3	95 ± 3	76 ± 5
mok2	73.1 ± 1.2	75 ± 2	66.9 ± 1.2	66.5 ± 1.0
pima	75.8 ± 1.1	74.2 ± 1.5	73.1 ± 0.9	65.0 ± 0.7
survi	73.9 ± 1.8	74.9 ± 0.8	74.8 ± 0.7	74.8 ± 0.7
vote	95.3 ± 0.5	95.9 ± 0.6	95.5 ± 0.6	94.5 ± 0.5
vowel	79.4 ± 0.9	77.5 ± 0.9	71.9 ± 1.2	66 ± 3
wdbc	96.8 ± 0.4	97.1 ± 0.4	96.9 ± 0.3	96.8 ± 0.4

Table C.83: Performance - *CVCv3Inverseboost* and *Output Average*

Database	3-net	9-net	20-net	40-net
aritm	75.1 ± 1.4	73.3 ± 1.5	69.7 ± 1.7	63.4 ± 1.7
bala	93.4 ± 0.9	91.9 ± 1.0	90.6 ± 1.1	89.2 ± 1.1
band	73.3 ± 1.4	70.5 ± 1.6	65.5 ± 1.1	65.3 ± 1.2
bupa	71.6 ± 1.5	69.1 ± 1.4	62.0 ± 1.3	59.7 ± 0.6
cred	87.5 ± 0.6	87.6 ± 0.5	87.7 ± 0.5	87.7 ± 0.5
derma	97.2 ± 0.5	96.8 ± 0.6	96.2 ± 0.8	94.9 ± 0.8
ecoli	84.7 ± 1.7	84.9 ± 1.6	82.6 ± 1.6	79.6 ± 1.5
flare	81.3 ± 0.6	81.3 ± 0.5	81.3 ± 0.5	81.3 ± 0.5
glas	91.2 ± 1.0	83.0 ± 1.3	79.8 ± 1.4	75 ± 2
hear	83.1 ± 1.5	82.4 ± 1.5	81.9 ± 1.5	81.0 ± 1.6
img	95.87 ± 0.17	95.0 ± 0.2	94.30 ± 0.16	94.04 ± 0.18
ionos	87.0 ± 1.4	85.3 ± 1.2	81.4 ± 1.8	77 ± 2
mok1	97.0 ± 1.3	94.0 ± 1.6	90 ± 2	86 ± 3
mok2	79 ± 2	73.1 ± 1.9	67.5 ± 1.2	66.5 ± 1.0
pima	76.9 ± 1.1	75.9 ± 0.9	73.2 ± 1.1	67.4 ± 1.3
survi	74.1 ± 0.9	74.6 ± 0.7	74.8 ± 0.7	74.8 ± 0.7
vote	95.0 ± 0.7	95.5 ± 0.4	95.3 ± 0.5	94.8 ± 0.5
vowel	84.0 ± 1.0	79.5 ± 1.1	73.6 ± 1.4	67.7 ± 1.3
wdbc	96.9 ± 0.5	97.3 ± 0.4	97.3 ± 0.4	97.4 ± 0.4

Table C.84: Performance - *CVCv3Inverseboost* and *Boosting Combiner*

Database	3-net	9-net	20-net	40-net
aritm	74.8 ± 1.4	73.4 ± 1.7	67.1 ± 1.7	61.5 ± 1.6
bala	93.2 ± 0.9	91.8 ± 1.0	87.6 ± 1.5	85.4 ± 1.3
band	72.9 ± 1.5	71.3 ± 1.6	65.3 ± 1.2	65.3 ± 1.2
bupa	69.9 ± 1.6	68.1 ± 1.5	61.7 ± 1.0	59.7 ± 0.6
cred	87.3 ± 0.6	87.7 ± 0.5	87.7 ± 0.5	87.7 ± 0.5
derma	97.2 ± 0.5	96.6 ± 0.7	95.4 ± 0.8	93.4 ± 1.2
ecoli	84.9 ± 1.6	84.4 ± 1.9	82.1 ± 1.8	78.4 ± 1.1
flare	81.3 ± 0.6	81.3 ± 0.5	81.3 ± 0.5	81.3 ± 0.5
glas	88.4 ± 1.3	82.6 ± 1.3	76 ± 2	74 ± 2
hear	83.7 ± 1.7	81.7 ± 1.6	80.7 ± 1.5	79.8 ± 1.5
img	95.68 ± 0.20	94.9 ± 0.2	94.04 ± 0.19	93.5 ± 0.2
ionos	87.4 ± 1.4	85.3 ± 1.2	80.9 ± 1.6	76.6 ± 2.0
mok1	97.1 ± 1.2	94.1 ± 2.0	91 ± 2	85 ± 3
mok2	78 ± 3	72.5 ± 1.9	67.3 ± 1.0	66.5 ± 1.0
pima	77.0 ± 1.1	75.5 ± 0.9	72.1 ± 1.5	66.0 ± 1.2
survi	73.9 ± 1.0	74.4 ± 0.7	74.8 ± 0.7	74.8 ± 0.7
vote	95.3 ± 0.6	95.8 ± 0.5	95.3 ± 0.4	94.8 ± 0.5
vowel	82.5 ± 1.3	76.8 ± 0.9	70.4 ± 1.5	65.1 ± 1.2
wdbc	96.9 ± 0.6	97.2 ± 0.4	97.3 ± 0.5	97.3 ± 0.4

Table C.85: Performance - *CVCv2ACB* and *Output Average*

Database	3-net	9-net	20-net	40-net
aritm	74.1 ± 1.4	76.4 ± 1.3	76.2 ± 1.3	76.2 ± 1.5
bala	94.6 ± 0.6	96.7 ± 0.4	96.6 ± 0.6	97.0 ± 0.5
band	70.7 ± 1.4	73.3 ± 1.0	74.9 ± 1.0	74.2 ± 1.1
bupa	70.4 ± 1.5	72.6 ± 1.6	73.7 ± 1.6	73.3 ± 1.4
cred	86.6 ± 0.8	87.2 ± 0.7	86.2 ± 0.8	87.0 ± 0.7
derma	97.3 ± 0.4	97.9 ± 0.5	97.9 ± 0.4	97.0 ± 0.6
ecoli	87.2 ± 1.3	87.6 ± 0.8	87.6 ± 0.7	87.1 ± 0.9
flare	82.3 ± 0.7	81.7 ± 0.5	82.0 ± 0.6	81.0 ± 0.7
glas	92.6 ± 1.2	96.8 ± 0.8	96.8 ± 0.7	96.6 ± 0.7
hear	85.3 ± 1.7	84.7 ± 1.4	83.2 ± 1.4	83.4 ± 1.2
img	96.5 ± 0.2	97.5 ± 0.2	97.55 ± 0.18	97.66 ± 0.18
ionos	89.1 ± 1.2	91.9 ± 0.7	92.0 ± 0.9	92.3 ± 0.8
mok1	98.1 ± 1.2	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	77.9 ± 2.0	86.4 ± 1.5	88.5 ± 1.0	88.5 ± 1.1
pima	76.5 ± 0.9	76.8 ± 0.8	75.7 ± 1.0	76.0 ± 0.9
survi	74.3 ± 1.4	73.0 ± 1.4	73.4 ± 1.2	71.0 ± 1.0
vote	96.4 ± 0.5	96.4 ± 0.8	96.1 ± 0.7	95.6 ± 0.7
vowel	87.3 ± 0.9	94.2 ± 0.6	96.2 ± 0.6	97.4 ± 0.4
wdbc	97.2 ± 0.3	96.7 ± 0.4	96.5 ± 0.6	96.5 ± 0.5

Table C.86: Performance - *CVCv2ACB* and *Boosting Combiner*

Database	3-net	9-net	20-net	40-net
aritm	73.4 ± 1.6	76.4 ± 1.5	75.3 ± 1.4	76.0 ± 1.6
bala	94.4 ± 0.5	96.4 ± 0.5	96.6 ± 0.5	96.6 ± 0.5
band	69.8 ± 2.0	72.4 ± 1.2	75.3 ± 1.1	74.4 ± 1.3
bupa	70.9 ± 1.4	73.0 ± 1.6	72.6 ± 1.4	73.4 ± 1.4
cred	87.0 ± 0.8	86.7 ± 0.7	86.8 ± 0.8	86.6 ± 0.9
derma	97.7 ± 0.3	97.9 ± 0.5	97.6 ± 0.5	97.2 ± 0.6
ecoli	86.2 ± 1.1	86.9 ± 1.0	87.9 ± 0.8	87.2 ± 0.8
flare	82.6 ± 0.7	82.1 ± 0.5	82.3 ± 0.5	81.3 ± 0.9
glas	92.4 ± 1.5	96.4 ± 0.8	95.8 ± 1.0	97.0 ± 0.5
hear	85.3 ± 1.5	85.1 ± 1.2	83.2 ± 1.4	84.4 ± 1.2
img	95.98 ± 0.15	97.3 ± 0.2	97.51 ± 0.18	97.61 ± 0.16
ionos	88.3 ± 1.1	91.7 ± 0.7	91.9 ± 0.8	92.6 ± 0.8
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	77.8 ± 1.3	86.6 ± 1.0	88.8 ± 1.0	89.6 ± 1.1
pima	76.5 ± 0.7	77.1 ± 1.2	76.9 ± 1.0	76.0 ± 0.8
survi	74.1 ± 1.2	72.3 ± 1.3	74.6 ± 1.3	72.1 ± 1.5
vote	95.6 ± 0.4	96.1 ± 0.6	96.3 ± 0.6	95.8 ± 0.7
vowel	85.4 ± 1.0	92.9 ± 0.7	96.1 ± 0.5	97.2 ± 0.4
wdbc	97.1 ± 0.2	96.9 ± 0.5	96.5 ± 0.5	96.4 ± 0.4

Table C.87: Performance - *CVCv3ACB* and *Output Average*

Database	3-net	9-net	20-net	40-net
aritm	75.2 ± 1.6	75.7 ± 0.9	76.7 ± 1.3	77.4 ± 1.3
bala	95.8 ± 0.8	96.8 ± 0.4	97.2 ± 0.3	97.0 ± 0.4
band	71 ± 2	74.2 ± 1.3	74.7 ± 0.9	75.5 ± 1.2
bupa	72.6 ± 1.7	74.0 ± 1.5	74.0 ± 1.3	74.4 ± 1.1
cred	86.7 ± 0.8	86.8 ± 0.6	86.7 ± 0.6	87.0 ± 0.7
derma	97.6 ± 0.5	97.5 ± 0.6	97.2 ± 0.6	97.0 ± 0.7
ecoli	86.3 ± 0.9	87.1 ± 1.1	87.5 ± 0.9	87.6 ± 0.9
flare	81.7 ± 0.7	82.6 ± 0.7	82.7 ± 0.7	82.1 ± 0.7
glas	93.8 ± 0.5	96.2 ± 1.0	96.4 ± 0.9	96.4 ± 0.8
hear	83.4 ± 1.8	84.1 ± 1.2	84.7 ± 1.5	83.6 ± 1.4
img	96.5 ± 0.3	97.3 ± 0.2	97.5 ± 0.2	97.55 ± 0.17
ionos	89.6 ± 0.9	91.4 ± 0.8	92.1 ± 0.8	91.7 ± 0.7
mok1	99.0 ± 1.0	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	81.5 ± 1.3	86 ± 2	87.3 ± 1.4	90.0 ± 1.5
pima	76.5 ± 0.9	76.6 ± 0.9	76.7 ± 0.7	76.9 ± 0.7
survi	74.4 ± 1.1	73.8 ± 1.1	73.9 ± 1.7	74.8 ± 1.6
vote	95.8 ± 0.6	96.1 ± 0.7	96.1 ± 0.6	96.3 ± 0.6
vowel	89.1 ± 1.0	93.9 ± 0.7	95.9 ± 0.6	97.2 ± 0.6
wdbc	96.5 ± 0.5	96.9 ± 0.4	96.5 ± 0.4	97.0 ± 0.5

Table C.88: Performance - *CVCv3ACB* and *Boosting Combiner*

Database	3-net	9-net	20-net	40-net
aritm	73.3 ± 1.4	75.5 ± 1.0	77.0 ± 0.9	77.1 ± 1.1
bala	95.7 ± 0.5	96.2 ± 0.4	97.0 ± 0.3	97.0 ± 0.4
band	73.1 ± 1.6	73.5 ± 1.4	72.4 ± 0.7	74.5 ± 1.2
bupa	71.1 ± 1.7	73.3 ± 1.7	73.9 ± 1.8	74.6 ± 1.3
cred	86.8 ± 0.9	86.7 ± 0.7	86.7 ± 0.6	87.0 ± 0.8
derma	97.6 ± 0.6	97.6 ± 0.6	97.3 ± 0.6	97.2 ± 0.7
ecoli	85.7 ± 1.0	86.3 ± 0.9	86.9 ± 0.9	87.1 ± 0.9
flare	82.0 ± 0.7	82.0 ± 0.5	82.6 ± 0.4	82.4 ± 0.7
glas	93.0 ± 0.8	96.2 ± 0.8	96.0 ± 0.9	96.6 ± 0.8
hear	83.1 ± 1.3	83.9 ± 1.2	83.9 ± 1.4	83.7 ± 1.3
img	96.0 ± 0.2	97.2 ± 0.2	97.59 ± 0.18	97.66 ± 0.17
ionos	89.9 ± 0.9	91.7 ± 0.9	92.1 ± 0.9	92.0 ± 0.8
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	80.3 ± 1.5	84 ± 2	87.0 ± 1.6	89.8 ± 1.5
pima	75.8 ± 0.9	76.2 ± 1.1	76.8 ± 0.8	76.7 ± 0.9
survi	73.6 ± 1.1	72.8 ± 1.2	73.3 ± 1.7	73.3 ± 1.6
vote	95.6 ± 0.5	95.9 ± 0.6	96.3 ± 0.6	96.0 ± 0.4
vowel	86.5 ± 1.0	93.0 ± 0.6	96.1 ± 0.6	96.9 ± 0.6
wdbc	96.7 ± 0.5	96.5 ± 0.4	96.7 ± 0.4	97.2 ± 0.5

Table C.89: Performance - *CVCv2WCB* and *Output Average*

Database	3-net	9-net	20-net	40-net
aritm	74.8 ± 1.7	76.2 ± 1.2	76.3 ± 1.2	74.3 ± 1.4
bala	95.0 ± 0.7	96.6 ± 0.6	96.6 ± 0.4	96.8 ± 0.5
band	69.8 ± 1.8	74.2 ± 1.6	73.6 ± 1.2	74.5 ± 1.1
bupa	65 ± 4	72.7 ± 1.4	73.7 ± 1.2	72.3 ± 1.0
cred	85.3 ± 0.7	85.9 ± 0.7	86.7 ± 0.9	86.7 ± 0.7
derma	96.3 ± 1.1	98.0 ± 0.4	97.9 ± 0.5	97.7 ± 0.5
ecoli	86.8 ± 1.0	87.1 ± 0.8	87.2 ± 1.1	86.2 ± 1.2
flare	82.2 ± 0.8	79.5 ± 0.8	81.1 ± 0.5	80.7 ± 0.8
glas	94.0 ± 0.9	96.4 ± 0.9	96.2 ± 0.9	96.6 ± 0.7
hear	82.4 ± 1.2	80.7 ± 1.9	84.2 ± 1.2	81.9 ± 1.4
img	96.8 ± 0.2	97.42 ± 0.12	97.66 ± 0.14	97.74 ± 0.17
ionos	88.6 ± 1.3	92.6 ± 0.8	92.7 ± 0.8	92.0 ± 0.8
mok1	99.1 ± 0.9	95.6 ± 1.3	100.00 ± 0.00	100.00 ± 0.00
mok2	80 ± 2	84.9 ± 1.3	90.5 ± 0.9	91.3 ± 1.0
pima	76.8 ± 0.9	73.2 ± 1.3	74.7 ± 1.2	75.3 ± 1.2
survi	75.1 ± 1.7	74.1 ± 1.3	72.3 ± 1.3	73.0 ± 1.4
vote	96.5 ± 0.5	96.0 ± 0.6	95.6 ± 0.6	95.4 ± 0.5
vowel	86.7 ± 1.1	96.7 ± 0.5	97.4 ± 0.5	97.2 ± 0.5
wdbc	96.5 ± 0.4	96.7 ± 0.5	95.9 ± 0.6	96.2 ± 0.6

Table C.90: Performance - *CVCv2WCB* and *Boosting Combiner*

Database	3-net	9-net	20-net	40-net
aritm	74.4 ± 1.9	75.3 ± 1.5	76.3 ± 1.4	74.7 ± 1.5
bala	95.0 ± 0.8	96.6 ± 0.5	96.8 ± 0.4	96.8 ± 0.5
band	68.4 ± 1.7	74 ± 2	72.2 ± 1.1	75.3 ± 0.9
bupa	65 ± 4	73.0 ± 1.3	73.9 ± 1.6	72.7 ± 1.1
cred	86.2 ± 0.5	85.2 ± 0.8	86.2 ± 0.9	86.3 ± 0.7
derma	96.2 ± 1.2	98.0 ± 0.5	97.9 ± 0.5	97.7 ± 0.5
ecoli	85.4 ± 1.4	87.1 ± 0.9	87.6 ± 1.0	86.8 ± 0.9
flare	81.1 ± 0.5	81.3 ± 0.7	81.8 ± 0.5	81.6 ± 0.7
glas	93.0 ± 1.1	96.0 ± 0.9	96.2 ± 1.0	96.4 ± 0.7
hear	82.9 ± 1.0	81.9 ± 1.3	83.4 ± 1.2	82.5 ± 1.5
img	96.5 ± 0.3	97.5 ± 0.2	97.66 ± 0.13	97.74 ± 0.16
ionos	89.1 ± 1.3	92.4 ± 0.9	92.3 ± 0.9	91.7 ± 0.8
mok1	100.00 ± 0.00	97.9 ± 1.1	100.00 ± 0.00	100.00 ± 0.00
mok2	78.5 ± 1.7	86.1 ± 1.4	90.3 ± 1.2	91.0 ± 1.6
pima	77.5 ± 1.0	70.8 ± 1.9	75.8 ± 1.1	76.0 ± 0.8
survi	75.1 ± 1.4	73.4 ± 1.3	73.6 ± 1.1	73.6 ± 1.6
vote	96.6 ± 0.5	96.1 ± 0.5	95.9 ± 0.5	96.0 ± 0.6
vowel	83.6 ± 1.2	96.3 ± 0.6	97.3 ± 0.6	97.3 ± 0.4
wdbc	96.2 ± 0.4	96.7 ± 0.5	96.5 ± 0.5	96.4 ± 0.4

Table C.91: Performance - *CVCv3WCB* and *Output Average*

Database	3-net	9-net	20-net	40-net
aritm	74.5 ± 1.4	76.6 ± 0.8	75.1 ± 1.2	77.0 ± 1.3
bala	95.7 ± 0.7	96.3 ± 0.5	96.6 ± 0.3	96.9 ± 0.4
band	73.6 ± 1.9	73.5 ± 1.2	72.0 ± 1.3	75.3 ± 1.1
bupa	71 ± 2	73.3 ± 1.6	72.0 ± 1.5	72.6 ± 1.8
cred	87.2 ± 0.7	84.9 ± 0.9	86.4 ± 0.8	86.5 ± 0.7
derma	97.2 ± 0.7	97.7 ± 0.4	97.7 ± 0.5	97.2 ± 0.5
ecoli	87.6 ± 0.9	86.6 ± 0.9	86.6 ± 0.8	87.4 ± 0.7
flare	80.9 ± 0.6	82.7 ± 0.8	82.3 ± 0.7	81.2 ± 1.0
glas	94.2 ± 1.1	96.0 ± 0.9	96.0 ± 0.9	95.6 ± 0.9
hear	82.5 ± 1.4	83.4 ± 1.2	82.9 ± 1.2	83.7 ± 1.2
img	97.2 ± 0.3	97.5 ± 0.2	97.44 ± 0.18	97.68 ± 0.19
ionos	90.7 ± 0.8	92.6 ± 0.5	92.6 ± 0.7	92.9 ± 0.6
mok1	97.8 ± 1.2	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	81 ± 2	86.5 ± 1.6	89.4 ± 1.4	89.9 ± 1.2
pima	75.2 ± 1.3	74.7 ± 1.2	75.2 ± 1.2	76.3 ± 1.0
survi	73.9 ± 1.0	74.1 ± 1.5	66 ± 4	73.9 ± 1.4
vote	96.0 ± 0.6	95.8 ± 0.8	96.5 ± 0.6	96.3 ± 0.6
vowel	89.7 ± 0.6	95.5 ± 0.5	96.8 ± 0.5	96.9 ± 0.5
wdbc	95.6 ± 0.7	96.7 ± 0.3	97.2 ± 0.4	96.9 ± 0.4

Table C.92: Performance - *CVCv3WCB* and *Boosting Combiner*

Database	3-net	9-net	20-net	40-net
aritm	74.0 ± 1.7	76.6 ± 1.0	75.5 ± 1.1	76.2 ± 1.4
bala	95.4 ± 0.7	96.2 ± 0.4	96.4 ± 0.5	96.6 ± 0.6
band	71.1 ± 1.7	72.0 ± 1.2	72.2 ± 1.3	74.9 ± 1.3
bupa	69.1 ± 1.8	72.9 ± 1.8	71.9 ± 1.9	72.7 ± 1.7
cred	87.3 ± 0.8	85.5 ± 0.9	87.2 ± 0.8	86.6 ± 0.7
derma	96.2 ± 0.8	97.3 ± 0.5	97.5 ± 0.4	97.0 ± 0.5
ecoli	86.3 ± 1.1	86.6 ± 1.3	87.8 ± 0.8	87.5 ± 0.9
flare	80.8 ± 0.5	83.0 ± 0.7	82.1 ± 0.6	81.9 ± 0.7
glas	92.8 ± 1.1	96.2 ± 0.8	96.0 ± 0.7	95.8 ± 0.9
hear	82.5 ± 1.6	82.7 ± 1.3	83.2 ± 0.9	82.9 ± 1.2
img	97.1 ± 0.3	97.3 ± 0.2	97.4 ± 0.2	97.70 ± 0.17
ionos	90.4 ± 1.0	92.9 ± 0.5	92.9 ± 0.7	92.3 ± 0.7
mok1	98.9 ± 0.8	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	79 ± 2	86.5 ± 1.7	89.3 ± 1.0	90.6 ± 1.2
pima	75.7 ± 0.8	75.4 ± 1.3	75.7 ± 1.4	76.0 ± 1.1
survi	74.9 ± 1.0	73.4 ± 1.0	70 ± 2	74.4 ± 1.6
vote	95.9 ± 0.5	96.0 ± 0.6	96.3 ± 0.6	96.3 ± 0.6
vowel	87.8 ± 0.9	95.5 ± 0.6	97.1 ± 0.5	96.9 ± 0.5
wdbc	95.6 ± 0.7	96.7 ± 0.5	97.2 ± 0.6	97.1 ± 0.4

Table C.93: Performance - Combination of *Simple Ensemble* of 3 *MF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	73.5±1.1	73.2±1.1	73.2±1.1	73.1±1	73.5±1.1	73.1±1	73.6±1	73±0.9	73.6±0.9	74.1±1.1	73.5±1.1	74.7±1.4	74.1±1.1	73.5±1.1
bala	96±0.5	94.6±0.8	94.6±0.8	95.4±0.5	95.4±0.5	95.4±0.5	95.5±0.6	96.2±0.7	95.9±0.7	95.7±0.6	96±0.5	96.2±0.5	95.7±0.6	96±0.5
band	73.5±1.2	72.9±1.5	72.9±1.5	73.6±1.2	73.3±1.2	73.6±1.2	72.9±1.3	71.8±1.8	73.6±1.2	72.4±1.6	73.5±1.2	72.4±1.8	72.4±1.6	73.1±1.2
bupa	72.3±1.2	72.1±1.2	72.1±1.2	72.1±1.2	72.3±1.2	72.1±1.2	72.4±1.3	71.1±1.6	72.1±1.2	72±1.4	72.3±1.2	71.3±1.5	71.3±1.7	72.3±1.3
cred	86.5±0.7	86.5±0.7	86.5±0.7	86.4±0.7	86.5±0.7	86.4±0.7	86.7±0.7	86.5±0.8	86.4±0.7	86.8±0.7	86.5±0.7	86.2±0.9	86.9±0.7	86.5±0.7
derma	97.2±0.7	96.9±0.6	96.9±0.6	96.9±0.8	97.3±0.7	97±0.7	97.2±0.7	96.3±0.7	96.9±0.8	97.2±0.7	97.2±0.7	97.3±0.7	97.2±0.7	97.2±0.7
ecoli	86.6±0.8	85.7±1	85.4±0.9	86±0.9	86.6±0.8	86.5±0.8	86.3±0.9	85.9±0.9	86.3±0.9	86.3±0.9	86.6±0.8	86±1.2	85.9±0.7	86.6±0.8
flare	81.8±0.5	81.4±0.6	81.4±0.6	81.5±0.5	81.8±0.5	81.5±0.5	81.7±0.5	81.3±0.6	81.5±0.5	81.7±0.5	81.8±0.5	81.5±0.6	81.8±0.5	81.8±0.5
glas	94±0.8	92.8±1.6	92.6±1.5	93.6±0.9	94.6±0.7	94.4±0.9	94±0.6	94±1.3	94.2±1	94±0.6	94±0.8	94.2±0.7	94±0.6	94±0.8
hear	82.9±1.5	82.4±1.3	82.4±1.3	82.9±1.5	82.9±1.5	82.9±1.5	83.1±1.5	82.9±1.4	82.9±1.5	83.2±1.4	82.9±1.5	83.2±1.5	83.2±1.4	82.9±1.5
img	96.5±0.2	95.7±0.2	95.7±0.2	96.2±0.3	95.8±0.2	95.7±0.2	96.4±0.2	96.7±0.3	96.3±0.3	96.3±0.2	96.5±0.2	96.6±0.3	96.3±0.2	96.5±0.2
ionos	91.1±1.1	90±1.2	90±1.2	91.3±1	91.3±1.2	91.3±1	91.1±1.1	91.3±0.9	91.4±1.1	91.3±1.1	91.1±1.1	91.9±1.1	91.1±1.1	91.1±1.1
mok1	98.3±0.9	97.1±1	97.1±1	98.3±0.9	98.3±0.9	98.3±0.9	98.1±1	98.5±0.8	98.4±0.9	98±1	98.3±0.9	98.5±0.8	98±1	98.3±0.9
mok2	88±2	87±2	87±2	88±2	88±2	88±2	88±2	88±2	89±2	88±2	88±2	88±2	88±2	88±2
pima	75.9±1.2	75.9±1.2	75.9±1.2	75.9±1.3	75.9±1.2	75.9±1.3	75.9±1.2	75.3±1.3	75.8±1.3	76.1±1.2	75.9±1.2	76±1	75.6±1.3	75.9±1.2
survi	74.3±1.3	74.4±1.4	74.4±1.4	74.4±1.4	74.3±1.3	74.4±1.4	73.9±1.4	74.1±1.3	74.3±1.4	74.1±1.4	74.3±1.3	74.3±1.3	74.1±1.3	74.3±1.3
vote	95.6±0.5	95.5±0.6	95.5±0.6	95.5±0.6	95.6±0.5	95.5±0.6	95.6±0.5	95.9±0.6	95.5±0.6	95.6±0.5	95.6±0.5	95.6±0.5	95.6±0.5	95.6±0.5
vowel	88±0.9	84.6±1.2	84.5±1.2	86.9±0.9	86.2±1	85.9±1	86.7±0.8	87.7±1	86.4±1	86.4±0.7	88±0.9	87.8±1	86.3±0.7	88±0.9
wdbc	96.9±0.5	96.9±0.5	96.9±0.5	96.9±0.5	96.9±0.5	96.9±0.5	96.9±0.5	96.9±0.5	96.9±0.5	96.9±0.5	96.9±0.5	96.9±0.5	96.9±0.5	96.9±0.5

Table C.94: Performance - Combination of *Simple Ensemble* of 9 *MF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	73.8±1.1	73.6±1	73.6±1	73.3±0.9	73.7±1	73.3±0.9	73.3±1.1	74.8±1	73.6±0.9	73.3±1.1	73.8±1.1	74.3±0.8	73.3±1.1	73.8±1.1
bala	95.8±0.5	94.5±0.6	94.6±0.6	95.4±0.6	95±0.5	94.9±0.6	96.2±0.7	96.4±0.6	95.2±0.6	96.2±0.7	95.8±0.5	96.2±0.6	96.2±0.7	95.7±0.5
band	72.9±1.5	72±2	72±2	73.5±1.2	72.7±1.6	73.5±1.2	73.3±1.7	73.3±2	73.5±1.4	73.1±1.7	72.9±1.5	73.5±1.5	73.1±1.7	72.9±1.5
bupa	72.4±1.1	73±1.4	73±1.4	72.4±1.2	72.4±1.1	72.4±1.2	72±1.2	71±1.7	72.3±1.2	71.9±1.2	72.4±1.1	71.4±1.4	72.1±1.3	72.4±1.1
cred	86.4±0.7	86.3±0.7	86.3±0.7	86.3±0.7	86.5±0.7	86.3±0.7	86.6±0.7	86.6±0.9	86.4±0.7	86.8±0.7	86.4±0.7	86.5±0.8	86.8±0.7	86.5±0.7
derma	97.5±0.7	97.2±0.6	97.2±0.6	97.5±0.7	97.5±0.7	97.5±0.7	96.9±0.8	97±0.7	96.3±0.8	97.2±0.8	97.5±0.7	97.5±0.6	97.2±0.8	97.5±0.7
ecoli	86.9±0.8	85.2±0.8	84.6±0.8	86±0.9	87.2±0.8	86.6±0.9	87.1±0.9	86.2±1.1	85.2±0.9	86.9±1	86.9±0.8	85.4±0.9	86.2±1	87.1±0.9
flare	81.6±0.4	81.5±0.5	81.5±0.5	81.5±0.5	81.5±0.4	81.5±0.5	81.5±0.4	81.6±0.4	81.6±0.5	81.5±0.5	81.6±0.4	81.4±0.6	81.5±0.5	81.6±0.5
glas	94±0.7	92.8±1.1	92.6±1.3	93.2±0.8	94.8±0.5	94.2±0.7	93.8±0.6	95.6±0.6	92.2±0.9	94.2±0.7	94±0.7	94.4±0.7	93.8±0.8	93.8±0.8
hear	83.1±1.5	83.1±1.4	83.1±1.4	82.9±1.5	82.9±1.5	82.9±1.5	82.9±1.4	83.1±1.3	82.9±1.5	83.1±1.4	83.1±1.5	82.5±1.2	83.1±1.4	82.9±1.5
img	96.7±0.3	96.2±0.2	96.1±0.2	96.6±0.3	96.24±0.18	96.17±0.18	96.5±0.2	96.8±0.3	96.1±0.3	96.3±0.2	96.7±0.3	96.7±0.3	96.3±0.2	96.7±0.3
ionos	90.3±1.1	90±1.1	90±1.1	90.6±1.2	90.4±1.2	90.6±1.2	90.9±1.3	91.9±1	93.1±1.4	90.7±1.2	90.3±1.1	92±1	90.4±1.2	90.4±1.1
mok1	98.8±0.8	98.8±0.9	98.8±0.9	98.3±0.9	99.5±0.5	98.3±0.9	99.5±0.5	99.8±0.3	99.8±0.3	99.5±0.5	98.8±0.8	99.8±0.3	99.4±0.5	99.4±0.5
mok2	90.8±1.8	87±3	87±3	90.3±1.8	91±1.5	90.3±1.8	90±1.2	91.5±0.9	89.6±1.7	89.5±1.6	90.8±1.8	90.9±1	89.6±1.7	91.1±1.5
pima	75.9±1.2	75.7±1.1	75.7±1.1	76.1±1.2	75.9±1.2	76.1±1.2	75.8±1.2	76.1±1.2	75.9±1.2	75.7±1.2	75.9±1.2	75.9±1	75.7±1.3	75.9±1.2
survi	74.3±1.3	74.3±1.3	74.3±1.3	74.1±1.3	74.1±1.4	74.1±1.3	74.6±1.3	73.3±1.3	74.1±1.3	74.4±1.4	74.3±1.3	74.4±1.4	74.1±1.5	74.3±1.3
vote	95.6±0.5	95.6±0.5	95.6±0.5	95.5±0.6	95.6±0.5	95.5±0.6	95.6±0.5	95.8±0.6	95.5±0.6	95.6±0.5	95.6±0.5	95.8±0.6	95.6±0.5	95.6±0.5
vowel	91±0.5	85.8±1	85.9±1	89.6±0.6	87.5±0.8	87.5±0.9	90.3±0.6	92±0.6	82.5±0.9	88.2±0.7	91±0.5	91.3±0.4	87.9±0.8	91±0.5
wdbc	96.9±0.5	97±0.4	97±0.4	96.9±0.5	96.9±0.5	96.9±0.5	96.9±0.5	97.2±0.5	97±0.4	96.9±0.5	96.9±0.5	96.9±0.5	96.9±0.5	96.9±0.5

Table C.95: Performance - Combination of *Simple Ensemble* of 20 *MF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	73.8±1	72.8±1.2	72.8±1.2	73.3±1	73.8±0.9	73.3±1	73.1±1.2	73.5±1.4	73.8±1	–	73.8±1	74.8±1.3	–	
bala	95.8±0.6	94.4±1	94.5±0.9	95.2±0.6	95±0.6	94.6±0.6	96.5±0.7	96.3±0.5	94.9±0.7	–	95.8±0.6	95.4±1.2	–	
band	73.8±1.3	74.2±1.3	74.2±1.3	73.6±1.1	74±1.4	73.6±1.1	73.1±1.4	71.6±1.6	73.5±1.3	–	73.8±1.3	71.5±1.6	–	
bupa	72.3±1.1	72.4±1.4	72.4±1.4	72.3±1.2	72.6±1.1	72.3±1.2	72.3±1.1	69.7±1.5	72.3±1.2	–	72.3±1.1	71.1±1.7	–	
cred	86.6±0.7	86.5±0.7	86.5±0.7	86.3±0.7	86.5±0.7	86.3±0.7	86.1±0.8	85.9±1	86.3±0.7	–	86.6±0.7	85.9±0.9	–	
derma	97.3±0.7	96.9±0.8	96.9±0.8	97.3±0.7	97.6±0.7	97.3±0.7	97±0.7	97.3±0.7	95.5±0.9	–	97.3±0.7	97.2±0.7	–	
ecoli	86.9±0.8	85.2±0.9	85.2±0.9	86.6±1	87.1±0.7	86.9±0.8	87.7±0.7	85.2±1	84.4±0.9	–	86.9±0.8	83.5±1.3	–	
flare	81.5±0.5	81.3±0.5	81.3±0.5	81.5±0.5	81.5±0.5	81.5±0.5	81.4±0.6	81.1±0.8	81.5±0.5	–	81.5±0.5	81.3±0.4	–	
glas	94±0.7	94.2±1.2	94.2±1.2	93.4±0.9	94.8±0.5	94.4±0.8	94.4±0.7	95.8±1	90.6±0.9	–	94±0.7	89.8±1.4	–	
hear	83.1±1.5	82.4±1.3	82.4±1.3	82.4±1.4	82.9±1.5	82.4±1.4	82.9±1.4	82.5±1.3	82.4±1.4	–	83.1±1.5	82.4±1.2	–	
img	96.7±0.2	96±0.3	96±0.3	96.8±0.2	96.34±0.18	96.47±0.15	96.4±0.3	97.3±0.3	95.7±0.3	–	96.7±0.2	96.7±0.3	–	
ionos	90.4±1	89.6±1.1	89.6±1.1	90±1.2	90.7±1.1	90±1.2	91.3±1.1	91.7±1	93.1±1.4	–	90.4±1	91.6±1	–	
mok1	98.3±0.9	97.6±1	97.6±1	98.1±1	99.3±0.6	98.1±1	100±0	100±0	100±0	–	98.3±0.9	100±0	–	
mok2	91.1±1.1	87±2	87±2	90.4±1.8	91±1.2	90.4±1.8	90±1.1	91.1±1.5	89.9±1.6	–	91.1±1.1	91.8±1.3	–	
pima	75.9±1.2	75.6±1.2	75.6±1.2	76±1.2	75.9±1.2	76±1.2	75.7±1.2	75.7±1.1	75.8±1.2	–	75.9±1.2	75.9±1	–	
survi	74.3±1.3	73.9±1.2	73.9±1.2	74.1±1.3	74.1±1.4	74.1±1.3	74.1±1.2	74.1±1.6	73.9±1.2	–	74.3±1.3	73.8±1.1	–	
vote	95.6±0.5	95.5±0.6	95.5±0.6	95.5±0.6	95.6±0.5	95.5±0.6	95.6±0.5	95.5±0.5	95.5±0.6	–	95.6±0.5	95.6±0.5	–	
vowel	91.4±0.8	85.3±1.1	85.2±1.1	90.6±0.6	88±0.9	88±0.9	89.7±0.7	93.9±0.3	74.9±1	–	91.4±0.8	88.4±0.9	–	
wdbc	96.9±0.5	96.9±0.5	96.9±0.5	97±0.4	96.9±0.5	97±0.4	97±0.4	96.9±0.5	97±0.4	–	96.9±0.5	96.9±0.5	–	

Table C.96: Performance - Combination of *Simple Ensemble* of 40 *MF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	73.8±1.1	73.2±1	73.2±1	73.5±1	74.3±1.1	73.5±1	73.1±1.2	74.4±1.5	74.1±1.1	–	73.8±1.1	74.9±1.3	–	
bala	95.9±0.5	94.1±0.7	94.2±0.7	95.4±0.6	94.7±0.4	95.2±0.6	96.6±0.6	97±0.3	95.2±0.6	–	95.9±0.5	94.3±1.4	–	
band	73.8±1.3	72±1.7	72±1.7	73.6±1.1	73.3±1.5	73.6±1.1	72.7±1.5	71.8±1.5	73.6±1.1	–	73.8±1.3	67±4	–	
bupa	72.7±1.1	71.9±1.5	72±1.5	72.4±1.2	72.6±1.1	72.4±1.2	72.1±1.3	69.6±1.8	72.3±1.3	–	72.7±1.1	66±3	–	
cred	86.5±0.7	86.4±0.7	86.3±0.7	86.3±0.7	86.5±0.7	86.3±0.7	86.2±0.8	85.7±0.9	86.3±0.7	–	86.5±0.7	83±3	–	
derma	97.6±0.7	97.5±0.6	97.5±0.6	97.6±0.7	97.6±0.7	97.6±0.7	96.9±0.7	97.3±0.5	94.8±1.2	–	97.6±0.7	95.6±1.3	–	
ecoli	86.9±0.7	85.3±0.9	85.3±0.8	86.3±0.9	86.9±0.6	86.6±0.7	87.5±0.6	84.9±1.1	83.7±1	–	86.9±0.7	79.6±1.5	–	
flare	81.6±0.5	81.6±0.5	81.6±0.5	81.5±0.5	81.5±0.4	81.5±0.5	81.3±0.7	80.3±0.9	81.7±0.5	–	81.6±0.5	81.3±0.5	–	
glas	94.2±0.6	93.2±0.9	93.4±0.9	94±0.8	94.8±0.7	94.4±0.8	93.8±0.9	95.8±0.9	90.2±0.9	–	94.2±0.6	83.8±1.9	–	
hear	82.9±1.5	82.4±1.4	82.5±1.4	82.5±1.4	83.1±1.5	82.5±1.4	82.9±1.4	81.2±1.3	82.7±1.4	–	82.9±1.5	75±4	–	
img	96.8±0.2	95.8±0.2	95.78±0.19	96.67±0.18	96.41±0.16	96.37±0.17	96.4±0.3	97.2±0.3	95.4±0.4	–	96.8±0.2	95.3±0.6	–	
ionos	90.3±1	89±1.2	89±1.2	90.1±1.2	90.7±1.1	90.1±1.2	91.6±1.1	90.9±1.5	93.4±1.4	–	90.3±1	91±1.3	–	
mok1	98.3±0.9	98.8±0.8	98.8±0.8	98.3±0.9	99.88±0.13	98.3±0.9	99.6±0.4	100±0	100±0	–	98.3±0.9	100±0	–	
mok2	91.1±1.2	86±3	86±3	91±1.6	91.3±1.2	91±1.6	90±1.6	89.3±1.5	90.3±1.5	–	91.1±1.2	91.6±1.2	–	
pima	75.9±1.2	75.6±1.3	75.6±1.3	76±1.2	75.9±1.2	76±1.2	75.6±1.2	75.2±0.9	75.7±1.2	–	75.9±1.2	75.6±1	–	
survi	74.3±1.3	74.1±1.2	74.1±1.2	74.3±1.3	74.3±1.3	74.3±1.3	74.3±1.2	73±0.8	73.9±1.2	–	74.3±1.3	74.6±0.9	–	
vote	95.6±0.5	95.5±0.6	95.5±0.6	95.5±0.6	95.6±0.5	95.5±0.6	95.6±0.5	93.4±0.8	95.5±0.6	–	95.6±0.5	95.8±0.6	–	
vowel	92.2±0.7	84.3±1.2	84.1±1.2	90.5±0.7	88.6±0.9	88.7±0.8	89.5±0.7	94.7±0.4	67.7±1.3	–	92.2±0.7	71.4±1.7	–	
wdbc	96.9±0.5	96.9±0.5	96.9±0.5	97±0.4	96.9±0.5	97±0.4	97±0.4	96.2±0.5	97±0.4	–	96.9±0.5	97±0.4	–	

Table C.97: Performance - Combination of *DECOv1* of 3 *MF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	74.9±1.3	72.9±1.1	72.9±1.1	74.9±1.1	75.3±1.3	74.9±1.1	74.9±1.1	74.4±1.2	74.8±1.2	74.9±1.2	74.9±1.3	74.6±1.2	74.9±1.2	75.1±1.3
bala	96.3±0.5	93.1±0.7	93.1±0.7	94.9±0.5	93.9±0.6	94.6±0.4	95.1±0.9	96.4±0.4	95.9±0.6	95.3±0.9	96.3±0.5	95.5±0.7	95.4±0.9	96.3±0.6
band	73.6±1.1	74±1.2	74±1.2	74.7±1.3	72.9±1.3	74.7±1.3	73.5±1.4	74.2±1.1	74.7±1.2	73.6±1.5	73.6±1.1	74.2±1.3	73.6±1.5	73.8±1.1
bupa	72.6±1.3	70.1±1.1	70.1±1.1	72.1±0.9	72.3±1.4	72.1±0.9	72.3±1.6	71±1.6	72.7±1.3	72.1±1.5	72.6±1.3	70.6±1.5	71.6±1.5	72.6±1.3
cred	86.4±0.7	86.4±0.7	86.4±0.7	86.4±0.7	86.4±0.7	86.4±0.7	86.3±0.7	86.3±0.8	86.4±0.7	86.3±0.8	86.4±0.7	86.2±0.9	86.3±0.8	86.4±0.7
derma	97.2±0.7	96.8±1.1	96.6±1.2	97.5±0.6	97.2±0.7	97±0.9	97±0.7	97.3±0.5	96.9±0.8	97±0.7	97.2±0.7	97.3±0.5	97±0.7	97.2±0.7
ecoli	86.6±0.6	85.2±0.7	85±0.8	86.3±0.7	86.3±0.5	86.6±0.8	86±0.6	86.6±0.7	86.8±0.5	86±0.6	86.6±0.6	86±0.9	86.2±0.8	86.6±0.7
flare	81.7±0.4	81.3±0.5	81.3±0.5	81.4±0.5	81.6±0.5	81.4±0.5	81.7±0.5	81.6±0.5	81.3±0.5	81.7±0.5	81.7±0.4	81.4±0.6	81.5±0.4	81.5±0.4
glas	94.8±0.6	90±1.5	90±1.5	92.2±1.3	94.4±0.8	93.6±0.8	95.8±0.9	96.4±0.8	95±0.5	95.8±0.9	94.8±0.6	96.6±0.7	95.6±0.9	95±0.8
hear	83.2±1.4	83.1±1.5	83.1±1.5	83.1±1.4	83.2±1.4	83.1±1.4	83.2±1.4	83.1±1.3	83.1±1.4	83.2±1.4	83.2±1.4	83.1±1.6	83.4±1.4	83.2±1.4
img	96.7±0.3	95.9±0.3	96±0.3	96.5±0.3	96.2±0.3	96±0.4	96.7±0.2	96.7±0.3	96.7±0.3	96.58±0.19	96.7±0.3	96.5±0.3	96.56±0.19	96.7±0.3
ionos	90.9±0.9	89.9±1.2	89.9±1.2	90.7±1.2	91±0.9	90.7±1.2	91.4±0.9	91.6±0.8	92.3±1	91.1±1	90.9±0.9	91.4±1	91±0.9	91.1±1
mok1	99.4±0.6	97.8±1.2	97.8±1.2	99.4±0.6	100±0	99.4±0.6	100±0	100±0	100±0	100±0	99.4±0.6	100±0	99.3±0.6	99.4±0.6
mok2	88.9±2	84±2	84±2	88.3±1.9	89.3±1.7	88.3±1.9	89.8±1.5	89±1.6	89±2	89.6±1.4	88.9±2	89±1.4	89.3±1.4	89.1±1.8
pima	76.4±1.2	75.2±1	75.2±1	75.8±1.1	76.4±1.2	75.8±1.1	76.1±1.1	76.1±0.9	75.7±1.1	75.9±1.1	76.4±1.2	76.6±1	75.5±1.1	76.4±1.1
survi	74.6±1.5	74.4±1.4	74.4±1.4	74.1±1.5	74.6±1.5	74.1±1.5	74.6±1.5	73.4±1.2	73.8±1.3	74.8±1.3	74.6±1.5	74.1±1.3	74.6±1.3	74.6±1.5
vote	95.8±0.6	95.6±0.6	95.6±0.6	95.9±0.5	95.8±0.6	95.9±0.5	95.8±0.6	95.6±0.7	95.9±0.5	95.8±0.6	95.8±0.6	95.4±0.8	95.9±0.6	95.8±0.6
vowel	91.5±0.6	85.8±0.7	86.3±0.8	89.4±0.5	88.1±0.8	87.8±0.8	91.2±0.7	91.1±0.4	88±0.4	90.1±0.5	91.5±0.6	90.6±0.6	90±0.5	91.7±0.6
wdbc	97±0.5	97.1±0.4	97.1±0.4	97±0.5	96.9±0.4	97±0.5	96.8±0.4	96.9±0.4	97±0.5	96.7±0.4	97±0.5	96.7±0.4	96.7±0.4	97±0.5

Table C.98: Performance - Combination of *DECOv1* of 9 *MF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	76.1±1.1	71.8±1.3	71.8±1.3	75.3±1.3	75.5±1.1	75.3±1.3	73.6±1.1	73.2±1	75.5±1.3	73.7±1.1	76.1±1.1	74.7±1.2	73.7±1.1	76.1±1.1
bala	96.7±0.5	93.8±0.5	93.6±0.6	95.9±0.7	94.1±0.7	94.2±0.8	94.8±1.1	96.6±0.7	96±0.5	96±0.6	96.7±0.5	95.6±0.9	95.9±0.6	96.8±0.4
band	73.8±1.3	73.3±1.5	73.3±1.5	74.9±1.3	73.5±1.3	74.9±1.3	73.1±1.3	72.4±1.2	73.6±1.6	72.9±1.3	73.8±1.3	74.9±1	72.7±1.3	74.2±1.3
bupa	72.7±1	67.1±1.9	67.1±1.9	71.6±1.1	73.1±1	71.6±1.1	73.1±1.4	71.4±1.4	72.1±0.9	72.6±1.5	72.7±1	71.4±1.4	73.1±1.4	72.9±1
cred	86.4±0.7	86.5±0.7	86.5±0.7	86.4±0.7	86.4±0.7	86.4±0.7	86.5±0.7	86.1±0.6	86.4±0.7	86.4±0.7	86.4±0.7	86.2±0.7	86.4±0.7	86.4±0.7
derma	97.6±0.7	97.5±0.6	97.5±0.6	97.6±0.6	97.3±0.6	97.6±0.6	97±0.7	97.2±0.5	95.9±1.2	97±0.7	97.6±0.7	97.8±0.6	97±0.7	97.6±0.7
ecoli	87.2±0.7	85.3±1	84.9±1.1	86.2±0.9	87.1±0.8	86.8±0.7	85.9±0.9	85.3±1.1	85.2±1.2	85.9±1	87.2±0.7	85.4±1	85.9±1	86.9±0.7
flare	81.6±0.6	81.4±0.5	81.4±0.5	81.5±0.6	81.6±0.6	81.5±0.6	81.7±0.5	81.2±0.7	81.4±0.5	81.7±0.5	81.6±0.6	81.5±0.5	81.8±0.5	81.5±0.6
glas	95.4±0.7	92.6±1.1	92.4±1.2	95.2±1	95.6±0.5	95.4±0.6	95.8±0.7	97.2±0.6	91±1.1	96±0.6	95.4±0.7	96.4±0.8	95.4±0.7	95.4±0.7
hear	83.2±1.4	83.9±1.3	83.9±1.3	82.9±1.4	83.2±1.4	82.9±1.4	82.9±1.4	82.9±1.3	82.9±1.4	82.9±1.4	83.2±1.4	82.9±1.2	83.1±1.4	83.2±1.4
img	96.9±0.2	95.7±0.3	95.7±0.3	96.8±0.2	96.1±0.3	96±0.2	96.2±0.3	96.9±0.3	96.2±0.3	96.1±0.3	96.9±0.2	96.58±0.16	96.1±0.3	96.9±0.3
ionos	90.7±1	90±1	90±1	91±1	90.9±1	91±1	91.1±1	91.4±1	93±0.9	91±1	90.7±1	91.4±0.7	91.1±1	90.7±1
mok1	100±0	98.1±1	98.1±1	99.4±0.6	100±0	99.4±0.6	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0
mok2	90.8±1	88±2	88±2	90.6±1.2	90.6±0.8	90.6±1.2	88.9±1	90.3±1.5	91.1±1.4	89.1±1	90.8±1	89.5±1.2	89±0.9	90.8±1.1
pima	76.1±1.1	75±1.2	75±1.2	75.4±1.1	76.2±1.1	75.4±1.1	76.4±1.1	75.9±0.8	75.6±1	76.1±1.1	76.1±1.1	76.3±0.9	75.7±1.3	76±1.1
survi	73.9±1.3	73.6±1	73.6±1	73.6±1.4	73.9±1.3	73.6±1.4	74.1±1.3	73±1.4	73.4±1.1	74.3±1.3	73.9±1.3	74.1±1.1	74.1±1.3	73.9±1.3
vote	95.4±0.7	95.6±0.6	95.6±0.6	95.6±0.6	95.4±0.7	95.6±0.6	95.6±0.5	95.3±0.7	95.5±0.6	95.5±0.6	95.4±0.7	95.5±0.6	95.6±0.7	95.4±0.7
vowel	92.8±0.7	86.1±0.6	86.3±0.6	91.8±0.5	89.1±1	88.9±0.9	90.4±0.7	93.7±0.5	81.5±0.8	88.9±0.6	92.8±0.7	92.5±0.6	88.7±0.6	92.9±0.7
wdbc	97±0.5	97±0.5	97±0.5	97±0.5	97±0.5	97±0.5	96.8±0.4	96.6±0.5	97.1±0.4	96.7±0.4	97±0.5	96.7±0.4	96.7±0.4	97±0.5

Table C.99: Performance - Combination of *DECOv1* of 20 *MF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.5±1.4	72±2	72±2	74.9±1.1	76.6±1	74.9±1.1	74.7±1.7	73.3±1.7	75.8±1.2	–	75.5±1.4	74.8±0.9	–	–
bala	96.6±0.5	93.2±0.7	93±0.7	95.7±0.6	93.6±0.7	94±0.5	94.5±0.9	96.6±0.6	95.2±0.5	–	96.6±0.5	96.2±0.5	–	–
band	73.6±1.4	71.3±1	71.3±1	73.6±1.5	73.8±1.1	73.6±1.5	73.3±1.1	72.4±1.6	73.1±1.5	–	73.6±1.4	72.4±1.6	–	–
bupa	73.1±1.2	64±2	64±2	71±2	72.9±1.4	71±2	72.9±1.2	70.7±1.6	72.9±1.6	–	73.1±1.2	71±2	–	–
cred	86.5±0.7	86.4±0.7	86.4±0.7	86.4±0.7	86.6±0.7	86.4±0.7	86.5±0.7	86.4±0.9	86.4±0.7	–	86.5±0.7	86.1±0.8	–	–
derma	97.6±0.7	97±0.7	97±0.7	97.6±0.7	97.6±0.7	97.6±0.7	97.3±0.8	97±0.6	95.6±1.2	–	97.6±0.7	96.6±1.3	–	–
ecoli	87.1±0.7	84.3±1.2	84.1±1.1	86.8±0.8	87.5±0.8	87.1±0.8	86.8±1	84.9±1.4	82.8±1.3	–	87.1±0.7	82.8±1.5	–	–
flare	81.4±0.5	81±0.5	81±0.5	81.4±0.5	81.4±0.5	81.4±0.5	81.4±0.4	80.8±0.6	81.4±0.5	–	81.4±0.5	81.3±0.6	–	–
glas	95.4±0.7	92.8±1.1	92.8±1.1	94.8±1	96±0.7	95.6±0.7	95.6±0.7	96.2±0.7	87.8±1.1	–	95.4±0.7	91±1.8	–	–
hear	83.1±1.4	82.5±1.1	82.5±1.1	83.1±1.4	82.9±1.4	83.1±1.4	82.9±1.3	82.5±1.2	83.1±1.4	–	83.1±1.4	82.2±1.2	–	–
img	96.9±0.2	95.9±0.4	95.9±0.4	96.9±0.2	96.3±0.2	96.3±0.3	96.3±0.3	97.1±0.3	95.7±0.3	–	96.9±0.2	96.7±0.3	–	–
ionos	91.1±0.9	92±0.7	92±0.7	91.6±0.8	91.4±0.9	91.6±0.8	92.1±1	92±1.1	93.1±1	–	91.1±0.9	91.6±0.6	–	–
mok1	99.5±0.5	98±1.1	98±1.1	98.9±0.8	100±0	98.9±0.8	100±0	100±0	100±0	–	99.5±0.5	100±0	–	–
mok2	91.4±0.9	88.8±1.7	88.8±1.7	91.5±0.9	91.4±0.9	91.5±0.9	88.9±1.2	91.8±1.1	92±1	–	91.4±0.9	90.9±1	–	–
pima	76.1±1	75±0.9	75±0.9	75.4±1	76.1±1	75.4±1	76.3±1.1	75.9±1	75.6±0.9	–	76.1±1	76.3±1	–	–
survi	74.1±1.4	74.6±1.3	74.6±1.3	73.9±1.4	74.1±1.4	73.9±1.4	73.9±1.2	75.1±1.7	73.4±1	–	74.1±1.4	74.1±1.1	–	–
vote	95.5±0.6	95.9±0.7	95.9±0.7	95.5±0.7	95.5±0.6	95.5±0.7	95.6±0.5	95.5±0.6	95.9±0.5	–	95.5±0.6	95.8±0.5	–	–
vowel	93.3±0.6	86.4±0.9	86.3±1	92±0.6	89.9±0.9	89.6±0.7	90.4±0.4	94.1±0.4	73.6±1.1	–	93.3±0.6	87.9±1.4	–	–
wdbc	97±0.5	97.3±0.4	97.3±0.4	97±0.5	97.1±0.5	97±0.5	97.1±0.5	96.8±0.5	97.1±0.4	–	97±0.5	97.1±0.5	–	–

Table C.100: Performance - Combination of *DECOv1* of 40 *MF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.6±1.3	72±1.2	72±1.2	75.1±1.2	76±1.1	75.1±1.2	73.3±1.3	72±1.4	75.6±1.2	–	75.6±1.3	73.1±1	–	–
bala	96.7±0.4	94.8±0.5	94.8±0.5	96.2±0.5	93.5±0.6	94.1±0.5	94±0.8	96.6±0.6	94.7±0.7	–	96.7±0.4	93.5±1.5	–	–
band	73.8±1.3	70.2±1.3	70.2±1.3	73.5±1.4	73.8±1.3	73.5±1.4	74±1.5	71.6±1.7	73.5±1.8	–	73.8±1.3	66.9±1	–	–
bupa	73±1.7	63.1±1.6	63.1±1.6	70±2	73±1.6	70±2	73.3±1.3	71.4±0.9	71±2	–	73±1.7	64±2	–	–
cred	86.5±0.7	86.2±0.7	86.2±0.7	86.4±0.7	86.5±0.7	86.4±0.7	86.5±0.7	85.9±0.8	86.4±0.7	–	86.5±0.7	83±3	–	–
derma	97.6±0.7	97.5±0.8	97.3±0.8	97.6±0.7	97.6±0.7	97.6±0.7	97.3±0.8	96.9±0.7	94.9±1.1	–	97.6±0.7	95.6±1.6	–	–
ecoli	87.5±0.7	83.7±0.7	83.4±0.7	87.2±0.7	87.4±0.7	87.4±0.6	87.1±0.9	81±1.7	82.9±1.1	–	87.5±0.7	77.5±1.6	–	–
flare	81.4±0.5	81.4±0.5	81.4±0.5	81.5±0.6	81.5±0.4	81.5±0.6	81.3±0.5	80.8±0.6	81.4±0.5	–	81.4±0.5	81.3±0.5	–	–
glas	95.2±0.7	94.4±0.8	94.4±0.8	95±1	95.6±0.7	95.4±0.7	95.8±0.8	95.8±0.9	86.2±1.2	–	95.2±0.7	82.8±1.7	–	–
hear	83.1±1.4	82.2±1	82.2±1	83.1±1.4	82.9±1.4	83.1±1.4	83.1±1.2	81.5±1.3	83.2±1.4	–	83.1±1.4	81±2	–	–
img	96.9±0.2	96±0.3	96±0.3	96.92±0.18	96.34±0.18	96.45±0.18	96.1±0.3	97.4±0.3	95.4±0.4	–	96.9±0.2	96.3±0.4	–	–
ionos	91±1	90.4±1	90.4±1	91.3±0.8	91.3±1	91.3±0.8	91.9±1	91±1.3	94.1±1	–	91±1	91±0.9	–	–
mok1	99.4±0.6	98.6±0.9	98.6±0.9	98.6±0.9	100±0	98.6±0.9	99.6±0.4	100±0	100±0	–	99.4±0.6	100±0	–	–
mok2	91.8±0.7	86±3	86±3	91.6±0.9	92.5±0.8	91.6±0.9	89.1±1.3	91.1±1.7	92±1	–	91.8±0.7	92±0.8	–	–
pima	75.9±1	75±1.1	75±1.1	75.3±1	75.9±1	75.3±1	76.3±1.1	75.3±1.2	75.1±0.9	–	75.9±1	71±3	–	–
survi	73.9±1.4	74.1±1.5	74.1±1.5	73.9±1.4	73.9±1.4	73.9±1.4	74.3±1.4	72.1±1.7	73.4±1.1	–	73.9±1.4	75.1±0.5	–	–
vote	95.6±0.6	95.9±0.7	95.9±0.7	95.5±0.7	95.6±0.6	95.5±0.7	95.6±0.5	95.4±0.4	95.8±0.5	–	95.6±0.6	95.8±0.5	–	–
vowel	93.1±0.6	85.7±1.3	85.7±1.3	92.4±0.6	90±0.9	90±0.8	90.1±0.5	94.8±0.7	66±1.2	–	93.1±0.6	73.1±1.7	–	–
wdbc	97±0.5	97.1±0.5	97.1±0.5	97±0.5	97.1±0.5	97±0.5	97.1±0.5	96.4±0.4	97.1±0.4	–	97±0.5	97.1±0.5	–	–

Table C.101: Performance - Combination of *CVCv2* of 3 *MF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	76.1±1.6	74.5±1.9	74.5±1.9	76±1.6	75.9±1.6	76±1.6	76.1±1.6	76±1.7	76.2±1.5	76.2±1.6	75.5±1.8	75.6±1.8	76.2±1.6	76±1.6
bala	95.9±0.9	94.6±1.1	94.6±1.1	95.2±0.8	94.7±0.8	95.4±0.8	95.9±0.9	95.9±0.8	95.8±0.8	96.1±0.9	95.4±0.9	95.6±0.7	96.1±0.9	96±0.9
band	74.2±1.4	72±2	72±2	74.4±1.5	74.2±1.4	74.4±1.5	73.3±1.3	74.7±1.4	73.8±1.2	73.3±1.4	73.8±1.4	74.6±1.2	73.5±1.7	74.2±1.4
bupa	72.6±1.3	69±2	69±2	72.6±1.6	72.7±1.3	72.6±1.6	72.9±1.3	72.7±1.5	73.9±1.3	72.3±1.4	72.6±1.5	70.7±1.6	71±2	72.4±1.5
cred	86.9±0.5	86.8±0.8	86.8±0.8	87.1±0.6	86.9±0.5	87.1±0.6	86.7±0.5	86.9±0.6	87.1±0.6	86.7±0.5	86.9±0.5	86.7±0.8	86.5±0.6	86.6±0.5
derma	98±0.3	96.9±0.6	96.9±0.6	97.9±0.3	97.6±0.6	97.9±0.3	97.8±0.4	97.5±0.4	97.3±0.7	97.5±0.3	97.3±0.4	97.5±0.6	97.5±0.3	98±0.3
ecoli	86.8±0.9	85.7±1.4	85.6±1.4	86.8±0.9	86.9±0.9	86.8±0.9	86.6±1.2	86.6±1	85.6±1.2	86.8±1.1	86.6±0.9	86.9±0.9	86.6±1.2	86.8±0.9
flare	82.5±0.6	82±0.6	82±0.6	81.7±0.8	82.6±0.6	81.7±0.8	82.6±0.6	82.7±0.5	82.1±0.6	82.6±0.7	82.4±0.7	82.4±0.7	82.7±0.7	82.6±0.6
glas	94±1.1	92.8±1.1	92.2±1.1	91.6±0.9	94.6±1	93.4±0.7	93.8±1.1	95.4±1	91.6±1.4	94.6±1	93.6±0.9	94.2±1	94.6±0.7	93.4±1
hear	84.2±1.5	84.4±1.5	84.4±1.5	84.6±1.4	84.2±1.5	84.6±1.4	84.2±1.4	84.4±1.4	84.6±1.4	83.7±1.5	84.4±1.3	84.1±1.3	84.1±1.7	84.6±1.4
img	96.9±0.3	95.7±0.3	95.7±0.3	96.5±0.3	96.3±0.3	96.1±0.3	96.8±0.3	96.9±0.2	96.6±0.2	96.7±0.3	96.8±0.3	96.6±0.3	96.7±0.3	96.8±0.3
ionos	89.7±1.4	89.6±1.5	89.6±1.5	90.3±1.3	90.7±1.3	90.3±1.3	90.9±1.3	89.9±1.5	92±1.2	89.4±1.2	90±1.2	91.4±1.3	89.4±1.3	90±1.5
mok1	98.5±1	97.5±1.1	97.5±1.1	98.5±1	99.1±0.8	98.5±1	100±0	100±0	100±0	99.88±0.13	98.8±0.8	100±0	99.6±0.3	98.5±1
mok2	88.5±1.2	84.9±2	84.9±2	87.3±1.4	89.5±1	87.3±1.4	89±1.1	90.9±0.7	88.4±1.2	89.1±1.1	87.4±1.7	89.9±1.1	89.5±1.1	88.8±1
pima	76.8±1	76.1±1	76.1±1	77.2±0.8	76.7±1.1	77.2±0.8	76.8±1.1	76.6±1	77.2±0.8	77.2±1.1	76.9±0.9	76.5±1.1	76.3±0.9	76.8±1
survi	74.1±1.2	75.1±1.2	75.1±1.2	75.1±1.5	73.9±1.2	75.1±1.5	73.9±1.2	74.3±1.4	73.9±1.3	73.9±1.1	74.1±1.4	74.4±1.2	73.9±0.9	74.3±1.3
vote	96.3±0.5	96.1±0.6	96.1±0.6	96.1±0.6	96.3±0.5	96.1±0.6	96.5±0.5	95.6±0.6	96.1±0.6	96.5±0.5	95.8±0.4	96.1±0.6	96.5±0.5	96.3±0.5
vowel	89.8±0.9	83.9±0.9	83.7±0.9	88.8±1	86.1±0.8	86±0.8	87.7±0.9	89.7±0.9	88.9±0.8	87±1	89.3±0.8	88.4±0.9	87±1	89.7±1
wdbc	96.7±0.3	96.3±0.3	96.3±0.3	96.7±0.3	96.7±0.3	96.7±0.3	97.1±0.3	96.2±0.3	96.5±0.3	97.09±0.18	96.6±0.3	96.4±0.4	97.09±0.18	96.7±0.3

Table C.102: Performance - Combination of *CVCv2* of 9 *MF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.4±1.5	72.1±1.5	72.1±1.5	74.7±1.4	74.9±1.5	74.7±1.4	73.1±1.5	72.8±1.6	74.3±1.4	72.5±1.7	75.2±1.5	72.8±1.6	72.5±1.7	75.5±1.5
bala	96.3±0.7	95±0.9	95.1±0.8	95.8±0.8	94.5±0.8	94.7±0.8	96.6±0.5	97.5±0.5	96.2±0.5	96.6±0.5	96.5±0.7	96.6±0.8	96.6±0.5	96.3±0.7
band	74.6±1.2	72.2±1.7	72.2±1.7	74.6±0.9	74.7±1.2	74.6±0.9	74.4±1.5	72.4±0.9	73.5±1.1	74.7±1.6	74±1	72.4±1.3	74.7±1.5	74.7±1.2
bupa	73.1±1.4	69±2	69±2	73.6±1.4	73.4±1.4	73.6±1.4	73±1.2	71.9±1.5	73±1.5	72.9±1.4	72.9±1.4	72.3±1.4	71.7±1.5	73.3±1.5
cred	86.8±0.7	85.9±0.9	85.9±0.9	86.2±0.8	86.9±0.8	86.2±0.8	85.8±0.9	84.9±1.2	86.5±0.7	85.8±0.9	86.7±0.6	84.7±1.2	85.7±0.9	86.7±0.7
derma	97.3±0.5	97.2±0.6	97.2±0.5	97.3±0.5	97.3±0.5	97.3±0.5	97.9±0.6	98.2±0.5	95.4±0.8	97.9±0.6	97.2±0.5	97.8±0.6	97.9±0.6	97.3±0.5
ecoli	86.6±1	86±1.5	86.2±1.3	86.5±0.9	86.6±1	86.6±0.9	86.5±1.1	86.2±1	85±1	86.2±1	86.6±1	86.2±0.9	86±1	86.5±1
flare	82.1±0.5	81.5±0.6	81.5±0.6	82±0.5	81.9±0.5	82±0.5	81.4±0.6	80.7±0.9	81.4±0.5	81.5±0.6	81.9±0.5	81.3±0.7	81.4±0.6	82.1±0.4
glas	95±1.1	92.8±1.2	92.8±1.2	95±0.9	95.2±0.9	94.8±1	94.8±0.7	96.8±0.5	90.8±1.2	94.4±0.9	94.8±1.1	95.4±1	93.8±1	95±1.1
hear	83.9±1.3	81.9±1.8	81.9±1.8	83.7±1.6	83.9±1.3	83.7±1.6	83.1±1.5	80.7±1.6	83.6±1.6	83.7±1.3	83.9±1.3	81.4±1.7	83.4±1.4	83.7±1.3
img	97±0.3	95.9±0.3	96±0.3	96.9±0.3	96.8±0.3	96.7±0.3	96.9±0.3	97.35±0.16	96.5±0.3	96.9±0.3	97±0.3	97.1±0.2	97±0.3	97±0.3
ionos	90.4±1.3	89±1.6	89±1.6	90.3±1.3	91.1±1.2	90.3±1.3	90.7±1.4	91.7±1.1	93±0.9	90.3±1.6	89.3±1.3	91.1±1.1	90.3±1.5	90.6±1.3
mok1	99.8±0.3	99.5±0.5	99.5±0.5	99.6±0.4	100±0	99.6±0.4	100±0	100±0	100±0	100±0	99.8±0.3	100±0	100±0	99.88±0.13
mok2	94±2	90±2	90±2	94.1±1.7	94±2	94.1±1.7	92.4±2	92.5±1.4	92.3±1.8	91.6±1.8	93.9±2	91.8±1.3	91.3±1.6	94±2
pima	76.8±1.1	75.7±1.5	75.7±1.5	76.8±1.1	76.2±1.2	76.8±1.1	75.2±1.1	74±1	77±1	75.2±1.1	77±1.2	74.4±1	75±1.1	76.6±1.1
survi	73±1	74.4±1.1	74.4±1.1	73.4±1.1	73.1±1	73.4±1.1	72.8±0.7	73±1.5	73.4±1.2	73.3±0.7	73.4±1.1	72.8±1.1	72.8±0.9	73±1
vote	96.3±0.5	95.4±0.8	95.4±0.8	96.5±0.5	95.5±0.5	96.5±0.5	95.3±0.6	95±0.7	96±0.6	95.1±0.6	95.9±0.6	95.1±0.6	95.1±0.6	96.3±0.5
vowel	92.7±0.7	85.6±0.8	85.3±0.7	91.3±0.5	88.5±0.9	88.7±0.5	90.7±0.7	93.6±0.5	82.4±1.1	89.9±0.8	92.4±0.7	92.7±0.7	90±0.7	92.7±0.7
wdbc	96.8±0.3	96.6±0.5	96.6±0.5	96.7±0.3	96.7±0.3	96.7±0.3	96.5±0.4	95.6±0.4	95.8±0.4	96.7±0.4	96.6±0.4	95.8±0.4	96.6±0.5	96.8±0.3

Table C.103: Performance - Combination of *CVCv2* of 20 *MF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	74.3±1.2	75.4±1.4	75.3±1.5	74.4±0.9	73.7±1.2	74.4±0.9	72.8±1.2	71.5±0.9	73.9±1.5	–	73.5±1.3	72.2±0.9	–	–
bala	96.3±0.5	94.3±1.6	94.2±1.8	95.8±0.5	94.6±0.8	95.2±0.6	97.2±0.5	97.5±0.3	95.8±0.5	–	96.5±0.4	97.2±0.4	–	–
band	75.3±1.7	71.1±1.8	71.1±1.8	75.8±1.2	75.6±1.4	75.8±1.2	73.3±1.1	74±1.2	73.5±1.7	–	75.3±1.6	73.8±1.2	–	–
bupa	72.7±1.3	71±4	71±4	72.6±1.6	72.7±1.3	72.6±1.6	72.9±1.5	70.6±1.2	72.7±1.7	–	72.7±1.4	71.9±1.3	–	–
cred	86±0.8	86.5±0.6	86.5±0.6	86.4±0.7	86.2±0.8	86.4±0.7	85.7±0.9	83.3±1.2	86.2±0.8	–	86.1±0.8	84.2±1.2	–	–
derma	97.5±0.6	96.1±1.1	96.1±1.1	97.3±0.6	97.6±0.6	97.3±0.6	97±0.7	97.2±0.6	94.7±0.9	–	97.5±0.6	96.8±0.7	–	–
ecoli	86.6±1.1	86.6±1	86±1.2	86.5±1.2	86.6±1	86.5±1.2	86.2±1	86±1	84±0.8	–	86.6±1.1	77.7±1.6	–	–
flare	81.8±0.4	81.6±0.6	81.6±0.6	81.8±0.5	81.8±0.4	81.8±0.5	81.1±0.5	80.1±0.8	81.5±0.6	–	81.7±0.5	81±0.5	–	–
glas	95.4±0.9	93.8±1.1	93.2±1.1	94.8±1	95.2±0.7	95.4±0.9	95.2±0.7	96.6±0.9	88±1.1	–	95.6±0.8	88.6±1.5	–	–
hear	84.1±1.2	83.1±1	83.1±1	83.9±1.1	83.6±1.1	83.9±1.1	82.2±1.1	80.2±1.4	83.9±1.1	–	83.4±1.3	81.5±1.1	–	–
img	97.03±0.17	96.22±0.16	96.22±0.16	96.97±0.16	96.62±0.15	96.75±0.16	96.8±0.2	97.3±0.3	96.1±0.3	–	97.01±0.19	97.1±0.3	–	–
ionos	91±0.9	89.4±1.3	89.4±1.3	91±1	91.4±1	91±1	92±1.6	92.6±0.9	94.1±0.9	–	92±0.9	92.3±1.2	–	–
mok1	100±0	96.9±1.4	96.9±1.4	100±0	100±0	100±0	100±0	100±0	100±0	–	100±0	100±0	–	–
mok2	94.9±1.5	91.1±1.2	91.1±1.2	94.4±1.7	94.6±1.6	94.4±1.7	93.4±1.4	91.6±1.2	88.5±1.3	–	95.3±1.4	93.4±1.3	–	–
pima	76.7±0.8	76.1±0.9	76.1±0.9	77.2±0.8	76.7±0.8	77.2±0.8	74.5±1	72.7±0.9	77.4±0.7	–	76.9±0.9	72.8±0.7	–	–
survi	73.6±1	73.8±1	73.8±1	74.3±1.3	73.3±1	74.3±1.3	71.6±0.6	69.2±1.4	73.6±1.1	–	73.6±1.1	73±0.7	–	–
vote	95.6±0.4	95.8±0.8	95.8±0.8	96.3±0.5	95.3±0.5	96.3±0.5	95±0.5	93.8±0.9	95.9±0.5	–	95.5±0.4	94.4±0.7	–	–
vowel	93.3±0.6	87.4±0.8	87.3±0.8	91.6±0.6	89.8±0.7	89.6±0.6	90.7±0.9	94.8±0.6	73.2±0.7	–	93.3±0.6	89.4±1.1	–	–
wdbc	95.9±0.6	96.4±0.5	96.4±0.5	96.3±0.5	95.8±0.6	96.3±0.5	96±0.5	95.9±0.6	96.2±0.4	–	96±0.6	96±0.6	–	–

Table C.104: Performance - Combination of *CVCv2* of 40 *MF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	77±0.8	72±2	72±2	77±1.3	76.7±1.1	77±1.3	74.4±1.9	72.2±1.7	73.8±1.8	–	77.1±0.8	73.8±1.6	–	–
bala	95.8±0.7	95.1±0.9	95.2±0.9	95.4±0.9	94.6±0.8	94.9±0.9	96.6±0.8	97±0.5	95±0.7	–	95.8±0.7	94.6±1.4	–	–
band	74.9±1.3	70.7±1.6	70.7±1.6	74.4±1.2	74.6±1.2	74.4±1.2	72.4±1.1	72±1.4	71.1±1.1	–	74.7±1.4	72.9±1	–	–
bupa	72.3±1.5	68.1±1.8	68.1±1.8	72±1.8	72.9±1.5	72±1.8	72.6±1.3	69.1±1.5	72.1±1.5	–	72.3±1.4	69±0.8	–	–
cred	86±0.8	86.4±0.7	86.4±0.7	86.5±0.7	86.8±0.8	86.5±0.7	85.8±1	83.5±0.9	86.3±0.9	–	85.9±0.9	84.9±1.2	–	–
derma	97.2±0.6	96.9±0.6	96.9±0.6	97.3±0.6	97.2±0.6	97.3±0.6	97.2±0.6	97.2±0.6	93.4±0.7	–	97±0.6	90±1.2	–	–
ecoli	86.3±0.9	85.4±1.2	85.9±1.1	86.2±1	86.3±0.9	85.7±1.1	85.7±1	85.2±1.1	84.7±1.3	–	86.3±0.9	74±2	–	–
flare	82.2±0.5	81.3±0.6	81.3±0.6	82.3±0.5	82±0.4	82.3±0.5	80.9±0.7	80±0.7	81.6±0.6	–	82.2±0.4	81.1±0.4	–	–
glas	95.4±0.9	94.2±1	94±0.8	94.4±0.9	95.2±1	95.2±1	95±0.8	95.6±0.9	87.2±1.4	–	95.4±0.9	81.2±1.6	–	–
hear	82.7±1.2	82.7±1	82.7±1	83.4±1.4	82±1.1	83.4±1.4	80.9±1.5	77.8±2	82.4±1.4	–	82.7±1.4	81.2±1	–	–
img	96.7±0.3	96.3±0.4	96.3±0.4	96.6±0.3	96.6±0.3	96.3±0.3	96.8±0.2	97.2±0.3	95.7±0.4	–	96.8±0.3	94.6±0.6	–	–
ionos	92±1	91±1.3	91±1.3	92±0.9	92.3±1.1	92±0.9	91.9±1.3	90.4±0.9	93.4±1.1	–	92±1	92.1±1.1	–	–
mok1	100±0	94±4	94±4	99.88±0.13	100±0	99.88±0.13	100±0	100±0	100±0	–	100±0	100±0	–	–
mok2	93.5±1.7	86±4	86±4	93±1.9	93.6±1.6	93±1.9	91±1.4	89.3±1	80.9±1.3	–	93.9±1.5	94.1±1.4	–	–
pima	76.1±0.9	73.5±1.3	73.5±1.3	75.9±1.2	74.8±1	75.9±1.2	73.3±1.1	72.8±1.2	75.9±1.3	–	75.8±0.9	72.4±1	–	–
survi	73.4±1.2	73.1±1.6	73.1±1.6	73.1±1.3	73.4±1	73.1±1.3	71.5±1.3	68.2±1.4	73.3±1	–	73.6±1.1	74.1±0.8	–	–
vote	95±0.7	94.5±0.8	94.5±0.8	95.3±0.5	94.9±0.7	95.3±0.5	95±0.5	93.5±0.9	95±0.7	–	94.9±0.8	94.6±0.8	–	–
vowel	92.9±0.7	85.3±1.3	85.6±1.3	91.2±0.8	88.8±0.9	88.5±0.9	90.6±0.8	95.5±0.5	66.4±1.2	–	92.9±0.8	78±2	–	–
wdbc	96±0.5	96±0.5	96±0.5	95.7±0.6	96±0.5	95.7±0.6	96.1±0.4	95.4±0.5	95.7±0.5	–	96±0.5	95.5±0.5	–	–

Table C.105: Performance - Combination of *Conservative Boosting* of 3 *MF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	74.9±1.1	69.8±1.6	69.8±1.6	74±1.4	74.6±1.1	74±1.4	74.4±1.1	73.6±1.4	74.1±1.4	74.6±1	74.9±1.1	74.4±1.4	74.6±1	74.9±1.3
bala	95.8±0.7	93±1	93.1±1	94.5±0.8	94.1±0.9	94.2±0.9	95±0.7	95.7±0.6	94.4±1	94.6±0.7	95.8±0.7	94.8±0.9	94.5±0.7	95.8±0.7
band	73.1±1.5	69±2	69.3±2	73.1±1.5	72.9±1.5	73.1±1.5	72.7±1.7	72.2±1.5	71.8±1.6	72.4±1.7	73.1±1.5	71.3±1.6	72.6±1.7	72.9±1.5
bupa	71.4±1.6	63.6±1.7	63.6±1.7	70.7±1.7	70.9±1.5	70.7±1.7	72.1±1.4	71.3±1.6	70.4±1.6	72±1.6	71.4±1.6	72±1.5	71±1.5	71.6±1.6
cred	86.3±0.6	82.5±1.4	82.5±1.4	86.3±0.7	86.2±0.7	86.3±0.7	85.9±0.7	86.2±0.8	86.3±0.7	85.8±0.7	86.3±0.6	85.9±0.7	85.6±0.8	86.2±0.7
derma	97.3±0.7	96.5±0.6	96.3±0.5	96.9±0.8	97.3±0.7	96.8±1	97.2±0.6	97.5±0.8	95.8±0.9	97.2±0.7	97.3±0.7	97.5±0.6	96.9±0.7	97.2±0.7
ecoli	86.5±1.1	82.7±0.9	82.1±1	85.2±1.2	85.4±1.4	84.6±1.3	87.5±1	87.2±1.1	84.1±1.3	87.2±0.9	86.5±1.1	86.5±0.8	86.9±1	86.2±1
flare	82.3±0.8	76±2	76±2	82.1±1	82.4±0.8	82.1±1	81.8±0.8	82.5±0.5	81.4±0.4	81.8±0.8	82.3±0.8	82.2±0.5	81.5±0.7	82.4±0.7
glas	95.8±1	92±1.1	92±1	94.4±0.8	95±1.2	94.2±1.2	93.2±0.8	93.6±1.4	92.8±1.2	93±0.8	95.8±1	95.6±1.1	92.6±0.9	95.6±1.1
hear	84.1±1.3	75.6±1.4	75.6±1.4	83.7±1.5	83.9±1.3	83.7±1.5	83.9±1.3	81.9±1	83.7±1.5	83.9±1.3	84.1±1.3	82.7±1.4	83.2±1.4	83.9±1.3
img	97±0.2	95±0.3	95±0.3	96.4±0.3	96±0.3	95.7±0.3	96.7±0.3	96.9±0.3	96.4±0.3	96.5±0.3	97±0.2	96.4±0.3	96.3±0.3	97±0.2
ionos	90±0.8	88.6±1	88.6±1	89.4±1	90.9±0.9	89.4±1	90.6±1.1	89.7±1	89.7±1	89.3±1	90±0.8	90.6±1	89.1±1	90±0.8
mok1	100±0	98.4±1.3	98.4±1.3	100±0	100±0	100±0	99.88±0.13	100±0	100±0	99.75±0.17	100±0	100±0	99.5±0.3	100±0
mok2	79.6±1.9	72.5±1.6	72.5±1.6	77±1.8	80±2	77±1.8	79±2	76±2	76±2	79±2	79.6±1.9	77±2	79±2	79.5±1.8
pima	76.6±0.9	73.2±1.4	73.2±1.4	76.1±1.1	76.4±1	76.1±1.1	76.2±0.8	76.5±1.1	75.9±1.2	76±0.9	76.6±0.9	76.7±0.9	75.6±0.9	76.7±0.9
survi	75.4±0.9	73.3±1.2	73.3±1.2	75.6±1	75.1±0.8	75.6±1	74.9±0.7	74.1±1.4	74.6±0.7	75.4±0.9	75.4±0.9	75.6±1.2	75.3±0.7	74.8±1
vote	96.1±0.7	93.1±0.8	93.1±0.8	96±0.6	95.9±0.7	96±0.6	95.6±0.8	96±0.7	96±0.6	95.5±0.8	96.1±0.7	96.3±0.7	95.5±0.8	96.1±0.7
vowel	90.3±1.3	80.6±2	80.6±1.9	87.8±1.3	83±1.6	82.8±1.5	87±1	90.4±1.2	88.4±1.1	85.8±1.3	90.3±1.3	87.8±1.3	85.6±1.4	90.2±1.4
wdbc	96.6±0.5	94.3±0.6	94.3±0.6	97±0.6	96.3±0.4	97±0.6	96±0.5	96.5±0.5	97±0.6	95.7±0.6	96.6±0.5	96.6±0.4	95.7±0.6	96.6±0.5

Table C.106: Performance - Combination of *Conservative Boosting* of 9 *MF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.8±0.8	57±4	57±4	75.3±1	74.8±0.7	75.3±1	72.3±0.8	74.3±1.1	75.2±0.9	72.1±1	75.8±0.8	72.4±0.8	72.1±1	75.8±0.9
bala	95.8±0.6	93.7±0.6	94.1±0.4	95.5±0.6	95±0.5	95.6±0.6	94.9±0.8	95.4±0.7	94.8±0.7	94.8±0.9	95.8±0.6	94.7±0.7	95.2±0.8	95.9±0.6
band	73.6±1.4	63±3	63±3	72±1.2	73.6±1.3	72±1.2	70.6±1.1	72±2	71±2	70±1.2	73.6±1.4	69±2	70.4±1.2	74.4±1.4
bupa	72.3±1.4	54±2	54±2	71.9±1.7	72±1.6	71.9±1.7	71±2	70.9±1.7	71.4±1.5	71±2	72.3±1.4	71.7±1.9	71±2	72.9±1.5
cred	87.2±0.9	73±3	73±3	86.5±0.7	86.5±1	86.5±0.7	84.4±1.3	86.2±0.8	86.7±0.7	84.3±1.3	87.2±0.9	84±1	83.9±1.3	87±0.8
derma	97.9±0.5	94.8±1.3	94.9±1.3	97.6±0.7	97.9±0.5	97.6±0.6	96.9±0.6	97.3±0.6	94.7±1	96.6±0.8	97.9±0.5	97.6±0.6	96.8±0.8	97.9±0.5
ecoli	86.8±1.1	72±2	72±2	84.4±1.8	86±1	84.3±1.4	86.8±0.9	87.7±0.9	82.1±1.2	86.6±0.9	86.8±1.1	84.3±1	86.3±1.2	86.2±1.1
flare	80.9±0.8	69±7	69±7	81.3±0.8	80.4±0.9	81.3±0.8	78.5±1.5	81.7±0.4	81.2±0.5	78.5±1.6	80.9±0.8	81.6±0.5	78.4±1.5	81.2±0.6
glas	95.8±0.9	84±4	83±4	95.6±0.8	96.4±0.7	96±0.8	83±4	95.6±1.1	87.4±1	82±5	95.8±0.9	95.4±1	82±4	95.8±0.8
hear	83.7±1.4	69±3	69±3	82.4±1.5	82.7±1.4	82.4±1.5	80.7±1.4	81.7±1.2	82.5±1.7	80.7±1.3	83.7±1.4	82.4±1.1	80±1.3	83.6±1.3
img	97.3±0.3	93.2±0.6	93±0.6	97.3±0.3	96.9±0.3	96.9±0.3	96.11±0.19	97.3±0.3	96.3±0.4	95.2±0.3	97.3±0.3	96.9±0.4	95±0.3	97.3±0.3
ionos	91.9±0.9	88±2	88±2	90.9±0.9	91.3±1	90.9±0.9	89.3±1.2	91.1±1.2	91.3±0.9	88.6±1.2	91.9±0.9	90.4±1.1	88.9±1.2	91.7±0.9
mok1	100±0	100±0	100±0	100±0	100±0	100±0	99.3±0.5	100±0	100±0	99.1±0.5	100±0	100±0	98.9±0.6	100±0
mok2	82.8±1.4	68.8±1.3	68.8±1.3	80.9±1.8	82.6±1.7	80.9±1.8	81±2	81.8±1.6	76.5±1.4	79±2	82.8±1.4	82±2	79±2	82.6±1.6
pima	76.6±0.8	66.1±1.5	66.1±1.5	76.1±1.1	76.5±1	76.1±1.1	75±0.8	76.6±0.7	74.6±0.7	74.7±1	76.6±0.8	76±0.8	74.5±1	76.8±0.8
survi	74.4±1.5	57±6	57±6	73.9±1.7	74.4±1.4	73.9±1.7	73±1.2	74.3±1.5	74.6±0.7	73.1±1.3	74.4±1.5	75.1±1.3	73.6±1.3	74.4±1.4
vote	95.9±0.7	88±3	88±3	95.6±0.7	95.4±0.8	95.6±0.7	94.1±1.1	95.6±0.7	95.5±0.7	94.1±1	95.9±0.7	95.1±0.8	94.1±1	95.9±0.7
vowel	95.1±0.7	74.9±1.1	75±1.2	94.1±0.7	89.1±0.7	89.4±0.5	86.7±0.9	94.6±0.6	86.4±0.8	83.2±1.1	95.1±0.7	91.8±0.7	83.2±1.2	94.9±0.7
wdbc	96.6±0.6	95.8±0.4	95.8±0.4	96.7±0.7	96.6±0.4	96.7±0.7	96.1±0.2	96.3±0.6	96.7±0.6	95.2±0.3	96.6±0.6	96.3±0.4	95.1±0.3	96.6±0.6

Table C.107: Performance - Combination of *Conservative Boosting* of 20 *MF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.3±1	66±3	66±3	74.5±1.4	74.6±0.8	74.5±1.4	70.6±0.8	72.6±1.1	75.8±1	–	75.3±1	70.6±1	–	–
bala	96.1±0.5	89.2±1.3	90.4±1.2	95.9±0.6	96.2±0.4	95.9±0.5	93.9±0.5	96.1±0.6	94.8±0.8	–	96.1±0.5	94.7±0.6	–	–
band	74.2±1.9	59±4	59±4	72±1.9	73.3±1.7	72±1.9	70±1.9	71.1±1.2	67.5±1.9	–	74.2±1.9	68.2±1.5	–	–
bupa	72.7±1.5	59±1.5	59±1.5	69.7±1.5	72.7±1.3	69.7±1.5	71.3±1.5	71.7±1	72±1.7	–	72.7±1.5	71.4±1.8	–	–
cred	86.9±0.5	71±3	71±3	85.7±0.9	85.7±0.8	85.7±0.9	82.1±1	85.2±0.9	86.5±0.6	–	86.9±0.5	82.9±0.8	–	–
derma	97.8±0.6	97.3±0.5	96.8±0.7	97.8±0.6	97.9±0.5	97.8±0.6	96.8±0.8	97.2±0.7	93.4±1.1	–	97.8±0.6	96.9±0.8	–	–
ecoli	87.9±1.1	75±3	73±3	85.3±1.6	87.5±1	86±1.1	86±1.2	87.9±0.8	79.6±1.5	–	87.9±1.1	69±3	–	–
flare	80.3±0.8	64±6	64±6	80.9±0.6	80.2±0.7	80.9±0.6	78.2±1.2	81.4±0.6	81.3±0.5	–	80.3±0.8	81.3±0.5	–	–
glas	96.6±0.8	79±5	79±5	96.2±0.8	96.4±0.8	96.6±0.8	72±6	95.4±1.2	83.4±1.2	–	96.6±0.8	88±2	–	–
hear	82.9±1.7	70±3	70±3	82.4±1.9	82.4±1.4	82.4±1.9	78.1±1.5	81.5±1	83.6±1.5	–	82.9±1.7	78.3±1.4	–	–
img	97.4±0.3	89±2	89±2	97.3±0.3	97.1±0.3	97.1±0.3	93.7±0.6	97.4±0.3	95±0.4	–	97.4±0.3	96.7±0.5	–	–
ionos	92.4±0.9	86±2	86±2	92.4±0.9	92.3±0.9	92.4±0.9	89.9±1.4	91.4±1.2	92.6±1	–	92.4±0.9	89.4±1.3	–	–
mok1	100±0	99±1	99±1	100±0	100±0	100±0	98.5±0.8	100±0	100±0	–	100±0	100±0	–	–
mok2	84.5±1.1	74±3	74±3	81±2	85.5±1.4	81±2	78.5±1.6	85.9±1.5	74.8±1.5	–	84.5±1.1	82.3±1.9	–	–
pima	76.3±0.8	67±3	67±3	75.8±0.9	75.9±1	75.8±0.9	72.9±1.2	75.3±1.1	70.6±1.7	–	76.3±0.8	72.4±1.7	–	–
survi	73.6±1.4	61±4	60±4	72.5±1.4	73.1±1.4	72.5±1.4	72.6±0.8	71.3±1.2	74.6±0.7	–	73.6±1.4	69±5	–	–
vote	95.9±0.7	78±4	78±4	95.5±0.7	95.5±0.7	95.5±0.7	92.1±1.7	95.6±0.6	96±0.6	–	95.9±0.7	95.4±0.8	–	–
vowel	96.7±0.6	72.8±1.4	72.9±1.4	96.8±0.6	93±0.5	94.3±0.5	85.3±1.1	95.9±0.5	79±1.1	–	96.7±0.6	48±3	–	–
wdbc	96.4±0.5	94.5±0.7	94.5±0.7	96.4±0.5	96.6±0.5	96.4±0.5	95.9±0.5	96.2±0.5	96.6±0.5	–	96.4±0.5	96.3±0.4	–	–

Table C.108: Performance - Combination of *Conservative Boosting* of 40 *MF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	74.9±1.2	68±3	68±3	75.3±1.2	73.9±1	75.3±1.2	71.3±1.1	72.3±0.8	75.2±1	–	74.9±1.2	60±4	–	–
bala	96.1±0.7	87±2	88±2	96±0.7	96.2±0.4	96.2±0.7	92.8±0.8	96.2±0.8	94±0.9	–	96.1±0.7	91.8±1.2	–	–
band	75.5±1.5	62±4	62±4	73.6±2	74.9±1.6	73.6±2	70.2±1.2	70.6±1.2	67.5±0.9	–	75.5±1.5	71.8±1.7	–	–
bupa	73±1.3	61±2	61±2	72±1.3	72±1.2	72±1.3	68.9±1.8	69.1±1.1	71.7±1.6	–	73±1.3	60±4	–	–
cred	85.9±0.6	70±2	70±2	86.5±1	85.8±0.8	86.5±1	82.8±1.1	85±0.8	86.6±0.7	–	85.9±0.6	71±5	–	–
derma	97.5±0.7	96.8±0.9	96.5±0.9	97.6±0.7	97.8±0.6	97.6±0.7	96.6±0.8	97.2±0.6	90.9±1.3	–	97.5±0.7	91.6±1.6	–	–
ecoli	87.9±1.1	77±3	76±3	85.6±1.6	87.5±0.7	86.5±1	84.4±1.6	85.2±0.6	77.4±1.4	–	87.9±1.1	83.1±1.7	–	–
flare	80.7±0.6	61±7	61±7	81.6±0.8	80.2±0.7	81.6±0.8	77.6±1.4	80.9±0.7	81.3±0.5	–	80.7±0.6	81.3±0.5	–	–
glas	97±0.6	84±3	84±3	97.2±0.6	97±0.7	96.8±0.7	59±3	96.6±1	79.8±1.7	–	97±0.6	86±6	–	–
hear	82.4±1.8	74±4	74±4	82.7±1.6	81.2±1.4	82.7±1.6	77.1±1.9	79.8±1.1	83.6±1.6	–	82.4±1.8	62±5	–	–
img	97.4±0.2	83±3	83±3	97.3±0.2	97.2±0.3	97.2±0.3	81.2±1.9	97.5±0.3	93.6±0.4	–	97.4±0.2	73±4	–	–
ionos	92±0.9	86.9±1.7	86.9±1.7	92.1±0.8	91.9±0.9	92.1±0.8	87.1±2	92.1±1.2	92±0.9	–	92±0.9	90.7±1	–	–
mok1	100±0	95.5±2	95.5±2	100±0	100±0	100±0	97±1.4	100±0	100±0	–	100±0	100±0	–	–
mok2	87.5±1.2	71±5	71±5	83.5±1.6	87.6±1.2	83.5±1.6	81±3	88.4±1.1	72.9±1.7	–	87.5±1.2	69±2	–	–
pima	77±0.8	70±2	70±2	76.6±0.8	76.5±0.8	76.6±0.8	73±1.4	74.3±1	70.7±1.2	–	77±0.8	65±0.7	–	–
survi	74.6±1.6	71±3	71±3	73.6±1.5	74.1±1.6	73.6±1.5	70±2	70.5±1.5	74.6±0.7	–	74.6±1.6	72.5±1.7	–	–
vote	95.4±0.7	79±6	79±6	95.6±0.7	94.9±0.7	95.6±0.7	90±2	95.6±0.6	95.9±0.8	–	95.4±0.7	80±6	–	–
vowel	97.4±0.5	74.2±0.8	74.1±0.9	97.4±0.6	95.4±0.8	95.7±0.8	81.9±1	97.2±0.6	69.7±0.9	–	97.4±0.5	90.4±1.2	–	–
wdbc	96.1±0.5	94.8±0.8	94.8±0.8	96.4±0.5	96.4±0.6	96.4±0.5	96.2±0.5	95.9±0.6	96±0.5	–	96.1±0.5	96.4±0.6	–	–

Table C.109: Performance - Combination of *Simple Ensemble* of 3 *RBF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.2±1	74.9±0.9	74.9±0.8	75.2±0.9	75.2±1	75.2±0.9	75.1±1	73.8±0.8	74.8±0.8	75.4±0.8	75.2±1	68±3	75.4±0.8	75.1±1
bala	89.6±0.7	89.6±0.7	89.6±0.7	89.6±0.7	89.6±0.7	89.6±0.7	89.8±0.7	89.8±0.7	89.7±0.7	89.9±0.7	89.6±0.7	70±5	89.8±0.7	89.6±0.7
band	72.9±1.5	71.6±1.8	71.5±1.7	71.6±1.4	72.7±1.5	71.6±1.4	72.7±1.5	71.1±1.2	71.6±1.4	72.6±1.6	72.9±1.5	70±2	72.7±1.8	72.9±1.5
bupa	71.7±1.3	72.6±1.4	72.4±1.4	72.3±1.3	71.7±1.3	72.3±1.3	72±1.2	71.3±1	72.3±1.3	72.4±1.2	71.7±1.3	65.1±1.9	72±1.4	71.9±1.2
cred	87±0.5	87±0.5	87.1±0.6	87.1±0.5	87±0.5	87.1±0.5	87.1±0.5	86.7±0.6	87.1±0.5	87.2±0.5	87±0.5	74±5	87.2±0.6	87±0.5
derma	96.8±0.5	97±0.4	97±0.4	96.9±0.5	96.8±0.5	96.9±0.5	97.2±0.5	96.9±0.5	96.8±0.5	97.2±0.5	96.8±0.5	82±5	97.2±0.5	96.8±0.5
ecoli	88.2±0.9	87.8±0.8	88.1±0.9	88.1±0.9	88.1±0.9	87.9±1	88.2±1	87.5±1.2	87.5±0.9	88.1±1	88.2±0.9	80.3±1.7	87.9±1	88.2±0.9
flare	81.6±0.6	81.9±0.5	82±0.5	81.6±0.5	81.6±0.6	81.6±0.5	81.8±0.6	82±0.8	81.9±0.5	81.8±0.6	81.6±0.6	74±5	81.8±0.6	81.6±0.6
glas	93.2±1	93.4±0.9	93.4±0.9	93.2±1	93.2±1	93.2±1	93.2±1	93.2±1	92.6±1	93.2±1	93.2±1	79±5	93.2±1	93.2±1
hear	83.6±1.7	83.4±1.7	83.4±1.7	83.6±1.8	83.6±1.7	83.6±1.8	82.5±2	83.7±1.7	83.6±1.8	82.7±1.9	83.6±1.7	72±5	82.4±1.8	83.6±1.7
img	96.9±0.3	96.9±0.3	96.9±0.3	97±0.3	96.9±0.3	97±0.3	97±0.3	96.9±0.4	96.9±0.3	96.9±0.3	96.9±0.3	82±5	96.9±0.3	96.9±0.3
ionos	90.7±1.1	90.7±1.1	90.4±1.1	90.3±1	90.7±1.1	90.3±1	90.7±1.1	90.6±1	90.7±1.1	90.9±1	90.7±1.1	75±7	90.7±1	90.7±1.1
mok1	99.6±0.3	99.75±0.17	99.8±0.3	99.8±0.3	99.6±0.3	99.8±0.3	99.6±0.3	100±0	100±0	99.6±0.3	99.6±0.3	91±3	99.6±0.3	99.6±0.3
mok2	90.3±1.3	89.5±0.8	89.3±1	91±0.9	90.4±1.2	91±0.9	90.9±1.5	91.4±1.3	91±0.9	91±1.4	90.3±1.3	77±6	91±1.3	90.4±1.2
pima	77.3±0.9	77.4±0.8	77.2±0.8	77±0.9	77.3±0.9	77±0.9	77.3±0.8	76.8±1	77.2±0.9	77.1±0.9	77.3±0.9	69±3	77.1±1	77.3±0.9
survi	75.6±1.5	75.3±1.4	75.1±1.5	75.4±1.6	75.4±1.5	75.4±1.6	76.1±1.7	76.2±1.8	75.4±1.6	75.9±1.8	75.6±1.5	68±6	76.1±1.8	75.6±1.5
vote	96.3±0.6	95.9±0.6	95.9±0.6	96.1±0.6	96.3±0.6	96.1±0.6	95.9±0.6	95.9±0.5	96.1±0.6	95.9±0.6	96.3±0.6	91±4	95.9±0.6	96.3±0.6
vowel	97.2±0.4	97.1±0.4	97±0.4	97.2±0.4	97.2±0.4	97.2±0.4	97.4±0.4	97.7±0.3	96.9±0.4	97.4±0.3	97.2±0.4	87.3±0.9	97.4±0.3	97.2±0.4
wdbc	97.2±0.3	97.1±0.4	97±0.4	97.2±0.3	97.2±0.3	97.2±0.3	97.3±0.3	97.1±0.3	97.1±0.3	97.3±0.3	97.2±0.3	85±6	97.3±0.3	97.2±0.3

Table C.110: Performance - Combination of *Simple Ensemble* of 9 *RBF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.4±0.9	75.1±1	74.9±0.9	75.4±0.8	75.4±0.9	75.4±0.8	75.4±1	74±0.8	75.2±0.9	75.1±1	75.4±0.9	67±2	75.1±1	75.4±0.9
bala	89.7±0.7	89.5±0.7	89.6±0.7	89.7±0.7	89.7±0.7	89.6±0.7	89.6±0.7	89.8±0.8	89.4±0.7	89.7±0.7	89.7±0.7	73±6	89.8±0.7	89.7±0.7
band	73.5±1.6	73.8±1.4	73.8±1.6	73.3±1.7	73.1±1.6	73.3±1.7	74.4±1.5	71.3±1	73.5±1.4	74.4±1.5	73.5±1.6	70.7±1.6	74.4±1.5	73.3±1.6
bupa	71.9±1.3	71.6±1.1	71.7±1.1	72±1.3	72±1.3	72±1.3	71.6±1.2	71±1.4	71.9±1.2	71.4±1.1	71.9±1.3	61±3	70.9±1.6	71.9±1.3
cred	87.2±0.5	87.8±0.4	87.6±0.4	87.3±0.5	87.2±0.5	87.3±0.5	87.2±0.5	86.8±0.5	87.5±0.5	87.1±0.5	87.2±0.5	74±5	87.1±0.5	87.2±0.5
derma	97.2±0.5	97.5±0.5	97.5±0.6	97.2±0.5	97.2±0.5	97.2±0.5	97.2±0.5	97±0.4	96.9±0.6	97±0.5	97.2±0.5	87±3	97±0.5	97.2±0.5
ecoli	88.2±0.9	87.2±1	87.5±1	88.2±0.9	88.1±0.9	87.8±1	88.1±0.9	87.4±1	87.7±0.9	88.1±0.9	88.2±0.9	72±3	88.2±0.9	88.2±0.9
flare	81.6±0.5	81.7±0.6	82±0.5	81.6±0.5	81.6±0.5	81.6±0.5	81.8±0.6	81.6±0.7	82±0.5	81.9±0.6	81.6±0.5	80.2±1.5	82±0.6	81.6±0.5
glas	93.4±0.9	93.6±0.9	93.2±0.8	93.6±0.9	93.6±0.9	93.6±0.9	93.6±0.9	94±1.2	92.6±1	93.4±1	93.4±0.9	78±3	93.4±1	93.6±0.9
hear	82.7±1.9	81.9±1.9	81.9±1.8	83.2±1.9	82.7±1.9	83.2±1.9	82.5±1.8	82.4±1.5	83.4±1.8	82.5±1.7	82.7±1.9	76±3	82.7±1.7	82.5±1.8
img	97±0.3	97±0.3	97±0.3	97±0.3	97±0.3	97±0.3	96.9±0.3	96.8±0.3	96.9±0.3	96.9±0.3	97±0.3	90.8±1.4	96.9±0.3	97±0.3
ionos	90.6±1.1	90.1±1	90±1	90.3±1.1	90.4±1.1	90.3±1.1	90.4±0.9	91±0.9	91±1	90.4±0.9	90.6±1.1	76±4	90.3±1	90.4±1.1
mok1	99.8±0.3	99.6±0.4	99.6±0.4	99.8±0.3	99.8±0.3	99.8±0.3	99.6±0.3	100±0	99.6±0.3	99.6±0.3	99.8±0.3	88±5	99.6±0.3	99.8±0.3
mok2	91.4±1.3	89.8±1.4	89.8±1.4	91.9±0.9	91.1±1	91.9±0.9	91±1.4	91.9±1.4	91.4±1.3	91±1.4	91.4±1.3	80±4	91±1.4	91.5±1.2
pima	77.3±0.9	77.2±0.8	77.1±0.9	77.2±0.9	77.3±0.9	77.2±0.9	77±0.9	76.9±0.8	77.2±0.9	77±0.9	77.3±0.9	74.8±1.3	77.1±0.9	77.3±0.9
survi	75.6±1.5	74.9±1.5	75.3±1.3	75.1±1.5	75.4±1.5	75.1±1.5	75.7±1.7	74.9±1.7	75.6±1.5	75.4±1.8	75.6±1.5	70±5	75.4±1.8	75.7±1.6
vote	96.3±0.7	95.8±0.7	95.6±0.7	96.4±0.6	96.3±0.7	96.4±0.6	96±0.6	96.1±0.7	96.4±0.6	96±0.6	96.3±0.7	85±6	96±0.6	96.3±0.7
vowel	97.3±0.3	97±0.4	97.1±0.4	97.4±0.3	97.3±0.3	97.4±0.3	97.2±0.4	97.5±0.4	96.6±0.4	97.2±0.4	97.3±0.3	87±3	97.2±0.4	97.3±0.3
wdbc	97.3±0.3	97.3±0.3	97.4±0.3	97.3±0.3	97.3±0.3	97.3±0.3	97.2±0.3	97±0.3	97.2±0.3	97.2±0.3	97.3±0.3	92±2	97.2±0.3	97.3±0.3

Table C.111: Performance - Combination of *Simple Ensemble* of 20 *RBF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.4±0.9	75.1±0.9	74.8±1	75.5±0.8	75.4±0.9	75.5±0.8	75.5±1	74.6±0.7	75.8±1	–	75.4±0.9	65±4	–	–
bala	89.7±0.7	89.7±0.7	89.8±0.8	89.8±0.7	89.7±0.7	89.6±0.7	89.8±0.7	89.8±0.7	89.6±0.7	–	89.7±0.7	84±4	–	–
band	74±1.4	73.1±1.6	73.5±1.2	74±1.8	74±1.4	74±1.8	74.4±1.5	72.6±1.6	74.6±1.7	–	74±1.4	65±3	–	–
bupa	71.9±1.4	71.1±1.3	71.1±1.2	72.1±1.2	72±1.3	72.1±1.2	71.4±1.4	70.4±1.5	72±1.3	–	71.9±1.4	60±4	–	–
cred	87.2±0.6	87.2±0.6	87.3±0.6	87.3±0.5	87.1±0.6	87.3±0.5	87.2±0.5	86.2±0.7	87.2±0.5	–	87.2±0.6	79±4	–	–
derma	97.3±0.5	97.2±0.5	97.3±0.5	97.2±0.5	97.3±0.5	97.2±0.5	97.2±0.5	97.5±0.4	96.9±0.6	–	97.3±0.5	84±3	–	–
ecoli	87.9±1	88.1±1	88.2±1	88.2±0.9	87.8±1	88.1±0.9	87.9±1	87.2±1	87.7±0.9	–	87.9±1	64±8	–	–
flare	81.9±0.5	81.7±0.7	81.9±0.5	81.6±0.5	81.8±0.5	81.6±0.5	81.8±0.6	81.7±0.6	81.7±0.5	–	81.9±0.5	81.6±0.4	–	–
glas	93.2±1	93.2±1	93±1	93±1	93.4±1	93.2±1	93.2±1.1	93.2±1	92±1.1	–	93.2±1	75±4	–	–
hear	82.5±1.8	82.9±1.2	82.9±1.3	83.1±1.7	82.7±1.7	83.1±1.7	82.9±1.7	82.7±1.4	83.7±1.5	–	82.5±1.8	65±5	–	–
img	97±0.3	96.9±0.3	96.9±0.3	97±0.3	97±0.3	97±0.3	96.9±0.3	64±14	96.9±0.3	–	97±0.3	81±7	–	–
ionos	90.9±1.1	90.4±1.4	90.3±1.3	90.6±1	90.6±1.1	90.6±1	90.4±1	91.4±0.8	91±0.9	–	90.9±1.1	83±4	–	–
mok1	99.8±0.3	99.8±0.3	99.8±0.3	99.8±0.3	99.8±0.3	99.8±0.3	99.6±0.3	100±0	99.3±0.3	–	99.8±0.3	93.6±1.8	–	–
mok2	91.4±1.2	90.8±1.4	90.4±1.2	91.1±1.2	90.8±1.3	91.1±1.2	91.3±1.3	92.5±1.7	91.6±1	–	91.4±1.2	77±6	–	–
pima	77.4±0.9	77.1±0.9	77±1	77.2±1	77.4±0.9	77.2±1	77.2±0.9	76.4±0.8	76.9±0.9	–	77.4±0.9	72.7±1.3	–	–
survi	75.6±1.5	75.4±1.4	74.9±1.3	75.3±1.4	75.4±1.5	75.3±1.4	75.9±1.7	74.6±1.4	75.4±1.5	–	75.6±1.5	71±3	–	–
vote	95.9±0.6	96.3±0.5	95.9±0.5	96±0.6	95.9±0.6	96±0.6	95.9±0.7	96.3±0.8	96.1±0.6	–	95.9±0.6	86±6	–	–
vowel	97.3±0.4	97.3±0.4	97.1±0.3	97.4±0.4	97.3±0.4	97.3±0.4	97.2±0.4	97.6±0.3	96.3±0.4	–	97.3±0.4	94.8±1.7	–	–
wdbc	97.1±0.4	97±0.4	97.1±0.4	97.2±0.3	97.1±0.4	97.2±0.3	97.2±0.3	97.2±0.3	97.1±0.3	–	97.1±0.4	92±2	–	–

Table C.112: Performance - Combination of *Simple Ensemble* of 40 *RBF* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.3±0.9	75.1±1	74.9±1.1	75.3±0.9	75.2±0.9	75.3±0.9	75.3±0.8	74.3±0.8	76±1	–	75.3±0.9	67±2	–	–
bala	89.7±0.7	89.8±0.7	90±0.7	89.8±0.7	89.7±0.7	89.7±0.7	89.6±0.8	89.8±0.8	89.7±0.7	–	89.7±0.7	78±5	–	–
band	74.7±1.4	72.9±1.5	72.9±1.9	74.4±1.6	74.9±1.3	74.4±1.6	73.8±1.3	72.2±1.6	74.6±1.6	–	74.7±1.4	70±2	–	–
bupa	72.1±1.2	71.4±1.2	72±1.2	72.3±1.2	72.4±1.3	72.3±1.2	71.7±1.5	69.7±1.9	71.9±1.2	–	72.1±1.2	62±3	–	–
cred	87.2±0.6	87.5±0.6	87.3±0.5	87.2±0.6	87.2±0.6	87.2±0.6	86.9±0.5	86.7±0.7	87.2±0.6	–	87.2±0.6	82±3	–	–
derma	97.2±0.5	96.8±0.6	96.6±0.6	97.2±0.5	97.2±0.5	97.2±0.5	97.3±0.5	97.3±0.4	96.8±0.6	–	97.2±0.5	84±4	–	–
ecoli	88.1±0.9	87.9±0.9	87.7±0.9	88.1±1	87.5±0.9	87.9±1	87.9±1	87.4±1	87.5±0.9	–	88.1±0.9	78±4	–	–
flare	81.7±0.5	81.7±0.6	82±0.6	81.6±0.5	81.7±0.5	81.6±0.5	81.9±0.6	81.4±0.6	81.8±0.5	–	81.7±0.5	81.2±0.4	–	–
glas	93.2±1	92.8±1	93.2±1	93±1	93.2±1	93±1	93.2±1	91.4±0.8	91.8±1.1	–	93.2±1	82±4	–	–
hear	83.2±1.7	83.4±1.6	83.1±1.7	83.1±1.5	83.1±1.6	83.1±1.5	83.7±1.5	81.4±1.3	83.6±1.5	–	83.2±1.7	65±4	–	–
img	96.9±0.3	96.9±0.3	97±0.3	96.9±0.3	96.9±0.3	96.9±0.3	96.9±0.3	14.6±0.4	96.9±0.3	–	96.9±0.3	89.9±1.8	–	–
ionos	90.7±1.1	90.4±1.1	90.3±1.1	91±1.1	90.6±1.1	91±1.1	90.7±1.1	92.1±0.7	91.4±1	–	90.7±1.1	77±6	–	–
mok1	99.8±0.3	99.5±0.3	99.5±0.3	99.8±0.3	99.6±0.4	99.8±0.3	99.5±0.4	100±0	99±0.4	–	99.8±0.3	93±3	–	–
mok2	91.5±1.2	91.3±1.1	91.3±1.1	91.1±1.1	91±1.2	91.1±1.1	90.5±1.4	92.4±1.5	91.3±0.9	–	91.5±1.2	85±2	–	–
pima	77.4±0.9	77.1±0.9	77±0.9	77.2±1	77.4±0.9	77.2±1	77.3±0.9	76.6±0.8	77.1±0.9	–	77.4±0.9	69±4	–	–
survi	75.7±1.5	75.6±1.5	75.9±1.6	75.6±1.5	75.7±1.5	75.6±1.5	75.7±1.7	71.3±1.1	75.6±1.6	–	75.7±1.5	70±5	–	–
vote	96±0.6	96.1±0.5	96±0.6	96.1±0.6	96±0.6	96.1±0.6	96±0.8	95.6±0.8	96.3±0.6	–	96±0.6	90±2	–	–
vowel	97.2±0.3	97.2±0.4	97.3±0.4	97.3±0.3	97.2±0.3	97.3±0.3	97.2±0.4	9.3±0.3	96.2±0.4	–	97.2±0.3	97±0.5	–	–
wdbc	97.1±0.3	97.1±0.3	97.1±0.3	97.2±0.3	97.1±0.3	97.2±0.3	97.3±0.3	96.6±0.4	97.3±0.2	–	97.1±0.3	85±4	–	–

Table C.113: Performance - Combination of *Simple Ensemble* of 3 *RBF-Threshold* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.2±1	74.9±0.9	74.9±0.8	75.2±0.9	75.2±1	75.2±0.9	75.1±1	73.6±1.1	74.8±0.8	75.4±0.8	75.2±1	75.1±1.1	75.4±0.8	75.1±1
bala	89.6±0.7	89.6±0.7	89.6±0.7	89.6±0.7	89.6±0.7	89.6±0.7	89.8±0.7	89.5±0.7	89.7±0.7	89.9±0.7	89.6±0.7	89.3±0.6	89.8±0.7	89.6±0.7
band	72.9±1.5	71.6±1.8	71.5±1.7	71.6±1.4	72.7±1.5	71.6±1.4	72.7±1.5	70.4±1.4	71.6±1.4	72.6±1.6	72.9±1.5	72±1.4	72.7±1.8	72.9±1.5
bupa	71.7±1.3	72.6±1.4	72.4±1.4	72.3±1.3	71.7±1.3	72.3±1.3	72±1.2	71.3±1.1	72.3±1.3	72.4±1.2	71.7±1.3	72±1.4	72±1.4	71.9±1.2
cred	87±0.5	87±0.5	87.1±0.6	87.1±0.5	87±0.5	87.1±0.5	87.1±0.5	86.9±0.5	87.1±0.5	87.2±0.5	87±0.5	87.2±0.5	87.2±0.6	87±0.5
derma	96.8±0.5	97±0.4	97±0.4	96.9±0.5	96.8±0.5	96.9±0.5	97.2±0.5	96.9±0.6	96.8±0.5	97.2±0.5	96.8±0.5	96.8±0.3	97.2±0.5	96.8±0.5
ecoli	88.2±0.9	87.8±0.8	88.1±0.9	88.1±0.9	88.1±0.9	87.9±1	88.2±1	87.1±1.2	87.5±0.9	88.1±1	88.2±0.9	87.7±1.1	87.9±1	88.2±0.9
flare	81.7±0.5	81.9±0.5	82±0.5	81.6±0.5	81.6±0.6	81.6±0.5	81.8±0.6	81.9±0.8	81.9±0.5	81.8±0.6	81.7±0.5	82.1±0.6	81.8±0.6	81.6±0.5
glas	93.2±1	93.4±0.9	93.4±0.9	93.2±1	93.2±1	93.2±1	93.2±1	93.4±1.1	92.6±1	93.2±1	93.2±1	93.4±1	93.2±1	93.2±1
hear	83.6±1.7	83.4±1.7	83.4±1.7	83.6±1.8	83.6±1.7	83.6±1.8	82.5±2	84.1±1.5	83.6±1.8	82.7±1.9	83.6±1.7	83.9±1.6	82.4±1.8	83.6±1.7
img	96.9±0.3	96.9±0.3	96.9±0.3	97±0.3	96.9±0.3	97±0.3	97±0.3	97±0.3	96.9±0.3	96.9±0.3	96.9±0.3	96.9±0.3	96.9±0.3	96.9±0.3
ionos	90.7±1.1	90.7±1.1	90.4±1.1	90.3±1	90.7±1.1	90.3±1	90.7±1.1	90.7±1	90.7±1.1	90.9±1	90.7±1.1	92±1.1	90.7±1	90.7±1.1
mok1	99.6±0.3	99.75±0.17	99.8±0.3	99.8±0.3	99.6±0.3	99.8±0.3	99.6±0.3	100±0	100±0	99.6±0.3	99.6±0.3	100±0	99.6±0.3	99.6±0.3
mok2	90.1±1.3	89.5±0.8	89.3±1	91±0.9	90.4±1.2	91±0.9	90.9±1.5	90.9±1.2	91±0.9	91.1±1.3	90.1±1.3	90.9±1.2	91±1.3	90.3±1.2
pima	77.3±0.9	77.4±0.8	77.2±0.8	77±0.9	77.3±0.9	77±0.9	77.3±0.8	76.7±1	77.2±0.9	77.1±0.9	77.3±0.9	77.3±0.9	77.1±1	77.3±0.9
survi	75.6±1.5	75.3±1.4	75.1±1.5	75.4±1.6	75.4±1.5	75.4±1.6	76.1±1.7	76.2±1.8	75.4±1.6	75.9±1.8	75.6±1.5	75.6±1.6	76.1±1.8	75.6±1.5
vote	96.3±0.6	95.9±0.6	95.9±0.6	96.1±0.6	96.3±0.6	96.1±0.6	95.9±0.6	95.9±0.5	96.1±0.6	96.3±0.5	96.3±0.6	96.4±0.5	96.3±0.5	96.3±0.6
vowel	97.2±0.4	97.1±0.4	97±0.4	97.2±0.4	97.2±0.4	97.2±0.4	97.4±0.4	97.8±0.3	96.9±0.4	97.4±0.3	97.2±0.4	97.4±0.4	97.4±0.3	97.2±0.4
wdbc	97.2±0.3	97.1±0.4	97±0.4	97.2±0.3	97.2±0.3	97.2±0.3	97.3±0.3	96.6±0.3	97.1±0.3	97.3±0.3	97.2±0.3	97.3±0.4	97.3±0.3	97.2±0.3

Table C.114: Performance - Combination of *Simple Ensemble* of 9 *RBF-Threshold* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.4±0.9	75.1±1	74.9±0.9	75.4±0.8	75.4±0.9	75.4±0.8	75.4±1	73.9±1	75.2±0.9	75.1±1	75.4±0.9	74.9±0.9	75.1±1	75.4±0.9
bala	89.7±0.7	89.5±0.7	89.6±0.7	89.7±0.7	89.7±0.7	89.6±0.7	89.6±0.7	89.5±0.8	89.4±0.7	89.7±0.7	89.7±0.7	90±0.7	89.8±0.7	89.7±0.7
band	73.5±1.6	73.8±1.4	73.8±1.6	73.3±1.7	73.1±1.6	73.3±1.7	74.4±1.5	71.1±1.1	73.5±1.4	74.4±1.5	73.5±1.6	72.2±1.4	74.4±1.5	73.3±1.6
bupa	71.9±1.3	71.6±1.1	71.7±1.1	72±1.3	72±1.3	72±1.3	71.6±1.2	71.4±1.6	71.9±1.2	71.4±1.1	71.9±1.3	71.9±1.4	70.9±1.6	71.9±1.3
cred	87.2±0.5	87.8±0.4	87.6±0.4	87.3±0.5	87.2±0.5	87.3±0.5	87.2±0.5	86.6±0.6	87.5±0.5	87.1±0.5	87.2±0.5	86.7±0.5	87.1±0.5	87.2±0.5
derma	97.2±0.5	97.5±0.5	97.5±0.6	97.2±0.5	97.2±0.5	97.2±0.5	97.2±0.5	96.8±0.6	96.9±0.6	97±0.5	97.2±0.5	97.5±0.5	97±0.5	97.2±0.5
ecoli	88.2±0.9	87.2±1	87.5±1	88.2±0.9	87.9±0.9	87.8±1	88.1±0.9	86.9±1	87.7±0.9	88.1±0.9	88.2±0.9	85.4±1.1	88.2±0.9	88.2±0.9
flare	81.6±0.5	81.7±0.6	82±0.5	81.6±0.5	81.6±0.6	81.6±0.5	81.8±0.6	81.2±0.8	82±0.5	81.9±0.6	81.6±0.5	81.6±0.7	82±0.6	81.6±0.5
glas	93.4±0.9	93.6±0.9	93.2±0.8	93.6±0.9	93.6±0.9	93.6±0.9	93.6±0.9	93.4±1.3	92.6±1	93.4±1	93.4±0.9	92.2±1.4	93.4±1	93.6±0.9
hear	82.7±1.9	81.9±1.9	81.9±1.8	83.2±1.9	82.7±1.9	83.2±1.9	82.5±1.8	82.9±1.4	83.4±1.8	82.5±1.7	82.7±1.9	83.2±1.4	82.7±1.7	82.5±1.8
img	97±0.3	97±0.3	97±0.3	97±0.3	97±0.3	97±0.3	96.9±0.3	96.8±0.3	96.9±0.3	96.9±0.3	97±0.3	96.7±0.3	96.9±0.3	97±0.3
ionos	90.6±1.1	90.1±1	90±1	90.3±1.1	90.4±1.1	90.3±1.1	90.4±0.9	90.6±0.8	91±1	90.4±0.9	90.6±1.1	90.1±0.8	90.3±1	90.4±1.1
mok1	99.8±0.3	99.6±0.4	99.6±0.4	99.8±0.3	99.8±0.3	99.8±0.3	99.6±0.3	100±0	99.6±0.3	99.6±0.3	99.8±0.3	99.8±0.3	99.6±0.3	99.8±0.3
mok2	91.5±1.2	89.5±1.4	89.8±1.4	91.9±0.9	91±1.1	91.9±0.9	90.8±1.4	92.5±1.5	91.4±1.3	90.9±1.4	91.5±1.2	91±1.2	90.9±1.4	91.5±1.2
pima	77.3±0.9	77.2±0.8	77.1±0.9	77.2±0.9	77.3±0.9	77.2±0.9	77±0.9	76.7±0.6	77.2±0.9	77±0.9	77.3±0.9	77.4±0.9	77.1±0.9	77.3±0.9
survi	75.6±1.5	74.9±1.5	75.3±1.3	75.1±1.5	75.4±1.5	75.1±1.5	75.7±1.7	75.1±1.8	75.6±1.5	75.4±1.8	75.6±1.5	75.4±1.6	75.4±1.8	75.7±1.6
vote	96.3±0.7	95.9±0.6	95.9±0.6	96.4±0.6	96.3±0.7	96.4±0.6	96±0.6	96±0.7	96.4±0.6	96.5±0.5	96.3±0.7	96.1±0.5	96.5±0.6	96.3±0.7
vowel	97.3±0.3	97±0.4	97.1±0.4	97.4±0.3	97.3±0.3	97.4±0.3	97.2±0.4	97.4±0.4	96.6±0.4	97.2±0.4	97.3±0.3	97.4±0.3	97.2±0.4	97.3±0.3
wdbc	97.3±0.3	97.3±0.3	97.4±0.3	97.3±0.3	97.3±0.3	97.3±0.3	97.2±0.3	96.7±0.3	97.2±0.3	97.2±0.3	97.3±0.3	97.2±0.3	97.2±0.3	97.3±0.3

Table C.115: Performance - Combination of *Simple Ensemble* of 20 *RBF-Threshold* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.4±0.9	74.9±1	74.8±1	75.5±0.8	75.4±0.9	75.5±0.8	75.5±1	74.1±0.8	75.8±1	–	75.4±0.9	75.5±0.8	–	–
bala	89.7±0.7	89.7±0.7	89.8±0.8	89.8±0.7	89.7±0.7	89.6±0.7	89.8±0.7	89.5±0.8	89.6±0.7	–	89.7±0.7	90.4±0.7	–	–
band	74±1.4	73.1±1.6	73.5±1.2	74±1.8	74±1.4	74±1.8	74.4±1.5	72±1.5	74.6±1.7	–	74±1.4	74.6±1.5	–	–
bupa	71.9±1.4	71.1±1.3	71.1±1.2	72.1±1.2	72±1.3	72.1±1.2	71.4±1.4	70.7±1.8	72±1.3	–	71.9±1.4	69.9±1.7	–	–
cred	87.2±0.6	87.2±0.6	87.3±0.6	87.3±0.5	87.1±0.6	87.3±0.5	87.2±0.5	86.3±0.6	87.2±0.5	–	87.2±0.6	86.9±0.5	–	–
derma	97.3±0.5	97.2±0.5	97.3±0.5	97.2±0.5	97.3±0.5	97.2±0.5	97.2±0.5	96.9±0.6	96.9±0.6	–	97.3±0.5	97.2±0.6	–	–
ecoli	87.9±1	88.1±1	88.2±1	88.2±0.9	87.5±1	88.1±0.9	87.9±1	86.8±1	87.7±0.9	–	87.9±1	79.1±1.7	–	–
flare	81.9±0.5	81.7±0.7	81.9±0.5	81.6±0.5	81.8±0.5	81.6±0.5	81.8±0.6	81.8±0.6	81.7±0.5	–	81.9±0.5	81.7±0.6	–	–
glas	93.2±1	93.2±1	93±1	93±1	93.4±1	93.2±1	93.2±1.1	93.2±1.2	92±1.1	–	93.2±1	92±2	–	–
hear	82.5±1.8	82.9±1.2	82.9±1.3	83.1±1.7	82.7±1.7	83.1±1.7	82.9±1.7	82.2±1.7	83.7±1.5	–	82.5±1.8	83.2±1.5	–	–
img	97±0.3	96.9±0.3	96.9±0.3	97±0.3	97±0.3	97±0.3	96.9±0.3	67±13	96.9±0.3	–	97±0.3	96.4±0.6	–	–
ionos	90.9±1.1	90.4±1.4	90.3±1.3	90.6±1	90.6±1.1	90.6±1	90.4±1	90.4±1.1	91±0.9	–	90.9±1.1	88.7±0.7	–	–
mok1	99.8±0.3	99.8±0.3	99.8±0.3	99.8±0.3	99.8±0.3	99.8±0.3	99.6±0.3	100±0	99.3±0.3	–	99.8±0.3	99.88±0.13	–	–
mok2	91.8±1.1	90.6±1.4	90.4±1.2	91.1±1.2	90.3±1.3	91.1±1.2	91±1.3	92.4±1.7	91.6±1	–	91.8±1.1	90.6±1.6	–	–
pima	77.4±0.9	77.1±0.9	77±1	77.2±1	77.4±0.9	77.2±1	77.2±0.9	76.7±0.7	76.9±0.9	–	77.4±0.9	77±0.9	–	–
survi	75.6±1.5	75.4±1.4	74.9±1.3	75.3±1.4	75.4±1.5	75.3±1.4	75.9±1.7	73.9±1.4	75.4±1.5	–	75.6±1.5	74.8±1.4	–	–
vote	95.9±0.6	96.4±0.4	95.9±0.5	96±0.6	95.9±0.6	96±0.6	95.9±0.7	96.1±0.8	96.1±0.6	–	95.9±0.6	96.1±0.6	–	–
vowel	97.3±0.4	97.3±0.4	97.1±0.3	97.4±0.4	97.3±0.4	97.3±0.4	97.2±0.4	97.93±0.18	96.3±0.4	–	97.3±0.4	92.9±0.9	–	–
wdbc	97.1±0.4	97±0.4	97.1±0.4	97.2±0.3	97.1±0.4	97.2±0.3	97.2±0.3	96.5±0.4	97.1±0.3	–	97.1±0.4	97±0.3	–	–

Table C.116: Performance - Combination of *Simple Ensemble* of 40 *RBF-Threshold*

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.3±0.9	75.1±1	74.9±1.1	75.3±0.9	75.2±0.9	75.3±0.9	75.3±0.8	73.1±1.2	76±1	–	75.3±0.9	75.3±0.7	–	–
bala	89.7±0.7	89.8±0.7	90±0.7	89.8±0.7	89.7±0.7	89.7±0.7	89.6±0.8	89.4±0.8	89.7±0.7	–	89.7±0.7	90.4±0.7	–	–
band	74.7±1.4	72.9±1.5	72.9±1.9	74.4±1.6	74.9±1.3	74.4±1.6	73.8±1.3	71.5±1.1	74.6±1.6	–	74.7±1.4	72.6±1.1	–	–
bupa	72.1±1.2	71.4±1.2	72±1.2	72.3±1.2	72.1±1.3	72.3±1.2	71.6±1.4	69.4±1.7	71.9±1.2	–	72.1±1.2	71.1±1.6	–	–
cred	87.2±0.6	87.5±0.6	87.3±0.5	87.2±0.6	87.2±0.6	87.2±0.6	86.9±0.5	86.4±0.7	87.2±0.6	–	87.2±0.6	86.7±0.6	–	–
derma	97.2±0.5	96.8±0.6	96.6±0.6	97.2±0.5	97.2±0.5	97.2±0.5	97.3±0.5	97.3±0.3	96.8±0.6	–	97.2±0.5	89.6±1.9	–	–
ecoli	88.1±0.9	87.9±0.9	87.7±0.9	88.1±1	87.1±1	87.9±1	87.9±1	86.8±0.9	87.5±0.9	–	88.1±0.9	73±3	–	–
flare	81.7±0.5	81.7±0.6	82±0.6	81.6±0.5	81.7±0.6	81.6±0.5	81.9±0.6	81.2±0.6	81.8±0.5	–	81.7±0.5	81.1±0.5	–	–
glas	93.2±1	92.8±1	93.2±1	93±1	93.2±1	93±1	93.2±1	91.4±0.6	91.8±1.1	–	93.2±1	82±3	–	–
hear	83.2±1.7	83.4±1.6	83.1±1.7	83.1±1.5	83.1±1.6	83.1±1.5	83.7±1.5	82.4±1.2	83.6±1.5	–	83.2±1.7	82.7±1.6	–	–
img	96.9±0.3	96.9±0.3	97±0.3	96.9±0.3	96.9±0.3	96.9±0.3	96.9±0.3	14.6±0.4	96.9±0.3	–	96.9±0.3	91.3±1.2	–	–
ionos	90.7±1.1	90.4±1.1	90.3±1.1	91±1.1	90.6±1.1	91±1.1	90.7±1.1	90.6±0.9	91.4±1	–	90.7±1.1	87±0.9	–	–
mok1	99.8±0.3	99.5±0.3	99.5±0.3	99.8±0.3	99.6±0.4	99.8±0.3	99.5±0.4	100±0	99±0.4	–	99.8±0.3	99.6±0.4	–	–
mok2	91.6±1.2	91±1	91.3±1.1	91.1±1.1	90.1±1.3	91.1±1.1	90.4±1.4	92.3±1.6	91.3±0.9	–	91.6±1.2	90.5±1.5	–	–
pima	77.4±0.9	77.1±0.9	77±0.9	77.2±1	77.4±0.9	77.2±1	77.3±0.9	76.3±0.7	77.1±0.9	–	77.4±0.9	77.2±0.9	–	–
survi	75.7±1.5	75.6±1.5	75.9±1.6	75.6±1.5	75.7±1.5	75.6±1.5	75.7±1.7	72.3±0.9	75.6±1.6	–	75.7±1.5	74.8±1.5	–	–
vote	96±0.6	96.3±0.4	96±0.6	96.1±0.6	96±0.6	96.1±0.6	96±0.8	95.8±0.8	96.3±0.6	–	96±0.6	96.3±0.6	–	–
vowel	97.2±0.3	97.2±0.4	97.3±0.4	97.3±0.3	97.2±0.3	97.3±0.3	97.2±0.4	9.3±0.3	96.2±0.4	–	97.2±0.3	82±2	–	–
wdbc	97.1±0.3	97.1±0.3	97.1±0.3	97.2±0.3	97.1±0.3	97.2±0.3	97.3±0.3	95.6±0.4	97.3±0.2	–	97.1±0.3	97±0.3	–	–

Table C.117: Performance - Combination of *Simple Ensemble* of 3 *RBF-MinMax* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.2±0.9	74.8±0.9	74.8±0.9	75.2±0.9	75.1±0.9	75.2±0.9	75.2±0.7	74.4±1	74.8±0.8	74.6±1.1	75.2±0.9	75.2±1	74.6±1.1	75.2±0.9
bala	89.6±0.7	89.5±0.7	89.5±0.7	89.6±0.7	89.7±0.7	89.6±0.7	89.4±0.8	89.8±0.7	89.7±0.7	89.7±0.8	89.6±0.7	89.6±0.7	89.7±0.8	89.6±0.7
band	71.6±1.4	71.3±1.8	71.3±1.8	71.6±1.4	73.3±1.6	71.6±1.4	72.4±1.6	70.2±1.9	71.6±1.4	70.9±1.8	71.6±1.4	73.3±1.6	70.9±1.8	71.6±1.4
bupa	72.3±1.3	72.4±1.3	72.4±1.3	72.3±1.3	71.1±1	72.3±1.3	71.9±1.2	72±1.2	72.3±1.3	71.6±1.1	72.3±1.3	71.1±1	71.6±1.1	72.3±1.3
cred	87.1±0.5	87.1±0.6	87.1±0.6	87.1±0.5	86.9±0.6	87.1±0.5	87.1±0.5	83±3	87.1±0.5	87.2±0.6	87.1±0.5	86.9±0.6	87.2±0.6	87.1±0.5
derma	96.8±0.5	97±0.5	97±0.4	96.9±0.5	96.8±0.5	96.9±0.5	96.8±0.5	96.9±0.6	96.8±0.5	96.9±0.7	96.8±0.5	96.9±0.5	96.9±0.7	96.8±0.5
ecoli	88.2±0.9	87.9±0.8	88.1±0.9	88.1±0.9	88.1±0.9	87.9±1	87.9±0.9	87.5±0.9	87.5±0.9	87.5±1	88.2±0.9	87.9±1	86.9±1.2	88.2±0.9
flare	81.6±0.5	82±0.6	82±0.6	81.6±0.5	81.7±0.7	81.6±0.5	81.5±0.6	81.5±0.6	81.9±0.5	82±0.6	81.6±0.5	81.7±0.7	82±0.6	81.6±0.5
glas	93.2±1	93.4±1.2	93.4±0.9	93.2±1	93.2±1	93.2±1	93±1.1	93.6±1.1	92.6±1	93.2±1.2	93.2±1	93.4±1.1	93.2±1.2	93.2±1
hear	83.6±1.8	83.4±1.7	83.4±1.7	83.6±1.8	83.4±1.7	83.6±1.8	83.4±1.6	83.7±1.5	83.6±1.8	82.2±2	83.6±1.8	83.7±1.5	82.2±2	83.6±1.8
img	96.9±0.3	96.8±0.3	96.9±0.3	97±0.3	96.9±0.3	97±0.3	97±0.3	97±0.3	96.9±0.3	96.8±0.3	96.9±0.3	97±0.3	96.8±0.3	96.9±0.3
ionos	90.3±1	90.3±1.1	90.3±1.1	90.3±1	90.7±1.1	90.3±1	90.6±1.1	75±8	90.7±1.1	89.9±1.3	90.3±1	90.7±1.1	89.9±1.3	90.3±1
mok1	99.8±0.3	99.8±0.3	99.8±0.3	99.8±0.3	99.8±0.3	99.8±0.3	99.6±0.3	49.4±1.2	100±0	99.4±0.3	99.8±0.3	99.8±0.3	99.4±0.3	99.8±0.3
mok2	91±0.9	89.3±1	89.3±1	91±0.9	90.8±1.1	91±0.9	90.8±1.1	91.1±1.3	91±0.9	88.8±1.3	91±0.9	90.8±1.1	88.8±1.3	91±0.9
pima	77±0.9	77.2±0.8	77.2±0.8	77±0.9	77.2±0.8	77±0.9	77.1±0.9	75±1.8	77.2±0.9	77.1±0.8	77±0.9	77.2±0.7	77.1±0.8	77±0.9
survi	75.4±1.6	75.1±1.5	75.1±1.5	75.4±1.6	75.7±1.4	75.4±1.6	76.4±1.6	76.1±1.6	75.4±1.6	75.3±1.4	75.4±1.6	75.6±1.4	75.3±1.4	75.4±1.6
vote	96.1±0.6	95.8±0.5	95.8±0.5	96.1±0.6	96.1±0.6	96.1±0.6	96.1±0.7	86±5	96.1±0.6	95.5±0.5	96.1±0.6	96.1±0.6	95.5±0.5	96.1±0.6
vowel	97.2±0.4	97.1±0.3	97±0.4	97.2±0.4	97.2±0.4	97.2±0.4	97.3±0.4	97.4±0.4	96.9±0.4	97.1±0.4	97.2±0.4	97.3±0.4	97.1±0.4	97.3±0.4
wdbc	97.2±0.3	97.1±0.4	97.1±0.4	97.2±0.3	97.1±0.3	97.2±0.3	97.1±0.3	54±9	97.1±0.3	97±0.4	97.2±0.3	97.1±0.3	97±0.4	97.2±0.3

Table C.118: Performance - Combination of *Simple Ensemble* of 9 *RBF-MinMax* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.4±0.8	74.9±0.9	74.9±0.9	75.4±0.8	74.3±0.9	75.4±0.8	75.2±0.7	74.4±1.1	75.2±0.9	74.7±0.9	75.4±0.8	75.3±0.8	74.7±0.9	75.4±0.8
bala	89.6±0.7	89.7±0.8	89.5±0.8	89.7±0.7	89.7±0.7	89.6±0.7	89.4±0.8	89.9±0.7	89.4±0.7	89.2±0.7	89.6±0.7	89.7±0.8	89±0.7	89.6±0.7
band	73.3±1.7	73.5±1.5	73.5±1.5	73.3±1.7	73.6±2	73.3±1.7	72.4±1.6	70±2	73.5±1.4	70.6±2	73.3±1.7	73.6±2	70.6±2	73.3±1.7
bupa	72±1.3	71.7±1.1	71.7±1.1	72±1.3	70.3±1.3	72±1.3	71.9±1.2	69±3	71.9±1.2	70.4±1.4	72±1.3	70.3±1.3	70.4±1.4	72.1±1.2
cred	87.3±0.5	87.5±0.4	87.5±0.4	87.3±0.5	86.7±0.5	87.3±0.5	87.1±0.5	80±4	87.5±0.5	87.2±0.5	87.3±0.5	86.7±0.5	87.2±0.5	87.3±0.5
derma	97.2±0.5	97.3±0.6	97.5±0.6	97.2±0.5	97.2±0.5	97.2±0.5	96.8±0.5	97±0.4	96.9±0.6	97±0.5	97.2±0.5	97.3±0.5	96.9±0.5	97.2±0.5
ecoli	88.2±0.9	87.2±1	87.7±1.1	88.2±0.9	88.2±0.9	87.8±1	87.9±0.9	87.2±1	87.7±0.9	86.5±1.3	88.2±0.9	87.5±1	85.6±1.4	87.9±0.9
flare	81.6±0.5	82±0.5	82±0.5	81.6±0.5	81.7±0.7	81.6±0.5	81.5±0.6	51±11	82±0.5	82.2±0.6	81.6±0.5	81.7±0.7	82.2±0.6	81.6±0.5
glas	93.6±0.9	93.8±1	93.2±0.8	93.6±0.9	93.6±0.9	93.6±0.9	93±1.1	94.2±0.9	92.6±1	93.6±1.3	93.6±0.9	93.8±0.9	93±1.1	93.6±0.9
hear	83.2±1.9	81.9±1.8	81.9±1.8	83.2±1.9	82.9±1.7	83.2±1.9	83.4±1.6	77±4	83.4±1.8	80.7±1.9	83.2±1.9	82.9±1.7	80.7±1.9	83.2±1.9
img	97±0.3	96.8±0.3	96.9±0.3	97±0.3	97±0.3	97±0.3	97±0.3	96.9±0.3	96.9±0.3	96.8±0.3	97±0.3	97±0.3	96.7±0.3	97±0.3
ionos	90.3±1.1	89.9±1	89.9±1	90.3±1.1	92.4±1.2	90.3±1.1	90.6±1.1	58±9	91±1	89.6±1.4	90.3±1.1	92.4±1.2	89.6±1.4	90.3±1.1
mok1	99.8±0.3	99.6±0.4	99.6±0.4	99.8±0.3	99.4±0.3	99.8±0.3	99.6±0.3	49.4±1.2	99.6±0.3	98.9±0.5	99.8±0.3	99.4±0.3	98.9±0.5	99.8±0.3
mok2	91.9±0.9	89.9±1.4	89.9±1.4	91.9±0.9	89.6±1.2	91.9±0.9	90.8±1.1	83±4	91.4±1.3	86.6±1.2	91.9±0.9	89.6±1.2	86.6±1.2	91.9±0.9
pima	77.2±0.9	77±0.9	77±0.9	77.2±0.9	77.2±0.8	77.2±0.9	77.1±0.9	73±2	77.2±0.9	76.8±0.9	77.2±0.9	77.2±0.8	76.8±0.9	77.2±0.9
survi	75.1±1.5	75.3±1.3	75.3±1.3	75.1±1.5	75.9±1.7	75.1±1.5	76.4±1.6	75.7±1.7	75.6±1.5	74.8±1.8	75.1±1.5	75.6±1.8	74.8±1.8	75.1±1.5
vote	96.4±0.6	95.6±0.7	95.6±0.7	96.4±0.6	96.4±0.5	96.4±0.6	96.1±0.7	68±5	96.4±0.6	95.5±0.6	96.4±0.6	96.4±0.5	95.5±0.6	96.4±0.6
vowel	97.3±0.3	97±0.4	97.1±0.4	97.4±0.3	97.3±0.3	97.4±0.3	97.3±0.4	97.1±0.3	96.6±0.4	96.9±0.4	97.3±0.3	97.4±0.3	96.9±0.4	97.3±0.3
wdbc	97.3±0.3	97.4±0.3	97.4±0.3	97.3±0.3	97.2±0.3	97.3±0.3	97.1±0.3	36.3±0.7	97.2±0.3	97.2±0.4	97.3±0.3	97.2±0.3	97.2±0.4	97.3±0.3

Table C.119: Performance - Combination of *Simple Ensemble* of 20 *RBF-MinMax* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.5±0.8	74.6±1	74.6±1	75.5±0.8	73.7±1	75.5±0.8	75.2±0.7	74.4±1	75.8±1	–	75.5±0.8	74.1±1	–	–
bala	89.6±0.7	89.6±0.7	89.6±0.7	89.8±0.7	89.5±0.7	89.6±0.7	89.4±0.8	90.2±0.8	89.6±0.7	–	89.6±0.7	89.8±0.7	–	–
band	74±1.8	73.6±1.1	73.6±1.1	74±1.8	75±2	74±1.8	72.4±1.6	68±1.6	74.6±1.7	–	74±1.8	75±2	–	–
bupa	72.1±1.2	71.1±1.2	71.1±1.2	72.1±1.2	70.3±1.5	72.1±1.2	71.9±1.2	62±5	72±1.3	–	72.1±1.2	70.3±1.5	–	–
cred	87.3±0.5	87.3±0.6	87.3±0.6	87.3±0.5	86.6±0.5	87.3±0.5	87.1±0.5	68±5	87.2±0.5	–	87.3±0.5	86.6±0.5	–	–
derma	97.3±0.5	97.2±0.5	97.3±0.5	97.2±0.5	97.3±0.5	97.2±0.5	96.8±0.5	97.5±0.5	96.9±0.6	–	97.3±0.5	97.5±0.5	–	–
ecoli	88.1±0.9	88.1±1.1	88.2±1	88.2±0.9	87.9±0.9	88.1±0.9	87.9±0.9	87.5±1.1	87.7±0.9	–	88.1±0.9	87.4±1	–	–
flare	81.6±0.5	81.9±0.5	81.9±0.5	81.6±0.5	81.8±0.8	81.6±0.5	81.5±0.6	45±10	81.7±0.5	–	81.6±0.5	81.8±0.8	–	–
glas	93.4±1	93.2±1.2	93±1.2	93±1	93.4±1	93.2±1	93±1.1	93±1.2	92±1.1	–	93.4±1	93.6±0.9	–	–
hear	83.1±1.7	82.9±1.3	82.9±1.3	83.1±1.7	82.7±1.7	83.1±1.7	83.4±1.6	65±4	83.7±1.5	–	83.1±1.7	82.7±1.7	–	–
img	97±0.3	96.8±0.3	96.9±0.3	97±0.3	97±0.3	97±0.3	97±0.3	67±13	96.9±0.3	–	97±0.3	97±0.3	–	–
ionos	90.6±1	90.1±1.4	90.1±1.4	90.6±1	92.6±1.1	90.6±1	90.6±1.1	47±7	91±0.9	–	90.6±1	92.6±1.1	–	–
mok1	99.8±0.3	99.6±0.3	99.6±0.3	99.8±0.3	99±0.4	99.8±0.3	99.6±0.3	49.4±1.2	99.3±0.3	–	99.8±0.3	99±0.4	–	–
mok2	91.1±1.2	90.3±1.2	90.3±1.2	91.1±1.2	89.1±1.3	91.1±1.2	90.8±1.1	76±4	91.6±1	–	91.1±1.2	89.1±1.3	–	–
pima	77.2±1	77±1	77±1	77.2±1	77.2±0.8	77.2±1	77.1±0.9	68±2	76.9±0.9	–	77.2±1	77.2±0.8	–	–
survi	75.3±1.4	75.1±1.4	75.1±1.4	75.3±1.4	75.3±1.6	75.3±1.4	76.4±1.6	76.4±1.6	75.4±1.5	–	75.3±1.4	74.9±1.7	–	–
vote	96±0.6	95.9±0.5	95.9±0.5	96±0.6	96.1±0.7	96±0.6	96.1±0.7	60.4±0.9	96.1±0.6	–	96±0.6	96.1±0.7	–	–
vowel	97.3±0.4	96.9±0.4	97±0.4	97.4±0.4	97.3±0.4	97.3±0.4	97.3±0.4	97.1±0.4	96.3±0.4	–	97.3±0.4	97.1±0.4	–	–
wdbc	97.2±0.3	97±0.3	97±0.3	97.2±0.3	97.3±0.2	97.2±0.3	97.1±0.3	36.3±0.7	97.1±0.3	–	97.2±0.3	97.3±0.2	–	–

Table C.120: Performance - Combination of *Simple Ensemble* of 40 *RBF-MinMax* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.3±0.9	74.9±1	74.9±1	75.3±0.9	73.5±0.8	75.3±0.9	75.2±0.7	69±3	76±1	–	75.3±0.9	73.5±0.8	–	–
bala	89.7±0.7	89.7±0.7	90±0.7	89.8±0.7	89.5±0.7	89.7±0.7	89.4±0.8	89.6±1	89.7±0.7	–	89.7±0.7	89.8±0.7	–	–
band	74.4±1.6	72.9±2	72.9±2	74.4±1.6	73±2	74.4±1.6	72.4±1.6	66.9±1.2	74.6±1.6	–	74.4±1.6	73±2	–	–
bupa	72.3±1.2	71.9±1.2	71.9±1.2	72.3±1.2	70.3±1.5	72.3±1.2	71.9±1.2	55±5	71.9±1.2	–	72.3±1.2	70.3±1.5	–	–
cred	87.2±0.6	87.3±0.5	87.3±0.5	87.2±0.6	86.5±0.6	87.2±0.6	87.1±0.5	62±4	87.2±0.6	–	87.2±0.6	86.5±0.6	–	–
derma	97.2±0.5	96.8±0.6	96.6±0.6	97.2±0.5	97.2±0.5	97.2±0.5	96.8±0.5	97.6±0.5	96.8±0.6	–	97.2±0.5	97±0.4	–	–
ecoli	88.1±0.9	87.2±0.9	87.7±0.9	88.1±1	87.8±0.9	87.9±1	87.9±0.9	87.2±1.1	87.5±0.9	–	88.1±0.9	87.8±0.9	–	–
flare	81.6±0.5	82±0.6	82±0.6	81.6±0.5	81.9±0.8	81.6±0.5	81.5±0.6	18.7±0.5	81.8±0.5	–	81.6±0.5	81.9±0.8	–	–
glas	93±1	92.6±1.1	93.2±1	93±1	93.2±1	93±1	93±1.1	93.2±1	91.8±1.1	–	93±1	93.2±1	–	–
hear	83.1±1.5	83.1±1.7	83.1±1.7	83.1±1.5	82.9±1.6	83.1±1.5	83.4±1.6	60±4	83.6±1.5	–	83.1±1.5	82.9±1.6	–	–
img	96.9±0.3	96.8±0.2	97±0.2	96.9±0.3	96.9±0.3	96.9±0.3	97±0.3	14.6±0.4	96.9±0.3	–	96.9±0.3	96.9±0.3	–	–
ionos	91±1.1	90.1±1.2	90.1±1.2	91±1.1	92.9±1	91±1.1	90.6±1.1	36.6±1.3	91.4±1	–	91±1.1	92.9±1	–	–
mok1	99.8±0.3	99.5±0.3	99.5±0.3	99.8±0.3	98.8±0.4	99.8±0.3	99.6±0.3	49.4±1.2	99±0.4	–	99.8±0.3	98.8±0.4	–	–
mok2	91.1±1.1	91.3±1.1	91.3±1.1	91.1±1.1	87.8±1	91.1±1.1	90.8±1.1	71±3	91.3±0.9	–	91.1±1.1	87.8±1	–	–
pima	77.2±1	77±0.9	77±0.9	77.2±1	77.2±0.8	77.2±1	77.1±0.9	67±1.9	77.1±0.9	–	77.2±1	77.2±0.8	–	–
survi	75.6±1.5	75.9±1.6	75.9±1.6	75.6±1.5	75.3±1.6	75.6±1.5	76.4±1.6	75.1±0.8	75.6±1.6	–	75.6±1.5	75.3±1.6	–	–
vote	96.1±0.6	95.9±0.5	95.9±0.5	96.1±0.6	96±0.6	96.1±0.6	96.1±0.7	60.4±0.9	96.3±0.6	–	96.1±0.6	96±0.6	–	–
vowel	97.3±0.3	96.8±0.4	97.2±0.4	97.3±0.3	97.3±0.3	97.3±0.3	97.3±0.4	9.3±0.3	96.2±0.4	–	97.3±0.3	97.2±0.3	–	–
wdbc	97.2±0.3	97.1±0.3	97.1±0.3	97.2±0.3	97.1±0.3	97.2±0.3	97.1±0.3	36.3±0.7	97.3±0.2	–	97.2±0.3	97.4±0.2	–	–

Table C.121: Performance - Combination of *Simple Ensemble* of 3 *RBF-Sum* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.2±1	74.8±0.9	74.8±0.9	75.2±0.9	75.2±1	75.2±0.9	75.1±1	73.6±1.3	74.8±0.8	75.2±0.8	75.2±1	75.3±1.1	75.2±0.8	75.1±1
bala	89.6±0.7	89.7±0.7	89.6±0.7	89.6±0.7	89.6±0.7	89.6±0.7	89.8±0.8	89.7±0.7	89.7±0.7	89.7±0.8	89.6±0.7	89.6±0.7	89.7±0.8	89.6±0.7
band	72.7±1.5	71.3±1.8	71.3±1.8	71.6±1.4	72.7±1.5	71.6±1.4	73.1±1.2	70.7±1.2	71.6±1.4	72.9±1.1	72.7±1.5	72.4±1.4	72.7±1.1	72.7±1.5
bupa	71.7±1.3	72.4±1.3	72.4±1.3	72.3±1.3	71.7±1.3	72.3±1.3	71.9±1.2	71.4±1.1	72.3±1.3	71.7±1.3	71.7±1.3	72.3±1.3	71.4±1.5	71.9±1.2
cred	87±0.5	87.1±0.6	87.1±0.6	87.1±0.5	87±0.5	87.1±0.5	87±0.5	86.9±0.6	87.1±0.5	87.1±0.6	87±0.5	87.1±0.5	87±0.5	87±0.5
derma	96.8±0.5	97±0.4	97±0.5	96.9±0.5	96.8±0.5	96.9±0.5	96.8±0.5	96.9±0.7	96.8±0.5	96.8±0.5	96.8±0.5	96.8±0.3	96.8±0.5	96.8±0.5
ecoli	88.2±0.9	88.2±0.9	88.1±0.9	88.1±0.9	88.1±0.9	87.9±1	88.4±0.9	87.7±1.2	87.5±0.9	88.4±1	88.2±0.9	87.5±1.1	88.2±1	88.2±0.9
flare	81.7±0.5	82±0.6	82±0.6	81.6±0.5	81.6±0.6	81.6±0.5	81.7±0.6	81.7±0.7	81.9±0.5	81.8±0.6	81.7±0.5	82.1±0.6	82±0.7	81.6±0.5
glas	93.2±1	93.4±0.9	93.4±0.9	93.2±1	93.2±1	93.2±1	93.2±1	93.6±0.9	92.6±1	93.6±1.2	93.2±1	93±1.1	93.6±1.2	93.2±1
hear	83.6±1.7	83.4±1.7	83.4±1.7	83.6±1.8	83.6±1.7	83.6±1.8	82.9±1.9	83.6±1.4	83.6±1.8	82.7±1.8	83.6±1.7	84.1±1.6	82.5±1.6	83.6±1.7
img	96.9±0.3	96.8±0.3	96.9±0.3	97±0.3	96.9±0.3	97±0.3	96.9±0.3	97±0.3	96.9±0.3	96.9±0.3	96.9±0.3	96.9±0.3	96.9±0.3	96.9±0.3
ionos	90.7±1.1	90.3±1.1	90.3±1.1	90.3±1	90.7±1.1	90.3±1	90.7±1.1	90.9±1	90.7±1.1	90.7±1.1	90.7±1.1	92±1	90.7±1.1	90.7±1.1
mok1	99.6±0.3	99.8±0.3	99.8±0.3	99.8±0.3	99.6±0.3	99.8±0.3	99.75±0.17	99.8±0.3	100±0	99.88±0.13	99.6±0.3	100±0	99.88±0.13	99.6±0.3
mok2	90.4±1.1	89.3±1	89.3±1	91±0.9	90.4±1.2	91±0.9	90±1.3	90.8±1.2	91±0.9	90±1.4	90.4±1.1	91±1.2	90±1.3	90.4±1.1
pima	77.3±0.9	77.2±0.8	77.2±0.8	77±0.9	77.3±0.9	77±0.9	77.3±0.9	76.8±1	77.2±0.9	77±1	77.3±0.9	77.3±0.9	77.1±1	77.3±0.9
survi	75.4±1.5	75.1±1.5	75.1±1.5	75.4±1.6	75.4±1.5	75.4±1.6	75.7±1.5	75.9±1.8	75.4±1.6	76.2±1.6	75.4±1.5	75.7±1.5	75.9±1.7	75.4±1.5
vote	96.3±0.6	95.8±0.5	95.8±0.5	96.1±0.6	96.3±0.6	96.1±0.6	96.1±0.7	95.8±0.5	96.1±0.6	96.1±0.7	96.3±0.6	96.1±0.6	96.1±0.7	96.3±0.6
vowel	97.3±0.4	97.1±0.3	97.1±0.3	97.2±0.4	97.2±0.4	97.2±0.4	97.4±0.3	96.8±0.3	96.9±0.4	97.4±0.3	97.3±0.4	97.1±0.3	97.4±0.3	97.3±0.4
wdbc	97.2±0.3	97.1±0.4	97.1±0.4	97.2±0.3	97.2±0.3	97.2±0.3	97.3±0.3	96.6±0.3	97.1±0.3	97.3±0.3	97.2±0.3	97.2±0.3	97.3±0.3	97.2±0.3

Table C.122: Performance - Combination of *Simple Ensemble* of 9 *RBF-Sum* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.4±0.9	74.9±0.9	74.9±0.9	75.4±0.8	75.4±0.9	75.4±0.8	75.1±0.9	73.6±1.4	75.2±0.9	75.3±0.9	75.4±0.9	74.8±0.9	75.3±0.9	75.4±0.9
bala	89.7±0.7	89.5±0.7	89.6±0.7	89.7±0.7	89.7±0.7	89.6±0.7	89.8±0.7	89.4±0.8	89.4±0.7	89.8±0.7	89.7±0.7	90.1±0.7	89.8±0.7	89.7±0.7
band	73.1±1.6	73.5±1.5	73.5±1.5	73.3±1.7	73.1±1.6	73.3±1.7	74.6±1.3	71.3±1.2	73.5±1.4	74.7±1.3	73.1±1.6	72.2±1.6	74.6±1.4	73.3±1.6
bupa	72±1.3	71.7±1.1	71.7±1.1	72±1.3	72±1.3	72±1.3	72.1±1.2	71.1±1.4	71.9±1.2	71.6±1.3	72±1.3	72±1.4	70.6±1.5	72±1.3
cred	87.2±0.5	87.5±0.4	87.5±0.4	87.3±0.5	87.2±0.5	87.3±0.5	87.2±0.5	86.5±0.6	87.5±0.5	87.2±0.5	87.2±0.5	86.6±0.5	87.2±0.5	87.2±0.5
derma	97.2±0.5	97.5±0.5	97.3±0.5	97.2±0.5	97.2±0.5	97.2±0.5	97.2±0.5	96.8±0.6	96.9±0.6	97.2±0.5	97.2±0.5	96.5±1	97.2±0.5	97.2±0.5
ecoli	88.2±0.9	87.5±1.1	87.5±1	88.2±0.9	88.2±0.9	87.8±1	88.1±0.9	86.9±1.5	87.7±0.9	88.1±0.9	88.2±0.9	82.8±1.3	88.2±0.9	88.2±0.9
flare	81.7±0.5	82±0.5	82±0.5	81.6±0.5	81.6±0.6	81.6±0.5	81.8±0.6	81.2±0.7	82±0.5	81.9±0.6	81.7±0.5	81.6±0.7	82.1±0.6	81.6±0.5
glas	93.6±0.9	93.8±0.9	93.2±0.8	93.6±0.9	93.6±0.9	93.6±0.9	93.4±1.1	94.8±1.1	92.6±1	93.6±1.1	93.6±0.9	92.8±1.1	93.6±1.1	93.6±0.9
hear	82.7±1.9	81.9±1.8	81.9±1.8	83.2±1.9	82.7±1.9	83.2±1.9	82.5±1.8	82.2±1.4	83.4±1.8	82.4±1.8	82.7±1.9	83.2±1.4	82.9±1.8	82.7±1.9
img	97±0.3	97±0.3	97±0.3	97±0.3	97±0.3	97±0.3	97±0.3	97±0.3	96.9±0.3	97±0.3	97±0.3	96.9±0.3	97±0.3	97±0.3
ionos	90.4±1.1	89.9±1	89.9±1	90.3±1.1	90.4±1.1	90.3±1.1	90.3±1	89.9±0.8	91±1	90.1±1.1	90.4±1.1	90±0.9	90.1±1.1	90.4±1.1
mok1	99.8±0.3	99.6±0.4	99.6±0.4	99.8±0.3	99.8±0.3	99.8±0.3	99.6±0.3	99.88±0.13	99.6±0.3	99.6±0.3	99.8±0.3	99.8±0.3	99.6±0.3	99.8±0.3
mok2	91.5±1	89.9±1.4	89.9±1.4	91.9±0.9	91±1.1	91.9±0.9	90.6±1.3	92.9±1.4	91.4±1.3	90.5±1.3	91.5±1	91±1.2	90.6±1.3	91.5±1
pima	77.3±0.9	77±0.9	77±0.9	77.2±0.9	77.3±0.9	77.2±0.9	77.2±0.9	76.9±0.5	77.2±0.9	77±0.9	77.3±0.9	77.4±0.9	77.2±0.9	77.3±0.9
survi	75.4±1.5	75.3±1.3	75.3±1.3	75.1±1.5	75.4±1.5	75.1±1.5	75.9±1.5	73.8±1.6	75.6±1.5	76.1±1.6	75.4±1.5	75.4±1.6	76.1±1.5	75.1±1.4
vote	96.3±0.7	95.6±0.7	95.6±0.7	96.4±0.6	96.3±0.7	96.4±0.6	96±0.6	96±0.7	96.4±0.6	96±0.6	96.3±0.7	96.3±0.5	96±0.6	96.3±0.7
vowel	97.4±0.4	97.1±0.4	97.1±0.4	97.4±0.3	97.3±0.3	97.4±0.3	97.6±0.3	95.9±0.4	96.6±0.4	97.5±0.3	97.4±0.4	96.4±0.9	97.5±0.3	97.4±0.4
wdbc	97.3±0.3	97.4±0.3	97.4±0.3	97.3±0.3	97.3±0.3	97.3±0.3	97.1±0.3	96.4±0.4	97.2±0.3	97.1±0.3	97.3±0.3	97.1±0.3	97.1±0.3	97.3±0.3

Table C.123: Performance - Combination of *Simple Ensemble* of 20 *RBF-Sum* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.4±0.9	74.6±1	74.6±1	75.5±0.8	75.4±0.9	75.5±0.8	75.1±0.9	73.2±0.9	75.8±1	–	75.4±0.9	75.4±1.1	–	–
bala	89.7±0.7	89.8±0.8	89.8±0.8	89.8±0.7	89.7±0.7	89.6±0.7	89.6±0.7	89.4±0.8	89.6±0.7	–	89.7±0.7	90.3±0.7	–	–
band	74±1.4	73.6±1.1	73.6±1.1	74±1.8	74±1.4	74±1.8	74.7±1.3	71.8±1.7	74.6±1.7	–	74±1.4	74.6±1.5	–	–
bupa	72±1.3	71.1±1.2	71.1±1.2	72.1±1.2	72±1.3	72.1±1.2	72±1.2	69±1.4	72±1.3	–	72±1.3	70.1±1.7	–	–
cred	87.1±0.6	87.3±0.6	87.3±0.6	87.3±0.5	87.1±0.6	87.3±0.5	87.2±0.6	86.3±0.6	87.2±0.5	–	87.1±0.6	86.9±0.5	–	–
derma	97.3±0.5	97.2±0.5	97.3±0.5	97.2±0.5	97.3±0.5	97.2±0.5	97.3±0.5	97.3±0.4	96.9±0.6	–	97.3±0.5	89±2	–	–
ecoli	88.1±0.9	87.9±1	88.1±1	88.2±0.9	87.9±0.9	88.1±0.9	88.1±0.9	86±1.2	87.7±0.9	–	88.1±0.9	67±3	–	–
flare	81.9±0.5	81.9±0.5	81.9±0.5	81.6±0.5	81.8±0.5	81.6±0.5	81.9±0.6	81.5±0.6	81.7±0.5	–	81.9±0.5	81.5±0.6	–	–
glas	93.4±1	93.2±1	93±1	93±1	93.4±1	93.2±1	93.8±1.1	93.2±1.3	92±1.1	–	93.4±1	89±4	–	–
hear	82.7±1.7	82.9±1.3	82.9±1.3	83.1±1.7	82.7±1.7	83.1±1.7	82.5±1.7	82.2±1.6	83.7±1.5	–	82.7±1.7	83.2±1.5	–	–
img	97±0.3	96.9±0.3	96.9±0.3	97±0.3	97±0.3	97±0.3	97±0.3	64±14	96.9±0.3	–	97±0.3	95.5±1.6	–	–
ionos	90.6±1.1	90.1±1.4	90.1±1.4	90.6±1	90.6±1.1	90.6±1	90.3±0.9	91.1±0.8	91±0.9	–	90.6±1.1	89.3±0.8	–	–
mok1	99.8±0.3	99.6±0.3	99.6±0.3	99.8±0.3	99.8±0.3	99.8±0.3	99.6±0.3	100±0	99.3±0.3	–	99.8±0.3	99.88±0.13	–	–
mok2	91.3±1.2	90.3±1.2	90.3±1.2	91.1±1.2	90.4±1.2	91.1±1.2	90.3±1.4	92.3±1.6	91.6±1	–	91.3±1.2	90.5±1.6	–	–
pima	77.4±0.9	77±1	77±1	77.2±1	77.4±0.9	77.2±1	77.2±0.9	76.5±0.7	76.9±0.9	–	77.4±0.9	77±0.8	–	–
survi	75.4±1.5	75.1±1.4	75.1±1.4	75.3±1.4	75.4±1.5	75.3±1.4	75.6±1.4	73.8±1.3	75.4±1.5	–	75.4±1.5	75.6±1.6	–	–
vote	95.9±0.6	95.9±0.5	95.9±0.5	96±0.6	95.9±0.6	96±0.6	96±0.6	95.6±0.8	96.1±0.6	–	95.9±0.6	96±0.6	–	–
vowel	97.3±0.4	97.2±0.4	97.1±0.3	97.4±0.4	97.3±0.4	97.3±0.4	97.5±0.4	95.5±0.4	96.3±0.4	–	97.3±0.4	67±4	–	–
wdbc	97.1±0.4	97±0.3	97±0.3	97.2±0.3	97.1±0.4	97.2±0.3	97.1±0.3	95.6±0.4	97.1±0.3	–	97.1±0.4	97±0.3	–	–

Table C.124: Performance - Combination of *Simple Ensemble* of 40 *RBF-Sum* networks

dataset	average	dan	dan2	vote	nash	borda	wta	w.ave	bayes	choquet	badd	zimm	choquet.dd	wave.ddw
aritm	75.3±0.9	74.9±1	74.9±1	75.3±0.9	75.2±0.9	75.3±0.9	75.1±0.7	73.7±1.4	76±1	–	75.3±0.9	74.7±1	–	–
bala	89.7±0.7	89.8±0.7	90±0.7	89.8±0.7	89.7±0.7	89.7±0.7	89.6±0.7	89.6±0.8	89.7±0.7	–	89.7±0.7	90.2±0.7	–	–
band	74.7±1.4	72.9±2	72.9±2	74.4±1.6	74.9±1.3	74.4±1.6	74.2±1.4	71±2	74.6±1.6	–	74.7±1.4	72.6±1.1	–	–
bupa	72.3±1.2	71.9±1.2	71.9±1.2	72.3±1.2	72.1±1.3	72.3±1.2	72.3±1.4	67.9±2	71.9±1.2	–	72.3±1.2	71±1.6	–	–
cred	87.2±0.6	87.3±0.5	87.3±0.5	87.2±0.6	87.2±0.6	87.2±0.6	87±0.5	86.2±0.7	87.2±0.6	–	87.2±0.6	86.9±0.5	–	–
derma	97.2±0.5	97.2±0.6	96.8±0.6	97.2±0.5	97.2±0.5	97.2±0.5	97.5±0.5	96.9±0.6	96.8±0.6	–	97.2±0.5	63±5	–	–
ecoli	88.1±0.9	87.9±0.9	87.7±0.9	88.1±1	87.8±0.9	87.9±1	88.2±0.9	84.9±1.2	87.5±0.9	–	88.1±0.9	76±4	–	–
flare	81.7±0.5	82±0.6	82±0.6	81.6±0.5	81.7±0.6	81.6±0.5	81.9±0.6	81.4±0.6	81.8±0.5	–	81.7±0.5	81.1±0.5	–	–
glas	93.2±1	93±1	93.2±1	93±1	93.2±1	93±1	93.8±1.1	92.8±1.4	91.8±1.1	–	93.2±1	72±6	–	–
hear	83.2±1.7	83.1±1.7	83.1±1.7	83.1±1.5	83.1±1.6	83.1±1.5	83.4±1.5	81.9±1.1	83.6±1.5	–	83.2±1.7	82.9±1.6	–	–
img	96.9±0.3	97±0.3	97±0.3	96.9±0.3	96.9±0.3	96.9±0.3	97±0.3	14.6±0.4	96.9±0.3	–	96.9±0.3	66±3	–	–
ionos	90.6±1.1	90.1±1.2	90.1±1.2	91±1.1	90.6±1.1	91±1.1	89.9±0.9	90.3±1.2	91.4±1	–	90.6±1.1	88.4±1	–	–
mok1	99.8±0.3	99.5±0.3	99.5±0.3	99.8±0.3	99.6±0.4	99.8±0.3	99.5±0.4	99.88±0.13	99±0.4	–	99.8±0.3	99.6±0.4	–	–
mok2	91±1.1	91.3±1.1	91.3±1.1	91.1±1.1	90.1±1.3	91.1±1.1	90.1±1.6	92.6±1.4	91.3±0.9	–	91±1.1	90±1.7	–	–
pima	77.4±0.9	77±0.9	77±0.9	77.2±1	77.4±0.9	77.2±1	77.2±0.9	76.3±0.8	77.1±0.9	–	77.4±0.9	77.2±0.9	–	–
survi	75.7±1.5	75.9±1.6	75.9±1.6	75.6±1.5	75.7±1.5	75.6±1.5	75.6±1.5	72±1.1	75.6±1.6	–	75.7±1.5	73.9±1.1	–	–
vote	96±0.6	95.9±0.5	95.9±0.5	96.1±0.6	96±0.6	96.1±0.6	96±0.6	95.5±0.8	96.3±0.6	–	96±0.6	96.1±0.7	–	–
vowel	97.3±0.3	97.3±0.4	97.4±0.4	97.3±0.3	97.3±0.3	97.3±0.3	97.5±0.4	9.3±0.3	96.2±0.4	–	97.3±0.3	88±6	–	–
wdbc	97.1±0.3	97.1±0.3	97.1±0.3	97.2±0.3	97.1±0.3	97.2±0.3	97.1±0.3	95.6±0.5	97.3±0.2	–	97.1±0.3	97±0.3	–	–

Table C.125: Performance - Experts of *Ting & Witten* as ensemble

Database	3-expert	9-expert	20-expert	40-expert
aritm	72.5 ± 1.5	72.4 ± 1.5	72.4 ± 1.5	72.4 ± 1.5
bala	94.7 ± 0.8	94.8 ± 0.7	94.9 ± 0.7	95.0 ± 0.7
band	71.8 ± 1.8	72.7 ± 1.5	72.9 ± 1.4	73.1 ± 1.5
bupa	71.1 ± 1.8	71.9 ± 1.8	72.3 ± 1.5	72.4 ± 1.7
cred	86.4 ± 0.6	86.5 ± 0.6	86.5 ± 0.6	86.5 ± 0.6
derma	96.6 ± 0.7	96.8 ± 0.6	96.6 ± 0.9	96.3 ± 0.9
ecoli	86.0 ± 1.0	86.0 ± 0.9	85.7 ± 1.0	85.4 ± 1.1
flare	82.0 ± 0.6	81.8 ± 0.6	81.8 ± 0.6	81.9 ± 0.6
glas	92.4 ± 0.5	92.8 ± 0.7	92.8 ± 0.5	93.0 ± 0.5
hear	84.2 ± 1.3	83.9 ± 1.3	83.6 ± 1.5	83.9 ± 1.3
img	96.5 ± 0.2	96.6 ± 0.2	96.6 ± 0.3	96.6 ± 0.2
ionos	88.0 ± 1.3	89.0 ± 1.2	88.9 ± 1.2	89.1 ± 1.1
mok1	98.5 ± 0.9	98.6 ± 0.9	98.6 ± 0.9	98.6 ± 0.9
mok2	84.5 ± 1.4	85.4 ± 1.4	86.6 ± 1.6	87.3 ± 1.5
pima	77.4 ± 0.8	77.2 ± 0.7	77.4 ± 0.7	77.2 ± 0.8
survi	74.9 ± 1.4	74.9 ± 1.4	75.1 ± 1.4	75.2 ± 1.4
vote	95.4 ± 0.8	95.4 ± 0.8	95.4 ± 0.8	95.4 ± 0.8
vowel	86.7 ± 0.7	88.6 ± 0.6	89.7 ± 0.7	89.8 ± 0.4
wdbc	96.9 ± 0.5	97.0 ± 0.6	97.0 ± 0.6	97.0 ± 0.6

Table C.126: Performance - Implementation of *Stacked* by *Ting & Witten*

Database	3-expert	9-expert	20-expert	40-expert
aritm	74.8 ± 1.5	74 ± 2	75 ± 2	74.9 ± 1.9
bala	96.3 ± 0.4	96.5 ± 0.4	96.9 ± 0.3	96.9 ± 0.4
band	70 ± 2	70.5 ± 1.9	71.3 ± 1.9	71 ± 2
bupa	71.1 ± 1.6	72.6 ± 1.5	71.4 ± 1.6	72.3 ± 1.8
cred	85.0 ± 0.9	85.2 ± 1.0	85.7 ± 0.7	85.4 ± 0.8
derma	96.6 ± 0.9	96.5 ± 1.0	96.5 ± 1.0	96.6 ± 1.1
ecoli	85.0 ± 1.2	84.3 ± 1.0	85.4 ± 1.3	84.6 ± 1.1
flare	81.8 ± 0.7	81.8 ± 0.7	82.0 ± 0.8	81.8 ± 0.7
glas	94.4 ± 0.6	95.0 ± 0.7	94.8 ± 0.7	96.0 ± 0.6
hear	83.4 ± 1.1	83.2 ± 1.0	81.9 ± 1.2	81.9 ± 1.5
img	96.9 ± 0.2	97.1 ± 0.3	97.1 ± 0.3	97.2 ± 0.2
ionos	89.4 ± 1.2	90.6 ± 0.9	90.6 ± 0.7	91.1 ± 1.0
mok1	98.6 ± 0.9	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	85.0 ± 1.3	85.9 ± 1.6	88.1 ± 1.0	88.8 ± 1.1
pima	76.3 ± 1.1	76.2 ± 1.1	76.5 ± 0.9	75.9 ± 1.4
survi	73.1 ± 1.2	73.4 ± 0.6	73.1 ± 1.0	73.4 ± 1.1
vote	95.5 ± 0.7	95.4 ± 0.6	95.3 ± 0.6	95.4 ± 0.6
vowel	87.0 ± 0.5	90.7 ± 0.6	92.6 ± 0.7	92.7 ± 0.7
wdbc	96.7 ± 0.6	96.6 ± 0.5	96.6 ± 0.5	96.5 ± 0.5

Table C.127: Performance - Experts of *Ghorbani & Owrangh* as ensemble

Database	3-expert	9-expert	20-expert	40-expert
aritm	74.5 ± 1.2	74.3 ± 1.0	74.3 ± 1.4	74.0 ± 1.0
bala	94.1 ± 0.7	95.8 ± 0.6	95.2 ± 0.5	95.0 ± 0.7
band	73.6 ± 1.3	74.7 ± 1.3	74.7 ± 1.0	76.2 ± 1.0
bupa	73.7 ± 1.3	73.3 ± 1.6	72.0 ± 1.3	72.0 ± 1.3
cred	86.8 ± 0.8	86.8 ± 0.6	87.4 ± 0.6	87.1 ± 0.7
derma	97.5 ± 0.7	97.7 ± 0.6	97.3 ± 0.6	97.3 ± 0.4
ecoli	87.5 ± 0.9	87.2 ± 1.0	87.2 ± 0.9	86.6 ± 1.2
flare	82.1 ± 0.5	81.9 ± 0.7	82.0 ± 0.6	81.7 ± 0.7
glas	94.2 ± 1.0	93.2 ± 0.9	93.8 ± 0.7	93.6 ± 0.9
hear	84.6 ± 1.3	84.6 ± 1.4	84.7 ± 1.2	83.7 ± 1.6
img	96.0 ± 0.3	96.7 ± 0.2	96.9 ± 0.3	96.7 ± 0.3
ionos	88.9 ± 0.8	89.1 ± 0.9	89.7 ± 1.5	88.9 ± 1.5
mok1	100.00 ± 0.00	99.3 ± 0.8	99.4 ± 0.6	98.6 ± 0.9
mok2	79.9 ± 1.7	90.5 ± 1.0	90.6 ± 1.1	92.1 ± 1.2
pima	76.1 ± 1.2	76.8 ± 0.9	77.4 ± 1.0	75.8 ± 0.6
survi	74.1 ± 1.4	74.1 ± 1.5	75.1 ± 1.1	74.4 ± 1.3
vote	96.3 ± 0.5	96.3 ± 0.6	95.6 ± 0.7	96.0 ± 0.7
vowel	86.9 ± 0.5	90.6 ± 0.5	91.1 ± 0.6	91.5 ± 0.7
wdbc	96.8 ± 0.5	97.0 ± 0.5	96.9 ± 0.5	96.7 ± 0.6

Table C.128: Performance - Implementation of *Stacked* by *Ghorbani & Owrangh*

Database	3-expert	9-expert	20-expert	40-expert
aritm	74.9 ± 1.3	74.5 ± 1.0	72.9 ± 1.3	75.6 ± 1.0
bala	96.0 ± 0.8	96.8 ± 0.4	96.9 ± 0.4	95.8 ± 0.6
band	74.4 ± 1.4	76.0 ± 1.4	73.8 ± 1.5	73.5 ± 1.2
bupa	73.0 ± 1.8	71.7 ± 1.8	72.0 ± 1.2	72.6 ± 1.5
cred	86.3 ± 0.8	86.2 ± 0.7	86.2 ± 0.8	85.9 ± 0.7
derma	97.5 ± 0.7	97.6 ± 0.7	97.3 ± 0.6	97.3 ± 0.4
ecoli	86.5 ± 0.9	86.3 ± 0.9	84.4 ± 1.1	86.0 ± 1.1
flare	81.6 ± 0.9	82.1 ± 0.8	81.4 ± 0.9	81.7 ± 0.7
glas	96.2 ± 0.5	96.2 ± 0.9	95.8 ± 0.6	95.4 ± 0.7
hear	83.9 ± 1.3	82.2 ± 1.0	82.4 ± 1.4	82.5 ± 1.5
img	97.0 ± 0.2	96.8 ± 0.3	96.8 ± 0.2	97.0 ± 0.2
ionos	90.0 ± 0.9	91.0 ± 1.0	90.1 ± 1.2	89.3 ± 1.5
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	80.9 ± 1.9	90.9 ± 1.0	90.8 ± 0.9	91.3 ± 1.4
pima	76.4 ± 1.0	76.6 ± 1.0	76.5 ± 1.4	74.5 ± 1.0
survi	73.9 ± 1.1	72.6 ± 1.1	72.8 ± 1.5	73.8 ± 1.0
vote	96.0 ± 0.7	95.6 ± 0.8	95.5 ± 0.4	95.5 ± 0.6
vowel	88.1 ± 0.6	92.2 ± 0.5	92.8 ± 0.7	93.6 ± 0.6
wdbc	96.5 ± 0.5	96.9 ± 0.4	96.5 ± 0.6	96.6 ± 0.6

Table C.129: Performance of Stacked Combiners - *STC - SE - SN*

Database	3-expert	9-expert	20-expert	40-expert
aritm	75.4 ± 1.4	75.1 ± 1.2	73.8 ± 1.3	73.9 ± 1.4
bala	96.7 ± 0.4	96.3 ± 0.5	96.2 ± 0.6	96.1 ± 0.6
band	72.9 ± 1.3	73.1 ± 1.5	72.7 ± 1.1	73.8 ± 0.9
bupa	72.0 ± 1.4	71.7 ± 1.3	71.9 ± 1.4	70.6 ± 1.8
cred	86.0 ± 0.5	86.0 ± 0.5	86.0 ± 0.8	85.8 ± 0.8
derma	97.2 ± 0.7	97.5 ± 0.7	97.5 ± 0.7	97.6 ± 0.7
ecoli	86.6 ± 0.9	86.8 ± 1.1	86.8 ± 0.8	86.8 ± 1.1
flare	81.3 ± 0.6	81.1 ± 0.5	81.4 ± 0.6	81.3 ± 0.7
glas	95.2 ± 0.9	96.0 ± 0.7	96.6 ± 0.8	95.8 ± 0.6
hear	83.4 ± 1.4	83.2 ± 1.0	83.2 ± 1.3	82.7 ± 1.2
img	96.5 ± 0.2	96.6 ± 0.3	97.0 ± 0.2	97.2 ± 0.2
ionos	92.0 ± 0.8	92.9 ± 1.0	92.7 ± 1.1	92.4 ± 1.0
mok1	98.4 ± 0.9	99.8 ± 0.3	100.00 ± 0.00	100.00 ± 0.00
mok2	89 ± 2	92.1 ± 1.2	91.5 ± 1.1	92.4 ± 1.2
pima	76.1 ± 1.0	76.1 ± 1.1	76.4 ± 0.9	75.9 ± 0.9
survi	74.4 ± 1.4	73.8 ± 1.4	73.8 ± 1.3	74.1 ± 1.2
vote	95.5 ± 0.6	95.5 ± 0.6	95.5 ± 0.7	95.5 ± 0.5
vowel	89.4 ± 0.8	92.3 ± 0.5	93.3 ± 0.6	94.2 ± 0.8
wdbc	97.1 ± 0.5	97.2 ± 0.4	97.2 ± 0.5	97.2 ± 0.5

Table C.130: Performance of Stacked Combiners - *STCP - SE - SN*

Database	3-expert	9-expert	20-expert	40-expert
aritm	74.4 ± 1.3	73.6 ± 1.7	74.7 ± 1.1	74.5 ± 1.3
bala	92.7 ± 0.8	96.4 ± 0.5	96.2 ± 0.6	96.2 ± 0.6
band	73.1 ± 1.5	73.3 ± 1.5	73.3 ± 1.0	73.1 ± 0.9
bupa	72.4 ± 1.2	72.0 ± 1.2	72.1 ± 1.2	70 ± 2
cred	85.9 ± 0.8	86.5 ± 0.7	86.2 ± 0.8	85.8 ± 0.7
derma	97.2 ± 0.7	97.3 ± 0.7	97.5 ± 0.7	97.6 ± 0.7
ecoli	86.8 ± 1.0	86.3 ± 1.2	86.8 ± 1.1	86.8 ± 1.0
flare	81.9 ± 0.4	81.7 ± 0.7	81.5 ± 0.7	81.1 ± 0.7
glas	95.6 ± 0.9	95.6 ± 0.8	96.6 ± 0.8	96.6 ± 0.8
hear	82.2 ± 1.2	82.7 ± 1.4	82.2 ± 1.5	82.0 ± 1.4
img	96.7 ± 0.2	96.8 ± 0.3	97.0 ± 0.3	96.8 ± 0.2
ionos	92.0 ± 0.9	92.7 ± 1.0	92.9 ± 1.2	92.4 ± 1.1
mok1	99.8 ± 0.3	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	89 ± 3	91.9 ± 1.3	91.5 ± 1.1	91.4 ± 1.2
pima	76.1 ± 1.0	75.7 ± 1.0	75.9 ± 1.2	75.9 ± 1.0
survi	73.9 ± 1.4	73.6 ± 1.3	73.8 ± 1.2	73.9 ± 1.4
vote	95.5 ± 0.6	95.5 ± 0.6	95.4 ± 0.7	95.4 ± 0.7
vowel	89.8 ± 0.8	92.3 ± 0.6	93.3 ± 0.7	94.1 ± 0.7
wdbc	97.2 ± 0.5	97.4 ± 0.5	97.3 ± 0.5	97.3 ± 0.5

Table C.131: Performance of Stacked Combiners - *STC-Adaboost-MFSN*

Database	3-expert	9-expert	20-expert	40-expert
aritm	74.0 ± 1.3	72.6 ± 1.3	72.9 ± 1.6	73.7 ± 1.3
bala	96.6 ± 0.5	96.6 ± 0.4	96.8 ± 0.3	96.8 ± 0.3
band	70.5 ± 1.7	73.5 ± 1.4	70.7 ± 1.5	73.6 ± 1.6
bupa	70.6 ± 1.9	71.9 ± 1.5	72 ± 2	69.9 ± 1.6
cred	85.8 ± 0.8	83.8 ± 0.8	83.5 ± 0.9	83.7 ± 0.8
derma	97.5 ± 0.7	97.2 ± 0.6	97.5 ± 0.7	97.6 ± 0.7
ecoli	84.3 ± 1.1	84.9 ± 0.8	83.2 ± 0.7	82.5 ± 1.0
flare	81.5 ± 0.6	81.2 ± 0.6	81.1 ± 0.6	81.0 ± 0.6
glas	94.8 ± 1.1	95.4 ± 0.9	96.4 ± 0.8	96.4 ± 0.8
hear	80 ± 2	80 ± 2	79.7 ± 1.5	79.0 ± 1.8
img	96.8 ± 0.2	96.0 ± 0.3	96.1 ± 0.3	96.32 ± 0.19
ionos	89.7 ± 1.2	90.4 ± 1.0	91.6 ± 0.9	91.1 ± 0.9
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	79 ± 2	80 ± 2	83 ± 2	84 ± 2
pima	76.4 ± 1.1	74.9 ± 1.1	73.1 ± 1.2	71.5 ± 1.0
survi	74.3 ± 1.3	74.1 ± 1.3	73.6 ± 1.3	73 ± 2
vote	96.0 ± 0.7	95.8 ± 0.7	95.8 ± 0.7	96.0 ± 0.6
vowel	91.2 ± 0.8	95.6 ± 0.5	96.1 ± 0.6	97.1 ± 0.6
wdbc	96.0 ± 0.5	96.1 ± 0.6	96.5 ± 0.5	96.4 ± 0.6

Table C.132: Performance of Stacked Combiners - *STCP-Adaboost-MFSN*

Database	3-expert	9-expert	20-expert	40-expert
aritm	73.6 ± 1.9	74.3 ± 1.3	73.7 ± 1.4	74.9 ± 1.3
bala	96.6 ± 0.5	96.2 ± 0.5	96.8 ± 0.3	96.7 ± 0.3
band	71.5 ± 1.8	72.7 ± 1.6	72 ± 2	73.6 ± 1.7
bupa	71.1 ± 1.4	71.1 ± 1.5	71 ± 2	69 ± 2
cred	86.1 ± 0.8	85.2 ± 0.9	84.5 ± 0.6	85.8 ± 0.7
derma	98.0 ± 0.6	97.5 ± 0.7	97.3 ± 0.6	97.3 ± 0.5
ecoli	84.7 ± 0.8	82.6 ± 1.6	83.1 ± 1.0	83.2 ± 0.9
flare	81.6 ± 0.6	81.6 ± 0.6	81.3 ± 0.6	81.4 ± 0.6
glas	95.4 ± 0.8	95.2 ± 1.0	96.2 ± 0.9	96.6 ± 0.7
hear	82.2 ± 1.5	81.9 ± 1.8	80 ± 2	81.9 ± 1.7
img	96.9 ± 0.2	97.16 ± 0.18	96.3 ± 0.3	97.2 ± 0.3
ionos	89.9 ± 1.2	90.6 ± 1.0	91.3 ± 0.9	91.3 ± 0.8
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	79 ± 2	81 ± 2	83 ± 2	84 ± 2
pima	76.1 ± 0.9	74.9 ± 1.1	73.4 ± 1.1	71.9 ± 1.0
survi	74.3 ± 1.3	74.3 ± 1.1	73.6 ± 1.1	71.0 ± 1.8
vote	96.1 ± 0.6	96.1 ± 0.7	96.0 ± 0.7	96.0 ± 0.8
vowel	90.7 ± 0.7	95.3 ± 0.7	94.6 ± 0.7	89 ± 9
wdbc	96.1 ± 0.5	96.1 ± 0.6	96.5 ± 0.5	96.4 ± 0.6

Table C.133: Performance of Stacked Combiners - *STC-CVCv3Conserboost-MFSN*

Database	3-expert	9-expert	20-expert	40-expert
aritm	74.8 ± 1.8	74.7 ± 1.4	74.0 ± 1.1	73.6 ± 1.7
bala	96.6 ± 0.4	96.3 ± 0.4	96.3 ± 0.4	96.6 ± 0.4
band	70.2 ± 1.4	71.6 ± 1.4	73.3 ± 1.6	70.5 ± 1.8
bupa	71.0 ± 1.3	71.0 ± 1.8	71.1 ± 1.6	72.1 ± 1.1
cred	85.8 ± 0.9	85.2 ± 0.8	84.6 ± 0.7	84.4 ± 0.7
derma	96.5 ± 0.8	97.3 ± 0.4	97.6 ± 0.5	97.3 ± 0.5
ecoli	84.1 ± 0.8	86.2 ± 0.9	86.0 ± 0.6	84.3 ± 1.1
flare	81.4 ± 0.6	81.5 ± 0.7	81.0 ± 0.6	80.3 ± 0.7
glas	95.2 ± 0.7	96.2 ± 0.9	96.4 ± 0.9	96.6 ± 0.8
hear	83.1 ± 1.5	83.4 ± 1.3	82.5 ± 1.3	81.7 ± 1.4
img	97.0 ± 0.3	97.51 ± 0.16	97.59 ± 0.17	97.72 ± 0.16
ionos	92.0 ± 1.1	90.9 ± 0.8	91.7 ± 0.6	92.4 ± 0.5
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	80.0 ± 1.5	89.8 ± 1.2	91.9 ± 1.5	92.5 ± 1.3
pima	76.1 ± 1.0	74.6 ± 1.1	74.3 ± 1.1	73.7 ± 1.2
survi	74.8 ± 1.7	73.1 ± 0.9	71.6 ± 1.5	70.0 ± 1.7
vote	96.0 ± 0.7	96.0 ± 0.6	95.6 ± 0.8	95.5 ± 0.5
vowel	91.3 ± 0.6	96.7 ± 0.5	97.5 ± 0.4	98.1 ± 0.4
wdbc	96.5 ± 0.4	96.4 ± 0.5	96.1 ± 0.5	96.5 ± 0.5

Table C.134: Performance of Stacked Combiners - *STCP-CVCv3Conserboost-MFSN*

Database	3-expert	9-expert	20-expert	40-expert
aritm	76.0 ± 1.5	74.6 ± 1.4	73.9 ± 1.2	72.9 ± 1.6
bala	96.3 ± 0.4	96.4 ± 0.4	96.2 ± 0.5	96.8 ± 0.4
band	71.3 ± 1.6	72.2 ± 1.6	73.3 ± 1.8	71.8 ± 1.4
bupa	70.9 ± 1.8	71.6 ± 1.7	71.9 ± 1.6	71.1 ± 1.4
cred	85.5 ± 0.9	85.8 ± 0.7	84.8 ± 0.6	84.4 ± 0.7
derma	97.2 ± 0.6	97.5 ± 0.5	97.6 ± 0.5	97.3 ± 0.5
ecoli	85.9 ± 0.8	86.5 ± 0.9	86.9 ± 0.7	83.8 ± 1.1
flare	81.3 ± 0.6	81.4 ± 0.5	81.1 ± 0.6	80.7 ± 0.7
glas	96.0 ± 0.7	96.2 ± 0.9	96.4 ± 0.9	96.6 ± 0.8
hear	83.1 ± 1.7	83.4 ± 1.3	82.2 ± 1.2	81.4 ± 1.5
img	97.0 ± 0.2	97.46 ± 0.18	97.63 ± 0.18	97.74 ± 0.17
ionos	90.1 ± 0.9	91.9 ± 0.6	91.9 ± 0.6	92.6 ± 0.6
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	80.3 ± 1.7	89.6 ± 1.2	92.1 ± 1.6	93.0 ± 1.3
pima	75.9 ± 1.0	75.2 ± 1.2	74.3 ± 1.0	73.9 ± 1.1
survi	74.1 ± 1.4	73.1 ± 0.9	72.1 ± 1.5	71.0 ± 1.8
vote	96.4 ± 0.6	96.1 ± 0.5	95.9 ± 0.6	95.5 ± 0.5
vowel	91.5 ± 0.7	96.5 ± 0.4	96.8 ± 0.4	98.4 ± 0.3
wdbc	96.5 ± 0.4	96.2 ± 0.4	96.1 ± 0.5	96.5 ± 0.5

Table C.135: Performance of Stacked Combiners - *STC-Inverse-MFSN*

Database	3-expert	9-expert	20-expert	40-expert
aritm	74.7 ± 1.4	76.6 ± 1.6	74.8 ± 1.2	75.4 ± 1.7
bala	95.8 ± 0.6	96.8 ± 0.4	96.6 ± 0.5	96.4 ± 0.5
band	71.6 ± 0.9	72.2 ± 0.9	71.5 ± 1.1	72.4 ± 1.2
bupa	73.1 ± 1.5	72.6 ± 1.7	71.1 ± 1.4	71.3 ± 1.4
cred	86.2 ± 0.7	86.5 ± 0.6	86.2 ± 0.6	86.2 ± 0.9
derma	94.8 ± 1.3	96.8 ± 0.7	96.9 ± 0.7	96.5 ± 1.0
ecoli	84.1 ± 0.8	85.6 ± 0.9	85.7 ± 0.9	84.4 ± 1.4
flare	81.1 ± 0.7	81.1 ± 0.7	81.4 ± 0.7	81.3 ± 0.7
glas	92.6 ± 1.1	93.2 ± 1.2	93.4 ± 1.0	93.2 ± 1.0
hear	83.1 ± 1.7	82.4 ± 1.7	82.9 ± 1.5	82.0 ± 1.5
img	95.72 ± 0.20	96.2 ± 0.2	95.9 ± 0.2	96.0 ± 0.3
ionos	89.3 ± 1.0	88.6 ± 0.9	88.9 ± 1.1	88.7 ± 0.9
mok1	98.8 ± 0.9	98.8 ± 0.9	100.00 ± 0.00	100.00 ± 0.00
mok2	77 ± 2	77.0 ± 2.0	75 ± 2	76 ± 2
pima	75.7 ± 1.3	75.7 ± 0.9	75.7 ± 1.0	76.2 ± 0.8
survi	73.4 ± 0.9	73.4 ± 1.0	73.3 ± 1.0	73.8 ± 0.9
vote	96.1 ± 0.7	95.8 ± 0.7	95.9 ± 0.6	95.5 ± 0.7
vowel	84.9 ± 0.9	87.2 ± 0.8	89.2 ± 0.7	90.7 ± 0.7
wdbc	96.5 ± 0.5	97.0 ± 0.4	96.7 ± 0.3	96.5 ± 0.4

Table C.136: Performance of Stacked Combiners - *STCP-Inverse-MFSN*

Database	3-expert	9-expert	20-expert	40-expert
aritm	73.9 ± 1.4	75.5 ± 1.5	75.3 ± 1.2	74.5 ± 1.6
bala	95.0 ± 0.8	96.3 ± 0.4	96.7 ± 0.4	96.7 ± 0.5
band	72.9 ± 1.2	72.0 ± 1.2	72.7 ± 1.5	72.0 ± 1.4
bupa	71.3 ± 1.2	71.0 ± 1.1	72.6 ± 1.0	71.0 ± 1.4
cred	86.0 ± 0.7	87.0 ± 0.6	86.4 ± 0.6	86.5 ± 0.6
derma	97.0 ± 0.8	96.9 ± 0.8	97.0 ± 0.6	97.2 ± 0.7
ecoli	84.1 ± 0.8	84.4 ± 1.0	84.9 ± 0.5	85.0 ± 0.9
flare	81.7 ± 0.7	81.9 ± 0.6	81.3 ± 0.7	81.5 ± 0.7
glas	95.0 ± 0.8	95.0 ± 1.2	93.4 ± 1.3	93.8 ± 1.1
hear	82.7 ± 1.5	83.7 ± 1.4	83.1 ± 1.5	81.2 ± 1.6
img	96.3 ± 0.2	96.1 ± 0.3	96.5 ± 0.3	96.1 ± 0.3
ionos	90.0 ± 1.0	90.1 ± 1.2	90.0 ± 1.3	89.3 ± 0.8
mok1	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
mok2	76 ± 2	77 ± 2	78 ± 2	78.4 ± 2.0
pima	75.5 ± 1.1	75.9 ± 1.1	76.2 ± 1.2	76.2 ± 1.4
survi	74.4 ± 1.1	73.6 ± 1.0	73.8 ± 1.4	73.1 ± 1.5
vote	95.9 ± 0.5	96.3 ± 0.6	95.8 ± 0.6	96.4 ± 0.6
vowel	89.1 ± 1.0	85.9 ± 0.5	90.4 ± 0.8	89.6 ± 0.8
wdbc	96.5 ± 0.5	97.2 ± 0.4	97.3 ± 0.2	96.5 ± 0.4

Table C.137: Performance of Stacked Combiners - *STC-RBFSE-MFSN*

Database	3-expert	9-expert	20-expert	40-expert
aritm	75.3 ± 1.1	76.7 ± 1.4	75.4 ± 1.4	75.3 ± 1.5
bala	92.4 ± 0.9	93.2 ± 0.9	92.8 ± 0.8	92.1 ± 0.8
band	72.7 ± 1.3	74.2 ± 1.2	74.0 ± 1.4	74.2 ± 1.2
bupa	72.0 ± 1.4	70.9 ± 1.3	71.3 ± 1.4	71.6 ± 1.3
cred	86.6 ± 0.5	87.0 ± 0.5	86.2 ± 0.7	86.9 ± 0.5
derma	96.8 ± 0.6	97.2 ± 0.5	97.3 ± 0.4	97.5 ± 0.4
ecoli	85.0 ± 1.4	85.7 ± 1.1	85.7 ± 1.0	85.3 ± 1.2
flare	81.5 ± 0.6	81.6 ± 0.8	81.9 ± 0.8	81.1 ± 0.6
glas	92.2 ± 1.0	92.6 ± 1.0	92.2 ± 1.1	92.6 ± 1.2
hear	84.4 ± 1.5	84.1 ± 1.5	82.4 ± 1.7	81.5 ± 1.7
img	96.5 ± 0.4	96.6 ± 0.4	96.5 ± 0.3	96.8 ± 0.3
ionos	93.4 ± 1.0	93.9 ± 1.0	93.4 ± 1.0	92.9 ± 1.1
mok1	99.88 ± 0.13	99.8 ± 0.3	99.8 ± 0.3	99.8 ± 0.3
mok2	91.0 ± 1.1	91.5 ± 1.3	91.5 ± 1.2	91.6 ± 1.3
pima	77.2 ± 1.0	76.6 ± 0.9	75.1 ± 1.2	75.7 ± 0.9
survi	74.8 ± 1.2	75.1 ± 1.2	74.6 ± 1.0	74.3 ± 1.3
vote	95.6 ± 0.8	95.9 ± 0.6	96.3 ± 0.8	96.0 ± 0.6
vowel	96.7 ± 0.5	97.0 ± 0.3	95.9 ± 0.3	95.6 ± 0.5
wdbc	97.4 ± 0.3	97.4 ± 0.3	97.3 ± 0.3	97.3 ± 0.3

Table C.138: Performance of Stacked Combiners - *STCP-RBFSE-MFSN*

Database	3-expert	9-expert	20-expert	40-expert
aritm	75.3 ± 1.0	74.5 ± 1.0	74.5 ± 1.3	75.9 ± 1.2
bala	93.8 ± 0.8	93.5 ± 0.8	93.4 ± 0.8	93.1 ± 0.8
band	73.8 ± 1.3	74.2 ± 1.3	74.4 ± 1.2	74.9 ± 1.1
bupa	70.4 ± 1.2	70.6 ± 1.5	69.0 ± 1.7	69.6 ± 1.5
cred	86.3 ± 0.7	86.2 ± 0.7	86.2 ± 0.8	86.8 ± 0.5
derma	97.0 ± 0.4	97.3 ± 0.5	97.3 ± 0.5	97.3 ± 0.4
ecoli	84.3 ± 1.0	84.9 ± 0.9	84.6 ± 0.9	84.6 ± 0.9
flare	81.3 ± 0.6	81.3 ± 0.6	81.9 ± 0.6	82.0 ± 0.6
glas	90.8 ± 1.0	91.8 ± 1.2	91.8 ± 1.1	92.0 ± 1.2
hear	82.7 ± 1.4	83.2 ± 1.5	83.4 ± 1.4	82.7 ± 1.6
img	96.5 ± 0.3	96.5 ± 0.3	96.8 ± 0.3	96.6 ± 0.4
ionos	92.0 ± 0.9	92.9 ± 1.0	92.4 ± 0.9	92.4 ± 0.9
mok1	99.8 ± 0.3	99.8 ± 0.3	99.8 ± 0.3	99.8 ± 0.3
mok2	91.3 ± 1.2	91.4 ± 1.3	91.5 ± 1.2	91.8 ± 1.2
pima	75.9 ± 1.0	76.2 ± 1.1	76.8 ± 0.9	76.9 ± 1.2
survi	74.8 ± 1.2	75.4 ± 1.4	74.4 ± 1.1	73.6 ± 1.5
vote	94.8 ± 0.7	95.8 ± 0.5	95.8 ± 0.5	95.8 ± 0.6
vowel	96.6 ± 0.4	97.2 ± 0.3	97.0 ± 0.2	96.6 ± 0.5
wdbc	97.3 ± 0.4	97.2 ± 0.3	97.3 ± 0.3	97.3 ± 0.3

Table C.139: Performance of Stacked Combiners - *STC-RBFSE$_{threshold}$-MFSN*

Database	3-expert	9-expert	20-expert	40-expert
aritm	75.1 ± 1.1	76.4 ± 1.4	75.3 ± 1.5	75.4 ± 1.5
bala	92.4 ± 0.9	92.8 ± 0.9	92.7 ± 0.8	93.4 ± 0.7
band	72.7 ± 1.3	73.6 ± 1.2	74.2 ± 1.2	74.0 ± 1.2
bupa	71.9 ± 1.5	71.9 ± 1.2	71.6 ± 1.4	71.3 ± 1.4
cred	86.8 ± 0.5	86.8 ± 0.5	86.5 ± 0.6	86.4 ± 0.8
derma	96.5 ± 0.6	97.3 ± 0.4	97.6 ± 0.4	97.5 ± 0.4
ecoli	85.0 ± 1.0	86.2 ± 1.2	85.1 ± 1.2	85.4 ± 1.1
flare	81.5 ± 0.6	81.6 ± 0.8	81.2 ± 0.5	81.1 ± 0.5
glas	92.2 ± 0.9	92.6 ± 1.0	92.4 ± 1.1	92.8 ± 1.3
hear	84.4 ± 1.5	83.7 ± 1.5	82.4 ± 1.4	82.5 ± 1.6
img	96.6 ± 0.4	96.6 ± 0.4	96.8 ± 0.3	96.6 ± 0.3
ionos	93.7 ± 1.0	93.7 ± 0.9	93.7 ± 1.1	92.9 ± 1.1
mok1	99.88 ± 0.13	99.8 ± 0.3	99.8 ± 0.3	99.8 ± 0.3
mok2	91.0 ± 1.0	91.9 ± 1.2	91.6 ± 1.2	91.6 ± 1.2
pima	77.1 ± 0.9	77.6 ± 0.8	76.0 ± 0.9	76.8 ± 0.9
survi	74.9 ± 1.3	73.6 ± 1.5	74.9 ± 1.0	74.1 ± 1.7
vote	95.8 ± 0.5	95.9 ± 0.6	95.8 ± 0.6	95.4 ± 0.7
vowel	97.0 ± 0.4	97.2 ± 0.4	97.1 ± 0.4	96.3 ± 0.5
wdbc	97.4 ± 0.3	97.4 ± 0.3	97.4 ± 0.3	97.3 ± 0.3

Table C.140: Performance of Stacked Combiners - *STCP-RBFSE$_{threshold}$-MFSN*

Database	3-expert	9-expert	20-expert	40-expert
aritm	74.4 ± 1.2	74.0 ± 1.5	76.1 ± 1.3	75.6 ± 0.6
bala	93.2 ± 0.9	93.6 ± 0.7	93.1 ± 0.7	93.1 ± 0.8
band	73.1 ± 1.8	74.5 ± 1.2	73.8 ± 1.2	74.9 ± 1.2
bupa	69.9 ± 1.4	70.3 ± 1.6	68.6 ± 1.6	69.7 ± 1.4
cred	86.3 ± 0.8	86.5 ± 0.8	86.4 ± 0.8	86.7 ± 0.6
derma	97.3 ± 0.5	97.2 ± 0.6	97.3 ± 0.4	97.5 ± 0.4
ecoli	85.6 ± 0.9	84.9 ± 1.1	84.6 ± 1.2	85.1 ± 1.2
flare	82.0 ± 0.6	81.3 ± 0.5	81.9 ± 0.6	81.9 ± 0.6
glas	91.8 ± 1.2	92.2 ± 1.3	91.8 ± 1.1	92.2 ± 1.0
hear	82.7 ± 1.6	83.9 ± 1.3	83.7 ± 1.2	82.7 ± 1.6
img	96.5 ± 0.3	96.6 ± 0.3	96.7 ± 0.3	96.6 ± 0.4
ionos	92.1 ± 0.9	92.7 ± 0.9	91.9 ± 1.0	92.4 ± 0.8
mok1	99.8 ± 0.3	99.8 ± 0.3	99.8 ± 0.3	99.8 ± 0.3
mok2	91.0 ± 1.3	91.4 ± 1.2	91.8 ± 1.2	91.6 ± 1.2
pima	76.9 ± 0.9	75.9 ± 0.9	75.0 ± 1.1	75.4 ± 1.2
survi	74.9 ± 1.2	75.6 ± 1.4	75.1 ± 1.3	74.6 ± 1.1
vote	94.9 ± 0.7	96.0 ± 0.4	96.0 ± 0.5	95.8 ± 0.7
vowel	97.2 ± 0.3	96.6 ± 0.4	96.9 ± 0.3	97.0 ± 0.3
wdbc	97.4 ± 0.4	97.2 ± 0.3	97.2 ± 0.3	97.2 ± 0.3

Table C.141: Performance of Stacked Combiners - *STC-RBFSE$_{min-max}$-MFSN*

Database	3-expert	9-expert	20-expert	40-expert
aritm	74.3 ± 1.7	74.4 ± 0.8	75.5 ± 1.1	74.8 ± 1.3
bala	92.8 ± 0.7	92.6 ± 1.1	92.6 ± 0.9	92.6 ± 0.9
band	69.6 ± 2.0	67.8 ± 1.7	71.5 ± 1.7	72.0 ± 2.0
bupa	71.3 ± 1.1	71.9 ± 1.8	71.4 ± 1.7	70.0 ± 1.4
cred	86.5 ± 0.7	86.8 ± 0.9	87.3 ± 0.7	87.0 ± 0.6
derma	96.8 ± 0.5	97.2 ± 0.4	97.3 ± 0.4	97.5 ± 0.4
ecoli	86.3 ± 0.9	85.3 ± 1.1	84.9 ± 0.7	85.6 ± 1.2
flare	82.8 ± 0.6	82.3 ± 0.6	82.0 ± 0.8	82.3 ± 0.8
glas	93.2 ± 1.0	93.2 ± 1.0	91.8 ± 1.2	93.6 ± 1.0
hear	82.2 ± 1.8	81.9 ± 1.3	82.0 ± 1.6	81.5 ± 1.7
img	96.6 ± 0.3	96.7 ± 0.3	96.6 ± 0.4	96.7 ± 0.3
ionos	91.0 ± 0.9	91.3 ± 0.9	89.6 ± 1.2	90.4 ± 0.9
mok1	99.8 ± 0.3	99.4 ± 0.5	99.6 ± 0.4	99.6 ± 0.4
mok2	89.5 ± 0.9	89.6 ± 1.1	88.8 ± 1.3	88.3 ± 1.4
pima	74.5 ± 1.3	76.0 ± 1.0	75.7 ± 0.9	75.2 ± 1.1
survi	74.6 ± 1.1	73.6 ± 0.8	74.8 ± 1.7	75.1 ± 1.3
vote	95.3 ± 0.7	95.8 ± 0.6	95.8 ± 0.6	95.4 ± 0.6
vowel	96.8 ± 0.4	97.0 ± 0.3	96.9 ± 0.4	96.4 ± 0.4
wdbc	97.4 ± 0.4	97.2 ± 0.4	96.7 ± 0.4	96.9 ± 0.4

Table C.142: Performance of Stacked Combiners - *STCP-RBFSE$_{min-max}$-MFSN*

Database	3-net	9-net	20-net	40-net
aritm	74.7 ± 1.1	73.7 ± 1.2	75.2 ± 1.5	74.6 ± 1.4
bala	92.2 ± 0.8	93.0 ± 0.8	92.7 ± 1.0	93.0 ± 0.8
band	69.1 ± 1.6	70.5 ± 1.5	70.4 ± 1.8	73.8 ± 1.7
bupa	70.9 ± 1.4	70.7 ± 1.5	72.0 ± 1.4	70.7 ± 1.5
cred	85.4 ± 1.4	86.9 ± 0.9	87.2 ± 0.7	87.2 ± 0.7
derma	97.2 ± 0.4	97.3 ± 0.3	97.3 ± 0.4	97.3 ± 0.4
ecoli	85.9 ± 0.8	83.8 ± 1.2	86.3 ± 0.9	85.0 ± 1.1
flare	81.7 ± 0.7	82.0 ± 0.7	82.0 ± 0.6	81.7 ± 0.6
glas	93.2 ± 1.2	92.2 ± 1.1	93.2 ± 1.0	92.8 ± 1.0
hear	82.4 ± 1.2	81.4 ± 1.8	80.8 ± 1.8	79.2 ± 1.7
img	96.7 ± 0.3	96.7 ± 0.3	96.6 ± 0.4	96.9 ± 0.4
ionos	92.0 ± 0.8	91.0 ± 1.3	90.9 ± 1.0	90.1 ± 1.2
mok1	99.6 ± 0.3	99.5 ± 0.5	99.8 ± 0.3	99.8 ± 0.3
mok2	89.8 ± 0.9	89.5 ± 1.0	89.1 ± 1.4	89.6 ± 1.5
pima	74.6 ± 1.2	75.9 ± 0.8	76.4 ± 0.9	74.8 ± 1.0
survi	72.1 ± 1.2	73.8 ± 1.1	74.6 ± 0.9	74.4 ± 1.4
vote	95.8 ± 0.7	96.0 ± 0.6	95.1 ± 0.7	96.8 ± 0.7
vowel	96.7 ± 0.4	97.3 ± 0.3	96.9 ± 0.4	97.2 ± 0.3
wdbc	97.2 ± 0.3	97.1 ± 0.4	96.8 ± 0.3	97.0 ± 0.4

Table C.143: Performance of Stacked Combiners - *STCP-RBFSE$_{sum}$-MFSN*

Database	3-expert	9-expert	20-expert	40-expert
aritm	74.9 ± 1.7	75.9 ± 1.3	76.4 ± 1.4	75.7 ± 1.3
bala	92.3 ± 1.0	92.6 ± 0.8	92.9 ± 0.7	93.2 ± 0.8
band	73.1 ± 1.2	74.2 ± 1.3	74.5 ± 1.2	74.2 ± 1.3
bupa	71.9 ± 1.2	72.7 ± 1.2	71.4 ± 1.4	71.0 ± 1.4
cred	86.8 ± 0.6	86.8 ± 0.7	86.2 ± 0.6	86.5 ± 0.7
derma	96.9 ± 0.5	96.9 ± 0.6	97.0 ± 0.5	97.3 ± 0.5
ecoli	86.9 ± 1.1	86.2 ± 1.1	86.0 ± 1.1	84.6 ± 0.8
flare	81.3 ± 0.6	81.9 ± 0.6	81.8 ± 0.7	81.3 ± 0.7
glas	93.2 ± 1.1	93.4 ± 1.0	93.4 ± 1.2	93.4 ± 1.0
hear	83.4 ± 1.6	83.7 ± 1.5	83.1 ± 1.6	83.2 ± 1.5
img	96.8 ± 0.4	97.0 ± 0.3	96.7 ± 0.4	96.7 ± 0.3
ionos	90.6 ± 1.2	91.7 ± 1.4	90.6 ± 1.3	90.4 ± 0.9
mok1	99.88 ± 0.13	99.8 ± 0.3	99.8 ± 0.3	99.8 ± 0.3
mok2	90.4 ± 1.0	91.3 ± 1.2	91.4 ± 1.3	91.0 ± 1.4
pima	76.8 ± 1.0	75.7 ± 0.7	76.3 ± 0.8	75.0 ± 1.0
survi	75.1 ± 1.4	75.6 ± 1.5	75.1 ± 1.1	75.4 ± 1.1
vote	95.9 ± 0.5	95.5 ± 0.6	95.8 ± 0.6	96.1 ± 0.6
vowel	96.6 ± 0.4	95.7 ± 0.5	97.0 ± 0.3	96.0 ± 0.4
wdbc	97.4 ± 0.3	97.4 ± 0.3	97.4 ± 0.3	97.3 ± 0.3

Table C.144: Performance of Stacked Combiners - *STCP-RBFSE$_{sum}$-MFSN*

Database	3-expert	9-expert	20-expert	40-expert
aritm	74.6 ± 1.8	74.5 ± 1.1	76.8 ± 1.0	76.1 ± 1.1
bala	92.8 ± 0.9	93.1 ± 0.8	93.4 ± 0.8	93.1 ± 0.7
band	73.8 ± 1.4	74.2 ± 1.3	74.5 ± 1.5	75.3 ± 1.0
bupa	71.3 ± 1.2	70.0 ± 1.6	70.3 ± 1.6	70.0 ± 1.5
cred	86.3 ± 0.8	86.2 ± 0.8	86.6 ± 0.8	85.7 ± 1.1
derma	96.5 ± 0.9	97.5 ± 0.4	97.5 ± 0.4	97.3 ± 0.4
ecoli	84.4 ± 0.7	85.6 ± 1.4	84.1 ± 0.8	83.8 ± 1.2
flare	81.9 ± 0.7	81.6 ± 0.5	81.4 ± 0.6	81.8 ± 0.6
glas	94.0 ± 1.0	93.6 ± 0.9	93.8 ± 1.3	93.4 ± 1.3
hear	82.9 ± 1.5	83.4 ± 1.6	83.6 ± 1.4	82.7 ± 1.5
img	96.7 ± 0.4	96.7 ± 0.4	96.7 ± 0.4	96.5 ± 0.3
ionos	90.4 ± 1.3	90.6 ± 1.4	90.1 ± 1.2	91.3 ± 1.1
mok1	99.8 ± 0.3	99.8 ± 0.3	99.8 ± 0.3	99.8 ± 0.3
mok2	90.6 ± 1.0	91.5 ± 1.0	91.3 ± 1.3	90.9 ± 1.3
pima	75.5 ± 1.0	76.1 ± 1.1	76.0 ± 0.9	76.8 ± 0.9
survi	75.1 ± 1.8	75.1 ± 1.6	75.2 ± 1.5	75.1 ± 1.3
vote	95.3 ± 0.4	96.0 ± 0.4	95.9 ± 0.5	95.9 ± 0.6
vowel	97.0 ± 0.4	96.8 ± 0.3	96.4 ± 0.5	96.2 ± 0.4
wdbc	97.4 ± 0.4	97.2 ± 0.3	97.3 ± 0.3	97.2 ± 0.3

Table C.145: Performance of Stacked Combiners - *STC-MFSE-MFSE*

	3-expert			9-expert			20-expert			40-expert		
Database	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**
aritm	75.3±1.4	75.4±1.5	75.3±1.4	75.6±1.4	75.6±1.4	75.6±1.4	73.8±1.3	73.8±1.3	73.8±1.3	74.5±1.4	74.4±1.4	74.3±1.4
bala	96.7±0.4	96.6±0.4	96.6±0.4	96.2±0.6	96.4±0.5	96.5±0.4	96.1±0.7	96.2±0.6	96.2±0.6	95.9±0.7	95.9±0.7	95.9±0.7
band	72.7±1.2	72.7±1.4	72.7±1.4	73.1±1.5	73.1±1.5	73.1±1.5	72.5±1.1	72.5±1.1	72.7±1.1	73.3±0.9	73.6±0.9	73.6±0.8
bupa	72.0±1.4	72.0±1.4	72.0±1.4	72.1±1.3	72.1±1.3	71.9±1.4	71.9±1.4	71.9±1.4	71.9±1.4	71.1±1.7	70.9±1.9	70.4±1.8
cred	86.0±0.5	86.0±0.5	86.0±0.5	86.5±0.8	86.4±0.8	86.4±0.8	85.8±0.8	85.8±0.8	85.8±0.8	85.9±0.8	85.9±0.8	85.9±0.8
derma	97.0±0.8	97.2±0.7	97.2±0.7	97.5±0.7	97.5±0.7	97.5±0.7	97.5±0.7	97.5±0.7	97.3±0.7	97.6±0.7	97.6±0.7	97.6±0.7
ecoli	86.5±0.9	86.6±0.9	86.3±0.9	86.9±1.0	86.9±1.1	86.8±1.2	86.8±1.0	86.5±1.0	86.6±1.0	85.9±1.2	85.9±1.1	85.7±1.1
flare	81.3±0.6	81.3±0.6	81.3±0.6	81.1±0.5	81.1±0.5	81.1±0.5	81.4±0.7	81.4±0.7	81.4±0.7	81.3±0.7	81.3±0.7	81.3±0.7
glas	95.2±0.8	95.8±0.9	95.6±0.9	96.0±0.7	96.0±0.7	96.2±0.7	96.4±0.8	96.4±0.8	96.2±0.8	96.4±0.8	96.2±0.8	96.4±0.8
hear	83.6±1.3	83.6±1.3	83.6±1.3	83.2±1.0	83.2±1.0	83.2±1.0	83.1±1.3	83.1±1.4	83.2±1.4	82.7±1.2	82.7±1.2	82.7±1.2
img	96.5±0.2	96.5±0.2	96.5±0.2	96.8±0.3	96.8±0.3	96.8±0.3	97.1±0.3	97.1±0.3	97.1±0.3	97.20±0.19	97.18±0.19	97.18±0.19
ionos	92.0±0.8	92.0±0.8	92.0±0.8	92.9±1.0	92.9±1.0	92.9±1.0	92.7±1.1	92.7±1.1	92.7±1.1	92.4±1.0	92.4±1.0	92.4±1.0
mok1	98.4±0.9	98.4±0.9	98.4±0.9	99.8±0.3	99.8±0.3	99.8±0.3	100±0	100±0	100±0	100±0	100±0	100±0
mok2	89± 2	89± 2	89± 2	91.8±1.3	91.8±1.3	91.8±1.3	91.4±1.1	91.5±1.1	91.5±1.1	92.4±1.2	92.4±1.2	92.4±1.2
pima	76.2±1.0	76.1±1.1	76.2±1.0	76.2±1.0	76.1±1.0	76.3±1.0	76.3±0.9	76.2±1.0	76.3±0.9	76.0±0.9	76.1±0.9	76.1±0.9
survi	74.3±1.4	74.3±1.4	74.3±1.4	73.8±1.4	73.8±1.4	73.8±1.4	73.8±1.3	73.8±1.3	73.8±1.3	74.1±1.2	74.1±1.2	74.1±1.2
vote	95.5±0.6	95.5±0.6	95.5±0.6	95.5±0.6	95.5±0.6	95.5±0.6	95.5±0.7	95.5±0.7	95.5±0.7	95.5±0.5	95.5±0.5	95.5±0.5
vowel	89.5±0.5	89.7±0.6	89.6±0.6	92.3±0.7	92.5±0.7	92.6±0.6	93.3±0.7	93.3±0.7	93.2±0.7	94.3±0.7	94.4±0.8	94.3±0.8
wdbc	97.1±0.5	97.1±0.5	97.1±0.5	97.2±0.4	97.2±0.4	97.2±0.4	97.2±0.5	97.2±0.5	97.2±0.5	97.2±0.5	97.2±0.5	97.2±0.5

Table C.146: Performance of Stacked Combiners - *STCP-MFSE-MFSE*

	3-expert			9-expert			20-expert			40-expert		
Database	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**
aritm	74.7±1.2	74.6±1.3	74.6±1.3	74.5±1.3	74.3±1.3	74.4±1.2	75.3±1.1	75.1±1.2	75.1±1.2	74.4±1.3	74.6±1.3	74.7±1.4
bala	92.2±0.8	92.3±0.8	92.3±0.8	96.2±0.5	96.4±0.5	96.4±0.5	96.2±0.6	96.2±0.6	96.2±0.6	96.2±0.6	96.2±0.6	96.2±0.6
band	72.7±1.4	72.7±1.4	72.7±1.4	73.5±1.5	73.6±1.5	73.5±1.5	73.1±1.1	73.1±1.1	72.9±1.2	73.1±0.9	73.1±0.9	73.1±0.9
bupa	72.6±1.3	72.6±1.3	72.6±1.3	72.1±1.2	72.0±1.2	72.1±1.2	72.0±1.2	72.0±1.2	72.0±1.2	70.4±1.8	71.0±1.7	70.6±1.8
cred	86.2±0.8	86.2±0.8	86.1±0.9	86.5±0.7	86.5±0.7	86.5±0.7	86.2±0.7	86.3±0.7	86.0±0.7	85.8±0.8	85.7±0.8	85.7±0.8
derma	97.3±0.7	97.3±0.7	97.2±0.7	97.3±0.7	97.5±0.7	97.5±0.7	97.3±0.7	97.3±0.7	97.5±0.7	97.6±0.7	97.6±0.7	97.6±0.7
ecoli	86.5±1.0	86.3±0.9	86.3±0.9	86.8±1.1	86.8±1.1	86.5±1.1	86.6±0.9	86.5±1.0	86.6±0.9	86.2±1.0	86.6±1.0	86.6±0.8
flare	82.0±0.4	82.0±0.4	82.0±0.4	81.3±0.7	81.4±0.7	81.5±0.6	81.6±0.7	81.5±0.7	81.5±0.7	81.1±0.7	81.0±0.7	81.0±0.6
glas	96.0±0.9	95.6±0.9	95.8±0.9	96.0±0.7	95.8±0.8	95.8±0.8	96.6±0.8	96.6±0.8	96.6±0.8	96.2±0.7	96.2±0.7	96.2±0.7
hear	82.5±1.2	82.2±1.2	82.2±1.2	82.4±1.5	82.2±1.5	82.5±1.5	82.2±1.4	82.4±1.5	82.4±1.5	82.0±1.4	82.2±1.3	82.4±1.3
img	96.6±0.2	96.6±0.2	96.6±0.2	96.9±0.3	96.8±0.3	96.8±0.3	97.1±0.3	97.1±0.3	97.1±0.2	97.0±0.2	97.0±0.2	97.05±0.20
ionos	92.0±0.9	92.0±0.9	92.0±0.9	92.7±1.0	92.7±1.0	92.7±1.0	92.9±1.2	92.9±1.2	92.9±1.2	92.6±1.1	92.6±1.1	92.6±1.1
mok1	99.1±0.9	99.1±0.9	99.8±0.3	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0
mok2	89± 2	89± 2	89± 2	91.9±1.3	91.9±1.3	91.9±1.3	91.4±1.2	91.5±1.1	91.4±1.2	91.5±1.3	91.4±1.3	91.5±1.3
pima	76.1±1.0	76.1±1.0	76.1±1.0	75.9±1.0	76.0±1.1	76.0±1.1	75.7±1.2	75.8±1.2	75.8±1.2	76.1±1.1	76.1±1.1	76.2±1.1
survi	73.9±1.2	73.9±1.2	74.1±1.3	73.8±1.3	73.8±1.3	73.8±1.3	73.9±1.3	73.9±1.3	74.1±1.4	73.8±1.1	73.6±1.2	73.9±1.1
vote	95.5±0.6	95.5±0.6	95.5±0.6	95.5±0.6	95.5±0.6	95.5±0.6	95.4±0.7	95.4±0.7	95.4±0.7	95.4±0.7	95.4±0.7	95.4±0.7
vowel	89.8±0.8	89.8±0.8	89.9±0.7	92.5±0.6	92.6±0.5	92.5±0.6	93.5±0.7	93.4±0.7	93.4±0.7	94.4±0.8	94.4±0.7	94.5±0.8
wdbc	97.3±0.5	97.3±0.5	97.3±0.5	97.4±0.5	97.4±0.5	97.4±0.5	97.3±0.5	97.3±0.5	97.3±0.5	97.3±0.5	97.3±0.5	97.3±0.5

Table C.147: Performance of Stacked Combiners - *STC-MFSE-CVCv3ACB* and *Output Average*

	3-expert			9-expert			20-expert			40-expert		
Database	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**
aritm	75.1±1.6	75.1±1.5	75.6±1.5	75.2±1.4	74.8±1.4	74.5±1.2	74.3±1.0	74.6±1.3	74.6±1.4	74.6±1.2	74.5±1.4	74.1±1.2
bala	96.9±0.3	96.4±0.5	96.2±0.5	96.5±0.5	96.8±0.4	96.7±0.4	96.6±0.6	96.5±0.5	96.6±0.5	96.2±0.7	96.4±0.6	96.2±0.6
band	72.0±1.8	72.9±1.8	72.0±1.9	72.9±1.7	72.9±1.9	73.1±1.8	72.4±1.4	73.1±1.6	72.4±1.4	71.3±1.5	70.5±1.6	71.1±1.5
bupa	71.6±1.7	71.4±1.6	71.4±1.5	71.4±1.4	71.0±1.5	70.4±1.5	70.6±1.8	71.1±1.6	70.4±1.6	71.0±1.7	70.9±1.6	70.4±1.7
cred	86.4±0.9	86.2±0.9	86.1±0.9	86.5±0.9	86.2±0.9	86.2±0.9	85.9±1.0	85.7±1.0	85.6±1.0	85.7±0.9	85.5±1.0	85.5±1.0
derma	97.5±0.7	97.6±0.6	97.6±0.6	97.3±0.6	97.3±0.7	96.9±0.7	97.3±0.7	97.9±0.5	97.3±0.6	97.6±0.6	97.9±0.5	97.7±0.5
ecoli	85.7±1.1	86.5±1.0	86.2±0.9	87.1±0.7	87.1±1.0	87.2±0.9	86.5±1.0	86.9±1.0	87.4±0.8	86.5±1.2	86.8±0.9	86.5±0.7
flare	81.5±0.6	81.6±0.7	81.4±0.8	81.5±0.7	81.1±1.0	81.1±1.0	81.2±1.0	81.3±0.9	81.1±0.9	81.2±0.7	81.2±0.9	81.0±1.0
glas	96.0±1.0	96.0±1.0	96.0±1.0	95.2±0.7	96.0±0.7	96.0±0.7	96.2±0.6	96.2±0.6	96.2±0.6	96.4±0.8	96.6±0.8	96.2±0.6
hear	82.9±1.3	82.9±1.2	82.5±1.4	83.2±1.3	83.2±1.3	83.4±1.4	82.9±1.4	82.9±1.4	82.9±1.5	82.7±1.5	82.4±1.6	83.1±1.7
img	96.7±0.3	96.8±0.3	96.6±0.2	96.9±0.3	96.9±0.3	96.9±0.2	97.2±0.2	97.23±0.19	97.29±0.18	97.3±0.2	97.3±0.2	97.27±0.18
ionos	91.9±0.9	91.7±0.9	91.7±0.9	92.4±1.5	92.4±1.5	92.7±1.3	92.1±1.1	91.9±1.1	91.4±1.3	92.1±1.1	92.4±1.1	91.9±1.2
mok1	98.5±0.8	98.4±0.9	98.1±1.0	99.8±0.3	99.8±0.3	99.8±0.3	100±00	100±00	100±00	100±00	100±00	100±0
mok2	89± 2	88± 2	89± 2	91.9±1.2	92.6±0.9	92.5±0.8	92.0±1.0	92.5±0.6	93.0±0.5	93.4±1.1	93.6±1.0	93.6±1.0
pima	75.9±1.1	76.4±1.1	76.5±1.2	76.7±1.0	76.5±1.1	76.3±1.1	76.1±1.0	75.5±1.1	76.2±1.0	75.7±0.8	75.2±0.9	76.1±1.0
survi	74.3±1.5	73.8±1.6	74.3±1.3	74.8±1.5	74.6±1.5	74.6±1.4	74.3±1.3	74.3±1.4	73.9±1.3	74.1±1.3	74.8±1.3	74.9±1.4
vote	95.4±0.7	95.4±0.7	95.6±0.5	95.3±0.6	95.4±0.7	95.6±0.5	95.4±0.6	95.5±0.7	95.4±0.7	95.5±0.5	95.4±0.7	95.5±0.7
vowel	90.2±0.7	90.4±0.5	90.4±0.6	93.1±0.6	93.1±0.6	93.2±0.6	94.4±0.6	94.6±0.5	94.7±0.5	95.4±0.6	95.7±0.6	95.7±0.5
wdbc	97.3±0.4	97.1±0.4	97.4±0.4	96.9±0.5	97.1±0.5	96.9±0.5	97.2±0.4	96.9±0.5	97.0±0.5	97.0±0.5	97.3±0.4	97.4±0.4

Table C.148: Performance of Stacked Combiners - *STC-MFSE-CVCv3ACB* and *Boosting Combiner*

	3-expert			9-expert			20-expert			40-expert		
Database	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**
aritm	75.2±1.6	75.4±1.4	75.1±1.7	74.9±1.5	74.7±1.1	74.6±1.3	74.1±1.4	73.9±1.5	74.7±1.3	74.8±1.5	75.1±1.4	74.9±1.3
bala	96.9±0.3	96.7±0.4	96.4±0.5	96.6±0.4	96.6±0.4	96.6±0.4	96.6±0.6	96.5±0.5	96.7±0.5	96.5±0.5	96.5±0.5	96.4±0.5
band	72.4±1.6	72.4±1.6	72.4±1.6	73.6±1.8	73.8±1.8	73.3±1.8	72.5±1.6	72.4±1.5	72.9±1.4	71.6±1.7	71.8±1.6	71.5±1.5
bupa	72.0±1.7	71.6±1.8	71.9±1.6	71.6±1.4	71.4±1.3	71.4±1.4	70.7±1.8	70.4±1.7	70.0±1.6	71.4±1.5	71.3±1.4	70.7±1.8
cred	86.6±0.8	86.6±0.8	86.2±0.9	86.5±0.8	86.7±0.8	86.4±0.8	85.8±1.0	85.7±1.0	85.5±1.0	85.8±0.9	85.7±0.9	85.8±0.9
derma	97.5±0.7	97.9±0.6	97.7±0.6	97.2±0.7	97.5±0.6	97.3±0.6	97.0±0.7	97.6±0.5	97.7±0.5	97.5±0.6	97.9±0.5	97.6±0.5
ecoli	86.3±1.0	86.8±1.0	86.8±1.0	87.1±0.8	86.8±1.1	86.9±1.0	86.3±1.1	87.4±1.0	86.9±0.7	85.9±1.0	86.3±0.9	86.5±0.8
flare	81.4±0.9	81.3±0.8	81.3±0.9	81.1±0.9	80.8±1.0	80.7±0.9	81.2±0.8	81.6±1.0	81.5±0.9	80.9±0.9	80.6±0.9	80.9±1.0
glas	95.8±1.0	96.0±0.9	95.8±0.9	96.0±0.7	96.0±0.5	96.4±0.7	96.4±0.6	96.4±0.6	96.2±0.6	96.8±0.7	96.8±0.7	96.4±0.7
hear	83.4±1.2	83.1±1.2	82.9±1.1	83.2±1.3	83.7±1.3	83.7±1.4	83.1±1.5	83.2±1.4	82.9±1.4	82.9±1.6	82.4±1.7	82.7±1.7
img	96.7±0.3	96.8±0.3	96.69±0.19	96.8±0.3	96.9±0.3	97.0±0.2	97.2±0.2	97.25±0.19	97.33±0.18	97.2±0.2	97.3±0.2	97.35±0.19
ionos	91.7±1.0	91.9±0.9	92.0±0.8	92.4±1.5	92.7±1.4	92.4±1.5	92.3±1.0	91.9±1.1	91.4±1.0	92.3±1.1	92.3±1.3	91.7±1.3
mok1	98.5±0.8	98.5±0.8	98.4±0.9	99.8±0.3	99.8±0.3	99.8±0.3	100±0	100±0	100±0	100±0	100±0	100±0
mok2	88± 2	89± 2	89± 2	91.9±1.2	92.4±1.1	92.6±0.9	92.0±1.1	92.1±0.6	92.8±0.7	93.1±1.1	93.6±0.8	93.6±0.9
pima	76.6±1.2	76.5±1.1	76.6±1.1	76.8±1.0	76.9±1.0	76.9±1.0	76.0±1.1	75.7±1.1	75.8±1.1	76.3±1.0	75.9±1.1	76.2±1.1
survi	74.1±1.4	73.3±1.5	74.3±1.4	74.4±1.0	73.8±1.1	74.6±1.2	73.6±1.3	73.4±1.4	73.1±1.3	73.9±1.3	74.8±1.3	75.2±1.2
vote	95.4±0.7	95.4±0.7	95.4±0.7	95.3±0.6	95.4±0.7	95.4±0.7	95.5±0.6	95.4±0.7	95.4±0.7	95.5±0.5	95.4±0.5	95.1±0.6
vowel	89.3±0.8	90.1±0.5	90.4±0.6	93.0±0.7	93.1±0.6	93.6±0.6	94.3±0.6	94.3±0.5	94.8±0.5	95.4±0.6	95.5±0.6	95.8±0.5
wdbc	97.2±0.4	97.1±0.4	97.1±0.4	96.9±0.5	96.9±0.5	97.1±0.5	97.3±0.4	97.1±0.4	97.1±0.4	96.9±0.5	97.3±0.4	97.4±0.3

Table C.149: Performance of Stacked Combiners - *STCP-MFSE-CVCv3ACB* and *Output Average*

	3-expert			9-expert			20-expert			40-expert		
Database	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**
aritm	74.8±1.3	75.3±1.3	75.2±1.2	74.9±1.1	75.4±1.0	74.8±0.9	74.6±1.2	73.9±1.3	74.3±1.2	75.3±1.3	75.1±1.4	74.7±1.2
bala	96.9±0.3	93.9±0.6	94.4±0.5	96.5±0.4	96.5±0.4	96.5±0.5	96.4±0.5	96.6±0.5	96.6±0.6	96.7±0.5	96.7±0.5	96.7±0.5
band	73.5±2.0	72.5±1.8	73.8±1.8	73.1±1.9	72.9±1.7	72.7±1.8	72.0±1.4	72.7±1.4	71.8±1.5	72.0±1.5	72.9±1.7	72.5±1.8
bupa	72.1±1.3	71.6±1.3	70.9±1.3	72.1±1.2	72.1±1.3	71.3±1.3	70.0±1.6	70.1±1.6	70.7±1.5	71.0±1.4	70.6±1.1	71.7±1.3
cred	86.6±0.8	86.9±0.8	86.8±0.8	86.8±0.8	86.6±0.8	86.6±0.7	86.1±1.1	85.7±1.0	86.0±1.0	86.1±1.0	85.9±1.0	85.8±1.0
derma	97.6±0.6	97.7±0.6	97.7±0.7	96.9±0.7	97.5±0.6	97.5±0.6	97.5±0.6	97.2±0.7	97.3±0.6	97.5±0.7	97.5±0.7	97.5±0.6
ecoli	86.8±1.1	86.9±1.0	86.9±1.0	86.0±1.0	86.5±0.8	86.9±0.8	87.2±1.0	86.9±1.1	87.8±0.7	86.5±0.9	86.9±0.8	87.5±0.8
flare	81.7±0.7	81.8±0.6	81.8±0.7	81.7±0.6	81.6±0.8	82.2±0.8	81.9±0.9	81.5±0.8	81.8±1.1	81.3±1.0	81.5±1.0	81.4±1.1
glas	95.6±0.9	96.0±0.9	95.8±0.9	96.0±0.7	96.4±0.7	96.2±0.7	96.4±0.7	96.4±0.7	96.4±0.7	95.6±0.8	95.6±0.8	95.8±0.7
hear	84.6±0.9	84.9±1.2	84.2±1.3	83.9±1.3	84.1±1.3	83.7±1.5	82.9±1.1	83.2±1.2	83.4±1.3	82.4±1.4	82.7±1.3	82.2±1.0
img	96.9±0.2	97.08±0.17	97.27±0.17	97.0±0.3	97.1±0.2	97.4±0.3	97.2±0.2	97.12±0.16	97.33±0.15	97.1±0.2	97.38±0.14	97.38±0.15
ionos	91.9±0.7	92.1±0.7	92.7±1.0	92.4±1.2	92.4±1.1	93.0±1.0	92.1±1.0	92.3±1.1	92.4±1.0	92.7±1.0	92.4±1.0	92.1±1.0
mok1	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0
mok2	89± 2	89± 2	88± 2	91.8±1.5	92.4±1.2	91.8±1.1	92.8±1.3	93.3±0.9	93.3±1.1	94.0±1.0	93.3±1.1	93.3±1.1
pima	76.7±0.8	76.4±0.7	76.6±0.9	76.5±1.1	76.1±0.8	76.6±0.9	76.8±1.0	76.6±1.0	76.5±1.0	76.8±0.8	76.6±1.1	75.7±1.1
survi	74.4±1.7	74.3±1.7	74.1±1.2	73.9±1.3	74.3±1.3	73.8±1.4	73.6±1.3	74.1±1.0	73.8±1.2	74.3±1.3	74.9±1.6	73.9±1.7
vote	95.8±0.5	95.9±0.6	96.0±0.6	95.6±0.5	95.6±0.6	95.5±0.6	95.9±0.6	95.8±0.6	95.3±0.7	95.4±0.6	95.4±0.6	95.5±0.5
vowel	90.6±0.5	91.1±0.6	91.8±0.6	93.2±0.7	93.4±0.7	93.7±0.6	94.6±0.6	94.9±0.7	95.1±0.6	95.3±0.5	95.6±0.5	95.9±0.5
wdbc	97.1±0.4	97.1±0.4	97.0±0.4	97.0±0.4	96.9±0.4	96.6±0.4	96.8±0.6	97.1±0.5	96.6±0.5	96.9±0.5	96.9±0.5	96.8±0.4

Table C.150: Performance of Stacked Combiners - *STCP-MFSE-CVCv3ACB* and *Boosting Combiner*

	3-expert			9-expert			20-expert			40-expert		
Database	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**
aritm	75.5±1.3	75.5±1.3	74.8±1.1	74.7±1.0	75.1±1.0	75.1±1.0	74.5±1.3	74.1±1.2	74.6±1.2	74.9±1.2	74.9±1.3	74.9±1.2
bala	96.9±0.3	95.2±0.5	95.4±0.5	96.6±0.4	96.5±0.4	96.6±0.4	96.5±0.5	96.6±0.5	96.6±0.5	96.6±0.4	96.6±0.5	96.9±0.5
band	72.5±1.7	72.4±1.5	72.7±1.9	73.1±1.7	73.8±1.8	72.7±2.0	72.0±1.4	72.2±1.5	72.0±1.6	72.9±1.6	73.3±1.6	73.6±1.8
bupa	71.9±1.3	72.0±1.3	71.6±1.4	72.0±1.3	71.6±1.2	71.6±1.3	70.7±1.6	70.7±1.8	71.0±1.7	71.0±1.3	70.9±1.6	72.1±1.7
cred	86.4±0.8	86.6±0.8	86.5±0.7	86.3±0.6	86.2±0.5	86.5±0.8	86.0±1.0	85.6±1.0	85.9±0.9	86.2±0.9	86.2±1.0	85.9±0.9
derma	97.7±0.6	97.7±0.6	97.3±0.7	97.2±0.6	97.3±0.5	97.2±0.6	97.0±0.5	97.2±0.6	97.2±0.6	97.6±0.6	97.6±0.6	97.3±0.7
ecoli	86.0±1.2	86.8±1.0	86.9±0.9	86.3±1.1	86.5±0.9	86.9±1.0	86.5±1.0	86.8±1.0	86.9±0.9	86.2±0.7	87.1±0.8	86.8±0.8
flare	81.4±0.7	81.2±0.8	81.3±0.7	81.5±1.0	81.9±0.7	82.0±0.8	81.8±1.0	81.0±0.8	80.9±0.8	80.6±0.8	81.0±0.9	81.6±1.0
glas	96.0±0.7	96.8±0.7	96.2±1.1	95.6±0.8	96.4±0.7	96.8±0.7	96.8±0.7	97.2±0.7	97.2±0.7	96.2±0.7	96.0±0.7	96.4±0.7
hear	83.9±0.9	84.2±1.3	84.2±1.2	84.1±1.3	83.9±1.3	83.6±1.3	83.2±1.2	83.1±1.2	83.4±1.1	82.5±1.5	83.1±1.2	82.7±1.2
img	96.8±0.3	97.01±0.20	97.25±0.20	96.9±0.3	97.0±0.2	97.2±0.3	97.08±0.19	97.14±0.18	97.38±0.16	97.1±0.2	97.40±0.12	97.51±0.14
ionos	91.9±0.7	91.7±0.9	92.3±0.9	92.9±1.2	92.4±1.1	92.6±1.2	92.4±1.0	92.6±1.1	92.7±1.0	92.6±1.0	92.6±1.0	92.3±1.0
mok1	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0
mok2	88± 2	89± 2	88± 2	92.3±1.3	92.5±1.3	91.8±1.2	92.6±1.2	93.3±0.8	93.0±1.1	94.0±1.0	93.5±0.8	93.6±0.8
pima	76.8±0.9	76.1±0.9	76.6±0.8	76.4±1.2	76.6±1.0	76.4±1.0	76.1±1.1	76.1±1.1	76.4±1.1	77.0±0.9	77.2±0.8	76.3±1.1
survi	74.1±1.5	73.3±1.6	73.4±1.3	73.8±1.2	74.4±1.4	73.9±1.4	73.9±1.4	74.4±1.4	74.4±1.3	75.1±1.3	75.2±1.6	74.8±1.4
vote	95.9±0.5	95.8±0.5	95.4±0.6	95.6±0.5	95.8±0.6	95.6±0.6	95.8±0.5	95.8±0.5	95.4±0.6	95.5±0.5	95.5±0.5	95.4±0.6
vowel	90.5±0.7	91.4±0.8	91.7±0.6	93.0±0.8	93.5±0.7	93.7±0.6	94.4±0.6	94.6±0.7	94.8±0.7	95.0±0.5	95.3±0.5	95.9±0.5
wdbc	97.0±0.4	96.9±0.4	96.8±0.4	96.9±0.4	96.9±0.4	96.9±0.4	96.6±0.5	97.0±0.5	96.6±0.5	97.0±0.5	96.8±0.5	96.9±0.5

Table C.151: Performance of Stacked Combiners - *STC-MFSE-CVCv3Conserboost* and *Output Average*

	3-expert			9-expert			20-expert			40-expert		
Database	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**
aritm	75.3±1.5	74.9±1.8	74.9±1.4	74.5±1.5	74.3±1.4	74.6±1.4	74.8±1.2	74.5±1.4	74.3±1.3	73.9±1.0	74.4±1.2	74.7±1.5
bala	96.4±0.5	96.3±0.5	96.1±0.6	96.5±0.5	96.3±0.5	96.2±0.5	96.6±0.5	96.6±0.5	96.5±0.5	96.9±0.4	96.6±0.6	96.4±0.6
band	72.0±1.7	71.8±1.9	72.2±1.7	72.5±1.5	73.8±1.9	73.3±1.9	71.8±1.8	72.2±1.6	72.0±1.6	71.1±1.3	72.0±1.5	71.8±1.8
bupa	71.6±1.3	71.4±1.5	71.9±1.3	71.4±1.3	70.0±1.6	69.1±1.6	71.1±1.5	71.3±1.4	70.6±1.7	70.4±1.8	69.4±1.7	69.0±1.9
cred	86.5±0.8	86.0±0.8	85.4±0.5	86.8±0.8	86.1±0.8	85.5±0.8	85.4±0.7	85.5±1.0	85.5±0.9	85.6±1.0	85.5±1.0	84.8±0.9
derma	97.5±0.7	97.3±0.7	97.5±0.7	97.3±0.6	96.9±0.7	97.3±0.6	97.2±0.7	97.3±0.6	97.5±0.5	97.5±0.7	97.6±0.6	97.6±0.5
ecoli	86.2±1.0	85.9±1.0	85.7±1.1	85.6±1.0	86.5±0.9	86.5±0.8	86.8±1.1	87.4±0.9	86.9±0.8	86.8±0.8	86.0±0.8	86.9±0.4
flare	81.8±0.7	81.1±0.9	81.1±0.8	81.6±0.8	81.0±0.9	80.7±0.8	80.5±1.1	79.9±1.0	80.3±0.9	81.2±1.0	80.6±0.9	80.1±0.9
glas	95.6±1.0	96.6±0.7	96.6±0.8	96.2±0.8	95.6±0.7	96.2±0.6	96.6±0.7	96.4±0.7	96.6±0.7	96.4±0.8	96.4±0.8	96.4±0.8
hear	82.2±1.2	82.4±1.2	82.2±1.2	83.1±1.2	82.7±1.2	82.9±1.1	82.7±1.2	82.5±1.3	83.1±1.5	83.2±1.4	83.4±1.5	83.1±1.6
img	96.5±0.3	96.2±0.2	96.6±0.2	97.1±0.2	97.1±0.2	96.9±0.2	97.1±0.2	97.16±0.19	97.25±0.18	97.0±0.2	97.27±0.20	97.29±0.17
ionos	91.7±1.1	91.7±1.1	92.1±1.0	92.1±1.4	92.3±1.3	92.4±1.4	92.4±1.2	91.6±1.2	91.4±1.3	91.3±1.2	91.0±1.3	91.3±1.2
mok1	98.3±1.0	97.5±1.3	97.5±1.3	99.8±0.3	99.8±0.3	99.8±0.3	100±0	100±0	100±0	100±0	100±0	100±0
mok2	89± 2	89± 2	89± 2	91.4±1.1	92.0±1.0	91.5±1.0	92.0±1.1	93.3±0.8	92.1±0.7	92.9±1.0	93.3±0.8	92.4±0.8
pima	76.5±1.2	76.5±1.2	76.5±1.2	76.3±1.2	76.5±1.1	76.2±0.9	76.1±1.1	76.5±1.1	76.1±0.9	75.5±1.0	75.7±1.2	75.5±1.0
survi	74.9±1.5	75.2±1.5	75.4±1.5	74.4±1.2	75.7±1.2	74.8±1.5	74.3±1.4	73.9±1.3	74.3±1.4	75.4±1.2	75.1±1.3	75.1±1.5
vote	95.5±0.7	95.9±0.6	96.0±0.6	95.4±0.7	95.8±0.6	95.6±0.6	95.5±0.7	95.5±0.6	95.6±0.5	95.5±0.5	95.6±0.5	95.5±0.5
vowel	90.5±0.7	90.4±0.6	90.3±0.5	93.3±0.5	93.2±0.7	94.0±0.7	94.6±0.6	94.7±0.5	94.9±0.6	95.4±0.5	95.5±0.5	95.7±0.5
wdbc	96.9±0.5	96.8±0.5	96.6±0.5	96.7±0.5	96.9±0.5	96.8±0.5	97.2±0.5	97.1±0.5	96.9±0.5	97.0±0.5	96.4±0.3	96.4±0.4

Table C.152: Performance of Stacked Combiners - *STC-MFSE-CVCv3Conserboost* and *Boosting Combiner*

	3-expert			9-expert			20-expert			40-expert		
Database	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**
aritm	75.2±1.7	75.4±1.6	75.4±1.6	73.7±1.1	73.3±1.2	74.0±1.2	73.8±1.3	74.4±1.1	74.1±1.3	73.6±0.8	73.7±1.0	73.9±0.9
bala	96.9±0.3	96.5±0.5	96.5±0.5	96.6±0.4	96.5±0.4	96.7±0.4	96.8±0.4	96.7±0.5	96.6±0.4	96.8±0.3	96.4±0.5	96.3±0.5
band	72± 2	72± 2	72.2±2.0	73.3±2.0	73.5±1.8	73.3±1.8	71.8±1.8	72.0±1.7	71.8±1.7	71.5±1.3	71.6±1.4	71.8±1.1
bupa	71.7±1.5	71.9±1.5	71.6±1.5	72.6±1.2	71.9±1.2	70.7±1.3	70.3±1.8	70.9±1.5	70.3±1.8	70.6±1.8	70.3±1.9	69.9±1.9
cred	86.2±0.9	86.5±0.8	86.2±0.7	86.5±0.5	86.9±0.7	87.0±0.7	85.8±0.9	85.8±1.0	85.9±1.0	85.8±1.0	85.5±0.9	85.5±1.0
derma	97.3±0.9	97.6±0.7	97.6±0.6	97.0±0.7	97.2±0.6	97.2±0.7	97.5±0.5	97.5±0.5	97.6±0.5	97.0±0.7	97.6±0.5	97.6±0.5
ecoli	86.2±1.0	86.6±0.9	86.6±1.0	86.0±1.1	86.3±1.1	87.1±0.9	86.8±0.9	86.5±0.8	87.1±1.0	85.3±0.8	86.6±0.7	86.6±0.6
flare	81.2±0.7	81.2±0.7	81.0±0.6	80.5±1.0	80.1±0.8	80.0±0.8	81.3±0.8	81.2±0.8	81.3±0.8	81.3±0.9	81.3±0.8	81.1±0.9
glas	96.0±0.8	96.4±0.7	96.6±0.7	95.6±0.8	95.4±0.8	96.6±0.7	96.4±0.7	96.6±0.7	96.4±0.7	96.2±0.6	96.8±0.6	96.4±0.7
hear	82.5±1.2	82.0±1.2	82.7±1.4	83.7±1.0	83.6±1.2	83.6±1.2	82.4±1.3	82.5±1.2	83.1±1.2	82.5±1.3	82.9±1.3	82.9±1.4
img	96.5±0.2	96.7±0.2	96.7±0.2	97.10±0.19	97.1±0.2	97.03±0.18	97.2±0.2	97.20±0.19	97.29±0.19	97.1±0.2	97.27±0.16	97.29±0.19
ionos	92.3±0.9	92.4±0.9	92.4±0.9	92.1±1.3	92.0±1.2	92.3±1.3	92.7±1.3	92.1±1.2	92.1±1.1	91.4±1.1	91.3±1.1	90.9±1.1
mok1	98.5±0.8	98.5±0.8	98.1±1.1	99.6±0.4	99.6±0.4	99.6±0.4	100±0	100±0	100±0	100±0	100±0	100±0
mok2	88± 2	89± 2	89± 2	90.6±1.2	91.3±0.9	91.0±1.1	92.0±1.1	92.5±0.8	92.4±1.0	93.0±1.0	93.3±0.8	92.9±0.8
pima	76.6±1.1	76.6±1.1	76.2±1.2	76.4±1.2	76.8±1.2	76.8±1.2	76.5±1.0	76.5±1.0	76.5±1.0	75.4±1.0	75.5±1.0	75.5±1.0
survi	75.1±1.2	74.9±1.3	74.8±1.3	73.9±1.4	73.9±1.4	74.4±1.4	75.2±1.4	74.4±1.3	74.4±1.5	75.2±1.5	74.6±1.5	74.3±1.4
vote	96.0±0.6	96.3±0.6	96.3±0.6	95.4±0.7	95.4±0.7	95.4±0.7	95.4±0.5	95.4±0.5	95.5±0.5	95.6±0.5	95.4±0.7	95.4±0.7
vowel	90.1±0.7	90.2±0.7	90.2±0.5	92.8±0.6	92.9±0.6	93.8±0.7	94.8±0.7	94.8±0.6	95.0±0.6	95.3±0.5	95.2±0.5	95.5±0.6
wdbc	97.0±0.6	97.1±0.5	97.1±0.5	96.8±0.5	96.9±0.5	96.8±0.5	97.1±0.4	97.2±0.4	97.3±0.4	97.1±0.5	97.1±0.5	97.1±0.5

Table C.153: Performance of Stacked Combiners - *STCP-MFSE-CVCv3Conserboost* and *Output Average*

	3-expert			9-expert			20-expert			40-expert		
Database	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**
aritm	75.6±1.4	75.3±1.3	75.4±1.3	74.5±1.2	74.5±1.5	75.4±1.3	74.7±1.2	75.3±1.5	74.3±1.4	74.3±1.1	74.7±1.2	74.1±1.2
bala	96.9±0.3	95.4±0.4	96.4±0.5	96.5±0.4	96.6±0.5	96.5±0.5	96.7±0.6	96.8±0.5	96.3±0.6	96.6±0.5	96.6±0.6	96.6±0.6
band	71.8±1.9	72.7±1.4	72.9±1.6	73.1±1.8	73.8±1.9	74.5±1.8	72.4±1.6	72.0±1.5	72.4±1.7	72.5±1.4	72.7±1.7	73.1±1.6
bupa	71.9±1.7	70.9±1.4	71.0±1.4	71.9±1.3	72.1±1.2	72.1±1.2	71.0±1.8	71.4±1.7	70.1±1.7	71.6±1.6	71.3±1.5	71.0±1.8
cred	86.3±0.7	86.2±0.7	86.4±0.7	86.5±0.8	86.6±0.7	85.8±0.6	85.2±0.9	85.1±1.0	84.8±1.2	86.2±0.9	85.4±1.0	85.3±1.0
derma	97.5±0.7	97.3±0.7	97.5±0.6	97.3±0.8	97.3±0.6	97.3±0.7	97.2±0.7	97.2±0.6	97.0±0.6	97.0±0.6	96.9±0.7	97.2±0.6
ecoli	86.8±0.9	87.4±0.8	86.5±0.8	86.5±1.0	87.1±0.8	87.4±0.7	86.2±0.7	86.9±0.7	85.9±1.0	87.5±0.7	86.3±0.9	86.0±0.9
flare	81.4±1.0	81.8±0.8	82.0±0.8	81.2±0.6	80.7±0.9	81.1±0.8	81.3±0.9	81.3±0.9	80.7±1.0	80.5±1.1	81.0±0.9	80.2±0.9
glas	96.6±0.7	96.2±0.8	96.6±0.8	96.4±0.5	96.4±0.6	96.4±0.7	96.2±0.6	96.6±0.7	96.6±0.7	95.6±0.7	95.6±0.7	96.2±0.6
hear	83.4±1.3	83.9±1.2	83.1±1.1	84.1±1.3	82.5±1.1	82.5±1.5	82.5±1.4	82.2±1.0	82.9±1.3	83.1±1.7	83.4±1.9	83.6±1.8
img	97.01±0.18	97.10±0.18	97.25±0.16	97.1±0.2	97.16±0.20	97.3±0.2	97.25±0.19	97.3±0.2	97.48±0.19	97.29±0.19	97.25±0.18	97.18±0.19
ionos	92.0±0.9	92.4±0.9	92.7±1.0	93.0±0.9	92.9±1.0	93.6±0.9	92.4±0.9	92.9±0.9	92.6±0.9	92.0±1.2	92.4±1.0	92.7±0.9
mok1	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0
mok2	89± 3	88± 2	88± 2	92.3±1.5	91.4±1.1	90.8±1.0	93.1±1.1	93.5±1.2	93.1±1.1	93.4±0.8	93.1±0.8	92.6±0.9
pima	76.5±1.0	76.6±0.9	76.7±1.0	75.8±1.1	75.9±1.0	76.5±0.9	76.3±1.1	76.3±1.1	76.3±1.0	75.9±1.0	76.1±1.0	75.5±0.9
survi	74.6±1.6	74.9±1.4	74.6±1.6	73.9±1.6	73.4±1.4	74.6±1.1	74.8±1.5	74.3±1.5	75.2±1.5	74.9±1.3	74.6±1.5	74.6±1.3
vote	96.0±0.6	96.1±0.5	96.0±0.5	95.6±0.5	95.8±0.5	95.3±0.7	95.5±0.7	95.8±0.7	95.6±0.6	95.0±0.6	95.6±0.6	95.6±0.7
vowel	90.3±0.7	91.4±0.7	91.6±0.7	93.4±0.7	93.7±0.7	93.8±0.7	94.7±0.5	95.1±0.5	95.1±0.5	95.2±0.5	95.7±0.6	95.9±0.5
wdbc	96.2±0.5	96.4±0.4	96.8±0.4	96.9±0.5	96.6±0.4	96.6±0.5	97.0±0.6	96.8±0.5	96.5±0.4	96.9±0.5	97.1±0.4	96.6±0.4

Table C.154: Performance of Stacked Combiners - *STCP-MFSE-CVCv3Conserboost* and *Boosting Combiner*

	3-expert			9-expert			20-expert			40-expert		
Database	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**	**3-comb**	**5-comb**	**9-comb**
aritm	76.1±1.3	75.1±1.4	76.0±1.5	74.9±1.0	74.6±1.0	74.6±1.1	74.5±1.3	75.1±1.2	73.6±1.0	74.4±1.0	74.4±0.8	75.6±1.0
bala	96.9±0.3	95.5±0.4	95.9±0.5	96.3±0.5	96.6±0.4	96.6±0.4	96.6±0.5	96.8±0.5	96.6±0.5	96.6±0.5	96.4±0.7	96.6±0.6
band	71.1±1.7	72.2±1.7	72.4±1.7	73.5±1.7	73.6±1.8	73.3±1.5	71.8±1.7	70.9±1.8	71.8±1.5	71.5±1.2	72.5±1.7	72.9±1.4
bupa	71.3±1.4	71.4±1.3	71.1±1.3	71.0±1.5	71.4±1.3	71.1±1.3	70.3±1.5	71.4±1.5	70.3±1.6	71.0±1.8	70.9±1.8	70.4±1.8
cred	87.2±0.8	87.0±0.8	86.8±0.7	86.0±0.6	86.4±0.6	86.2±0.4	84.8±0.7	86.2±0.9	85.7±0.9	85.8±0.8	85.4±1.0	85.2±1.0
derma	97.7±0.7	97.6±0.6	97.7±0.6	97.7±0.6	97.5±0.5	97.3±0.6	97.5±0.6	97.5±0.6	97.5±0.6	97.5±0.5	97.2±0.6	97.5±0.5
ecoli	86.8±1.0	86.9±1.0	86.9±0.9	85.6±1.3	86.5±1.4	87.4±0.8	85.3±0.8	86.2±0.9	86.0±0.8	85.9±0.8	87.1±0.9	86.9±0.8
flare	81.2±0.9	81.9±0.7	82.2±0.6	81.3±0.6	81.8±0.7	81.7±0.7	81.3±0.9	81.4±0.9	81.3±0.8	81.0±1.1	81.3±1.0	81.2±0.8
glas	97.0±0.7	96.8±0.8	96.8±0.9	96.8±0.6	96.6±0.5	97.0±0.6	96.4±0.8	97.4±0.7	97.2±0.7	96.0±0.7	95.8±0.7	96.0±0.7
hear	81.9±1.3	83.1±1.2	83.9±1.2	84.1±1.3	83.1±1.5	81.7±1.5	83.2±1.1	83.2±1.0	83.6±1.2	81.7±1.7	81.5±1.9	83.2±1.6
img	96.9±0.2	97.0±0.2	97.20±0.18	96.9±0.2	97.14±0.19	97.20±0.19	97.16±0.19	97.33±0.19	97.27±0.15	97.29±0.16	97.20±0.17	97.1±0.2
ionos	91.9±0.7	92.1±0.6	92.7±0.8	93.1±1.0	93.1±1.0	93.4±1.0	92.3±0.9	92.3±0.8	92.3±0.8	92.1±1.2	92.3±1.3	92.3±1.1
mok1	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0
mok2	88± 3	88± 3	88± 2	91.9±1.4	91.5±1.0	91.0±1.0	92.9±1.1	93.5±0.8	93.5±0.7	93.3±0.8	93.4±1.1	92.0±0.9
pima	76.3±1.0	76.6±1.0	76.8±1.0	75.9±0.9	76.3±0.9	76.5±0.9	76.4±1.1	76.1±1.3	76.3±1.2	75.8±1.1	76.1±1.0	75.8±1.1
survi	74.1±1.5	74.9±1.6	74.4±1.5	73.9±1.4	74.3±1.4	75.1±1.5	74.9±1.3	74.4±1.4	73.8±1.4	74.9±1.5	74.3±1.2	75.1±1.5
vote	96.0±0.6	96.0±0.6	96.1±0.5	95.6±0.5	95.4±0.6	95.3±0.7	95.3±0.6	95.3±0.5	95.1±0.6	95.0±0.6	95.1±0.5	95.5±0.5
vowel	89.8±0.9	91.1±0.7	91.3±0.7	93.6±0.6	93.9±0.7	93.9±0.6	94.9±0.6	95.4±0.5	95.3±0.5	95.3±0.6	95.7±0.6	96.0±0.5
wdbc	96.4±0.6	96.5±0.5	97.1±0.5	97.2±0.5	96.6±0.5	96.9±0.4	96.9±0.5	97.0±0.5	96.9±0.5	96.7±0.5	96.7±0.6	96.5±0.5

Table C.155: Performance - *MIX-BN-BN*

Database	3-net	9-net	20-net	40-net
aritm	50 ± 2	70.2 ± 1.5	72.5 ± 0.7	73.2 ± 1
bala	90.5 ± 0.9	90.2 ± 1	91 ± 0.7	89.8 ± 0.8
band	73 ± 2	74.4 ± 1.3	74 ± 1.9	75.5 ± 1.3
bupa	70.6 ± 1.9	73.4 ± 1.5	72.4 ± 1.1	72.1 ± 1.7
cred	86.8 ± 0.5	86.9 ± 0.5	86.5 ± 0.6	86 ± 0.5
derma	96.8 ± 0.7	97 ± 0.6	97.3 ± 0.6	97.2 ± 0.8
ecoli	52.5 ± 1.1	48 ± 2	69.9 ± 1.5	74.7 ± 1.4
flare	81.5 ± 0.5	81.7 ± 0.5	81.7 ± 0.6	81.8 ± 0.6
glas	89.4 ± 1	91.2 ± 1.1	90.2 ± 1.3	91 ± 1.1
hear	81.4 ± 1.3	83.1 ± 1.5	82.4 ± 1.6	83.1 ± 1.7
img	71.6 ± 1.6	89.6 ± 1.1	89.6 ± 1.7	88.3 ± 1.3
ionos	84.3 ± 1.8	84.7 ± 1.6	84 ± 1.8	82.3 ± 1.5
mok1	88 ± 2	94 ± 3	94 ± 2	94 ± 3
mok2	62.1 ± 1.7	67.5 ± 2	66.8 ± 1.6	68 ± 2
pima	66.8 ± 1.2	67.9 ± 1.1	69.1 ± 1	69.2 ± 1.2
survi	72.3 ± 1.2	72.6 ± 0.9	73.8 ± 0.9	73.6 ± 1.2
vote	95 ± 1.2	96.1 ± 0.6	96.1 ± 0.6	96.5 ± 0.7
vowel	47.7 ± 1.6	73.8 ± 2	78.3 ± 1	77.6 ± 0.7
wdbc	94.7 ± 0.5	94.9 ± 0.4	95.1 ± 0.6	94.6 ± 0.5

Table C.156: Performance - *MIX-MF-BN*

Database	3-net	9-net	20-net	40-net
aritm	73.8 ± 1.3	73.6 ± 1.3	72.6 ± 1.5	73.7 ± 1.2
bala	94.1 ± 1	93.9 ± 1.2	94.6 ± 1.1	95.2 ± 0.7
band	75.5 ± 1.9	74 ± 2	74.7 ± 1.7	73.8 ± 1.6
bupa	71.3 ± 1.6	72.6 ± 1.3	71.1 ± 1.8	72.1 ± 2
cred	85.8 ± 0.5	86.8 ± 0.7	86.3 ± 0.7	86.6 ± 0.6
derma	97 ± 0.8	97.2 ± 0.8	97 ± 0.7	96.9 ± 0.8
ecoli	85 ± 0.8	86.5 ± 1	85.9 ± 0.7	84.6 ± 1.3
flare	82.1 ± 0.6	81.9 ± 0.5	81.4 ± 0.5	81.5 ± 0.5
glas	94.6 ± 1	94.6 ± 1.2	94.2 ± 1.3	95 ± 1.2
hear	82.2 ± 1.3	82.9 ± 0.9	82 ± 1	82.4 ± 1
img	95.9 ± 0.4	96.2 ± 0.2	96.2 ± 0.2	96.3 ± 0.2
ionos	89.4 ± 1	89.9 ± 0.9	89.7 ± 0.9	90.1 ± 0.8
mok1	99.3 ± 0.8	99.3 ± 0.8	98.8 ± 0.9	100 ± 0
mok2	77 ± 3	77 ± 2	84 ± 1.8	80.3 ± 1.8
pima	75.3 ± 1.2	75.6 ± 1.1	75.6 ± 1.1	75.6 ± 1.1
survi	74.6 ± 1.3	74.9 ± 1.2	74.6 ± 1.1	75.1 ± 1.2
vote	96.1 ± 0.6	96.1 ± 0.6	96.1 ± 0.6	95.8 ± 0.6
vowel	85.1 ± 1.1	86.7 ± 1.3	86.2 ± 0.9	85.4 ± 0.9
wdbc	96.9 ± 0.5	96.9 ± 0.5	96.9 ± 0.5	96.9 ± 0.5

Table C.157: Performance - *MIX-MF-MF*

Database	3-net	9-net	20-net	40-net
aritm	75.4 ± 1.2	74.1 ± 0.7	74.6 ± 1.1	74.4 ± 0.9
bala	95.1 ± 0.6	95 ± 1	94.2 ± 0.9	94.9 ± 0.6
band	74.2 ± 1.4	73.5 ± 1.2	74.6 ± 1.6	74.9 ± 1.5
bupa	72.3 ± 1.4	72 ± 1.2	72.4 ± 1.3	71.6 ± 1.6
cred	86.5 ± 0.8	86.9 ± 0.8	87.1 ± 0.7	86.5 ± 0.8
derma	97.2 ± 0.7	96.9 ± 0.9	97 ± 0.8	96.3 ± 1.1
ecoli	85.4 ± 0.6	84.3 ± 0.8	86.3 ± 1	86.5 ± 0.8
flare	81.5 ± 0.5	81.6 ± 0.5	81.5 ± 0.4	81.6 ± 0.5
glas	95.2 ± 0.7	94.6 ± 1	95.2 ± 1	93.8 ± 1
hear	82.9 ± 1.5	82.9 ± 1.3	82.7 ± 1.5	83.1 ± 1.3
img	96.2 ± 0.2	96 ± 0.3	96 ± 0.4	95.7 ± 0.2
ionos	89.6 ± 1.4	90.1 ± 1.1	91 ± 0.6	91.1 ± 1
mok1	98.6 ± 0.9	98.3 ± 0.9	99.5 ± 0.5	98.4 ± 0.8
mok2	90.3 ± 1.2	87.3 ± 1.5	88.5 ± 1.5	90.8 ± 1.6
pima	75.9 ± 1.1	75.6 ± 1.2	75.4 ± 1	75.9 ± 1.3
survi	74.6 ± 1.2	74.8 ± 1.4	74.4 ± 1.2	74.3 ± 1.3
vote	96 ± 0.6	96 ± 0.6	96 ± 0.6	96.1 ± 0.6
vowel	86.3 ± 0.9	85.5 ± 0.9	86.5 ± 0.8	84.3 ± 1
wdbc	96.9 ± 0.5	96.8 ± 0.5	97 ± 0.4	97 ± 0.4

Table C.158: Performance - *MIX-SE-BN*

Database	3-net	9-net	20-net	40-net
aritm	73.3 ± 1	73.2 ± 0.8	73.1 ± 0.9	73.3 ± 1
bala	96.2 ± 0.6	96.1 ± 0.7	96.5 ± 0.5	96.6 ± 0.5
band	71.6 ± 1.9	72.9 ± 1.8	72.4 ± 1.4	72 ± 1.2
bupa	72.1 ± 1.3	71.6 ± 1.4	71.6 ± 1.5	71.3 ± 1.5
cred	86.9 ± 0.7	86.6 ± 0.8	86.2 ± 0.7	86.2 ± 0.7
derma	97 ± 0.7	97.3 ± 0.7	97.3 ± 0.7	97.3 ± 0.7
ecoli	86.2 ± 0.9	87.4 ± 0.8	87.2 ± 0.6	86.9 ± 0.6
flare	81.4 ± 0.5	81.4 ± 0.5	81.2 ± 0.6	81 ± 0.6
glas	93.8 ± 0.6	94.4 ± 0.6	94.4 ± 0.6	94.8 ± 0.7
hear	82.7 ± 1.3	82.4 ± 1.2	82.4 ± 1.2	82.4 ± 1.2
img	96.6 ± 0.2	96.6 ± 0.3	96.8 ± 0.3	96.7 ± 0.2
ionos	91 ± 1	91.4 ± 1.2	91.9 ± 1	91.9 ± 0.9
mok1	98.5 ± 0.8	99.8 ± 0.3	100 ± 0	100 ± 0
mok2	87 ± 2	91.1 ± 1.5	91.8 ± 1.4	91.1 ± 0.9
pima	75.9 ± 1.3	75.7 ± 1.2	75.6 ± 1.2	75.7 ± 1.1
survi	74.1 ± 1.6	74.8 ± 1.6	74.3 ± 1.2	74.3 ± 1.2
vote	95.6 ± 0.5	95.6 ± 0.5	95.6 ± 0.5	95.6 ± 0.5
vowel	87.5 ± 0.9	91.4 ± 0.7	92.3 ± 0.7	93.3 ± 0.6
wdbc	96.9 ± 0.5	96.9 ± 0.5	96.9 ± 0.5	97 ± 0.4

Table C.159: Performance - *MIX-SE-MF*

Database	3-net	9-net	20-net	40-net
aritm	73.3 ± 1	73.9 ± 1.1	73.2 ± 1.2	73.5 ± 1.2
bala	96.2 ± 0.5	96.2 ± 0.5	95.9 ± 0.8	96.2 ± 0.7
band	72.6 ± 1.6	72.9 ± 1.7	72.9 ± 1.4	72.4 ± 1.4
bupa	72 ± 1.5	72.3 ± 1.3	71.4 ± 1.4	71.6 ± 1.4
cred	86.8 ± 0.7	86.5 ± 0.7	85.9 ± 0.8	86 ± 0.9
derma	97.2 ± 0.7	97.2 ± 0.7	97.3 ± 0.7	97.3 ± 0.7
ecoli	86.5 ± 0.9	87.5 ± 0.7	86.8 ± 0.8	86.9 ± 0.6
flare	81.8 ± 0.5	81.5 ± 0.5	81 ± 0.8	81.3 ± 0.5
glas	94 ± 0.8	94 ± 0.7	94.4 ± 0.7	94.2 ± 0.6
hear	82.9 ± 1.5	82.7 ± 1.1	82.4 ± 1.3	82.5 ± 1.5
img	96.6 ± 0.2	96.6 ± 0.3	96.7 ± 0.2	96.8 ± 0.2
ionos	91.1 ± 0.9	91.3 ± 1.1	91.1 ± 1	91.9 ± 0.9
mok1	98.5 ± 0.8	99.8 ± 0.3	100 ± 0	100 ± 0
mok2	88 ± 2	91.5 ± 1.3	91.8 ± 0.9	91.6 ± 1.4
pima	75.9 ± 1.2	75.4 ± 1.3	75.7 ± 1.2	75.7 ± 1.1
survi	74.3 ± 1.5	74.4 ± 1.5	74.3 ± 1.3	74.3 ± 1.3
vote	95.6 ± 0.5	95.6 ± 0.5	95.6 ± 0.5	95.6 ± 0.5
vowel	87.6 ± 0.9	91.3 ± 0.5	91.8 ± 0.8	92.3 ± 0.6
wdbc	96.9 ± 0.5	96.9 ± 0.5	96.9 ± 0.5	97 ± 0.4

Bibliography

[1] Friedrich Leisch. *Ensemble Methods for Neural Clustering and Classification.* PhD thesis, Institut für Statistik, Wahrscheinlichkeitsttheorie und Versicherungsmathematik, Technische Universität Wien, 1998.

[2] Henrik Haraldsson. *Neural Network Ensembles and Combinational Optimization with Applications in Medicine.* PhD thesis, Faculty of Neural Sciences, Lund University, 2003.

[3] Nayer Wanas. *Feature Based Architecture for Decision Fusion.* PhD thesis, University of Waterloo, 2003.

[4] Gavin Brown. *Diversity in Neural Network Ensembles.* PhD thesis, University of Birmingham, 2004.

[5] Zengchang Qin. *Learning with Fuzzy Labels.* PhD thesis, Department of Mathematics, University of Bristol, 2005.

[6] Samuel Robert Reid. *Partial Ensemble Selection for Multiclass Classification.* PhD thesis, Faculty of the Graduate School, University of Colorado, 2007. Phd Proposal.

[7] Samuel Robert Reid. *Model Combination in Multiclass Classification.* PhD thesis, Faculty of the Graduate School, University of Colorado, 2010. Phd Thesis.

[8] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943. Reeditado en 1988 [176].

[9] Donald O. Hebb. *The Organization of Behavior, a Neuropsychological Theory.* John Wiley, New York, NY, USA, 1949. Reeditado en 1988 [176].

[10] H. D. Block. The perceptron: A model for brain functioning. i. *Reviews of Modern Physics*, 34:123–135, 1962. Reeditado en 1988 [176].

[11] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958. Reeditado en 1988 [176].

[12] Frank Rosenblatt. Two theorems of statistical separability in the perceptron. *Mechanzization of Thought Processes of a Symposium held at the National*

Physical Laboratory, 1:421–456, 1958.

[13] Marvin L. Minsky and Seimour A. Papert. *Perceptrons*. MIT Press, Cambridge, MA, USA, 1969. Reeditado y mejorado en 1988 [177].

[14] John J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. In *Proceedings of the National Academy of Sciences USA*, volume 81, pages 3088–3092, Mayo 1984.

[15] John J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences USA*, volume 79, pages 2554–2558, Abril 1986.

[16] David B. Parker. Learning logic. Technical Report TR-87, Center for Computational Research in Economics and Management Science, Massachusetts Institute of Technology, Cambridge, MA, USA, 1985.

[17] Yann LeCun. *Learning processes in an asymmetric threshold network*, pages 233–340. Springer-Verlag, Les Houches, France, 1986.

[18] Carlos A. Hernández Espinosa and Maria de las Mercedes Fernández Redondo. Apuntes del curso de doctorado "1241011 - redes neuronales", Curso académico 2003/2004.

[19] Mercedes Fernández Redondo. *Hacia un diseño óptimo de la Arquitectura Multilayer Feedforward*. PhD thesis, Universitat Jaume I, Septiembre 2001.

[20] Javier García de Jalón and José Ignacio-Rodríguez. *Aprenda Matlab 7.0 como si estuviera en primero*. Escuela Técnica Superior de Ingenieros Industriales. Universidad Politécnica de Madrid, 2005.

[21] Bernardo Cascales Salinas, Pascual Lucas Saorín, José Manuel Mira Ros, Antonio Pallares Ruiz, and Salvador Sanchez-Pedreño Guillen. *LaTeXuna imprenta en sus manos*. Aula Documental de Investigación, 1999.

[22] Bernardo Cascales Salinas, Pascual Lucas Saorín, José Manuel Mira Ros, Antonio Pallares Ruiz, and Salvador Sanchez-Pedreño Guillen. *El libro de LaTeX*. Pearson, Prentice Hall, 2003.

[23] Tomas Bautista, Tobias Oetiker, Hubert Partl, Irece Hyna, and Elisabeth Schlegl. Una descripción de latex 2_ε. Technical report, Universidad de las Palmas de Gran Canaria, 1998.

[24] Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[25] Alianna J. Maren, Craig T. Harston, and Robert M. Pap. *Handbook of Neural Computing Applications*. Academic Press, Inc., Orlando, FL, USA, 1990.

[26] Robert Hecht-Nielsen. *Neurocomputing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[27] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.

[28] Brian D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, UK, 1996.

[29] Nicolaos B. Karayiannis. Reformulated radial basis neural networks trained by gradient descent. *IEEE Transactions on Neural Networks*, 10(3):657–671, 1999.

[30] Nicolaos B. Karayiannis and Mary M. Randoph-Gips. On the construction and training of reformulated radial basis function neural networks. *IEEE Transactions on Neural Networks*, 14(4):835–846, 2003.

[31] John Moody and Christian J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:284–294, 1989.

[32] Asim Roy, Sandeep Govil, and Raymond Miranda. An algorithm to generate radial basis function (rbf)-like nets for classification problems. *Neural Networks*, 8(2):179–201, 1995.

[33] Asim Roy, Sandeep Govil, and Raymond Miranda. A neural-network learnign theory and polynomial time rbf algorithm. *IEEE Transactions on Neural Networks*, 8(6):1301–1313, 1997.

[34] Young-Sup Hwang and Sung-Yang Bang. An efficient method to construct a radial basis function neural network classifier. *Neural Networks*, 10(9):1495–1503, 1997.

[35] Friedhelm Schwenker, Hans A. Kestler, and Günther Palm. Combination of supervised and unsupervised learning for radial-basis-function networks. In H. H. Bothe and R. Rojas, editors, *Proceedings of the Second International ICSC Symposium on Neural Computation*, pages 22–28. ICSC Academic Press, Wetaskiwin, Canada, 2000.

[36] Friedhelm Schwenker, H.A. Kestler, and Günther Palm. Radial-basis-function networks: learning and applications. In *Knowledge-Based Intelligent Engineering Systems and Allied Technologies, 2000. Proceedings. Fourth International Conference on*, volume 1, pages 33–43, 2000.

[37] Mercedes Fernández-Redondo, Carlos Hernández-Espinosa, Mamen Ortiz-Gómez, and Joaquín Torres-Sospedra. Training radial basis functions by gradient descent. In Leszek Rutkowski, Jörg H. Siekmann, Ryszard Tadeusiewicz, and Lotfi A. Zadeh, editors, *Artificial Intelligence and Soft Computing - ICAISC 2004, 7th International Conference, Zakopane, Poland, June 7-11, 2004, Proceedings*, volume 3070 of *Lecture Notes in Computer Science*, pages 184–189. Springer, 2004. ISBN: 3-540-22123-9.

[38] Joaquín Torres-Sospedra, Carlos Hernández-Espinosa, and Mercedes Fernández-Redondo. An experimental study on training radial basis

functions by gradient descent. In Friedhelm Schwenker and Simone Marinai, editors, *Artificial Neural Networks in Pattern Recognition. Second IAPR Workshop, ANNPR 2006, Ulm, Germany, August 31-September 2, 2006. Proceedings*, volume 4087 of *Lecture Notes in Artificial Intelligence*, pages 81–92. Springer, 2006. ISBN: 978-3-540-37951-5 , ISSN: 0302-9743.

[39] Evelyn Fix and John Joseph Lawson Hodges. Nonparametric discrimination: Consistency properties. Technical report, USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[40] Josef Kittler, Mohamad Hatef, and Robert P. W. Duin. Combining classifiers. In *ICPR '96: Proceedings of the 13th International Conference on Pattern Recognition*, page 897, Washington, DC, USA, 1996. IEEE Computer Society.

[41] Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, March 1998.

[42] Ulf Johansson, Tuve Löfström, and Lars Niklasson. The importance of diversity in neural network ensembles - an empirical investigation. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2007, Celebrating 20 years of neural networks, Orlando, Florida, USA*, pages 661–666, 2007.

[43] Avrim Blum and Ronald L. Rivest. Training a 3-node neural network is np-complete. In *Proceedings of the first annual workshop on Computational learning theory.*, pages 9–18, San Francisco, CA, USA, 1988. Morgan Kaufmann Publishers Inc.

[44] Thomas G. Dietterich. Ensemble methods in machine learning. In Josef Kittler, editor, *Multiple Classifier Systems. First International Workshop, MCS 2000*, number 1857 in Lecture Notes in Computer Science, pages 1–15. Springer Berlin / Heidelberg, 2000. ISBN: 978-3-540-67704-8.

[45] Tin Kam Ho. Complexity of classification problems and comparative advantages of combined classifiers. In Josef Kittler and Fabio Roli, editors, *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 97–106. Springer, 2000.

[46] Tin Kam Ho. *Multiple Classifier Combination: Lessons and Next Steps*, volume 47 of *Series in Machine Perception and Artificial Intelligence*, chapter 7, pages 171–198. World Scientific, 2001.

[47] Kagan Tumer and Joydeep Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341–348, 1996.

[48] Kagan Tumer and Joydeep Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection Science*, 8(3-4):385–403, 1996.

[49] Thomas G. Dietterich. Ensemble methods in machine learning. In Josef Kittler

and Fabio Roli, editors, *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2000.

[50] Gasser Auda and Mohamed Kamel. Cmnn: Cooperative modular neural networks for pattern recognition. *Pattern Recognition Letters*, 18(11-13):1391 – 1398, 1997.

[51] Jakub Zavrel and Walter Daelemans. Bootstrapping a tagged corpus through combination of existing heterogeneous taggers. In *In Proceedings of the second international conference on language resources and evaluation (LREC-2000*, pages 17–20, 2000.

[52] H. Jair Escalante. Cleaning training-datasets with noise-aware algorithms. *Mexican International Conference on Computer Science*, 0:151–158, 2006.

[53] Mariette Awad, Xianhua Jiang, and Yuichi Motai. Incremental support vector machine framework for visual sensor networks. *EURASIP J. Appl. Signal Process.*, 2007(1):222–222, 2007.

[54] Md. Faijul Amin, Md. Monirul Islam, and Kazuyuki Murase. Ensemble of single-layered complex-valued neural networks for classification tasks. *Neurocomputing*, 72(10-12):2227 – 2234, 2009. Lattice Computing and Natural Computing (JCIS 2007) / Neural Networks in Intelligent Systems Designn (ISDA 2007).

[55] Thomas G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

[56] Yoram Reich and S. V. Barai. Evaluating machine learning models for engineering problems. *Artificial Intelligence in Engineering*, 13:257–272, 1999.

[57] Xuezhi Zheng, Zhihua Cai, and Qu Li. An experimental comparison of three kinds of clustering algorithms. In *Proceedigns of the International Conference on Neural Networks and Brain*, 2005.

[58] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition edition, 2005.

[59] Joaquín Torres-Sospedra. Conjuntos de redes neuronales (3 créditos). Master's thesis, Departamento de Ingenieria y Ciencia de los Computadores, 2005.

[60] Carlos Hernández-Espinosa, Mercedes Fernández-Redondo, and Joaquín Torres-Sospedra. First experiments on ensembles of radial basis functions. In Fabio Roli, Josef Kittler, and Terry Windeatt, editors, *Multiple Classifier Systems. 5th International Workshop, MCS 2004, Cagliari, Italy, June 9-11, 2004. Proceedings*, volume 3077 of *Lecture Notes in Computer Science*, pages 253–262. Springer, 2004. ISBN:978-3-540-22144-9.

[61] Yong Liu and Xin Yao. A cooperative ensemble learning system. In *Proceedings of the 1998 IEEE International Joint Conference on Neural Networks (IJCNN'98), Anchorage, USA, 4– 9 May 1998.*, pages 2202–2207, 1998.

[62] Bruce E. Rosen. Ensemble learning using decorrelated neural networks. *Connection Science*, 8(3-4):373–384, 1996.

[63] Gasser Auda and Mohdamed Kamel. Evol: Ensembles voting on-line. In *Proceedings of of the World Congress on Computational Intelligence, WCCI 1998, Anchorage, AK, USA*, volume 2, pages 1356–1360, 1998.

[64] Min Jang and Sungzoon Cho. Ensemble learning using observational learning theory. In *Proceedings of the International Joint Conference on Neural Networks, 1999. IJCNN '99.*, volume 2, pages 1287–1292. IEEE, 1999.

[65] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons Inc, June 1973.

[66] Nayer Wanas, Lovell Hodge, and Mohamed Kamel. An adaptive training algorithm for an ensemble of networks. In *International Joint Conference on Neural Networks, 2001. Proceedings. IJCNN '01.*, volume 4, pages 2590–2595. IEEE, 2001.

[67] Zbigniew Michalewicz. *Genetic algorithms + data structures = evolution programs*. Springer-Verlag New York, Inc., New York, NY, USA, third edition edition, 1996.

[68] Yong Liu, Xin Yao, and Tetsuya Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, November 2000.

[69] Gasser Auda and Mohdamed Kamel. Cmnn: cooperative modular neural network. In *International Conference on Neural Networks*, volume 1, pages 226–231 vol.1, Jun 1997.

[70] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007. University of California, Irvine, School of Information and Computer Sciences.

[71] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[72] Yuval Raviv and Nathan Intratorr. Bootstrapping with noise: An effective regularization technique. *Connection Science, Special issue on Combining Estimators*, 8:356–372, 1996.

[73] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 231–238. The MIT Press, 1995.

[74] Bambang Parmanto, Paul W. Munro, and Howard R. Doyle. Improving committee diagnosis with resampling techniques. In David S. Touretzky, Michael Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, November 27-30, 1995*, pages 882–888. MIT Press, 1996.

[75] Banbang Parmanto, Paul W. Munro, and Howard R. Doyle. Reducing variance of committee prediction with resampling techniques. *Connection Science*, 8:405–426(22), 1996.

[76] Antanas Verikas, Arunas Lipnickas, Kerstin Malmqvist, Marija Bacauskiene, and Adas Gelzinis. Soft combination of neural classifiers: A comparative study. *Pattern Recognition Letters*, 20(4):429–444, 1999.

[77] Rozita A. Dara and Mohamed S. Kamel. Sharing training patterns among multiple classifiers. In Roli et al. [178], pages 243–252.

[78] Michael Kearns. Thoughts on hypothesis boosting. ML class project, 1988.

[79] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[80] Harris Drucker, Corinna Cortes, Lawrence D. Jackel, Yann LeCun, and Vladimir Vapnik. Boosting and other ensemble methods. *Neural Computation*, 6(6):1289–1301, 1994.

[81] Leo Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–849, 1998.

[82] Ludmila Kuncheva and Christopher J. Whitaker. Using diversity with three variants of boosting: Aggressive, conservative and inverse. In *Proceedings International Workshop on Multiple Classifier Systems , Calgiari, Italy, June 2002. Springer.*, volume 2364 of *Lecture Notes in Computer Science*. Springer, 2002.

[83] Nikunj C. Oza. Boosting with averaged weight vectors. In Terry Windeatt and Fabio Roli, editors, *Multiple Classifier Systems*, volume 2709 of *Lecture Notes in Computer Science*, pages 15–24. Springer Berlin / Heidelberg, 2003. ISBN: 978-3-540-40369-2.

[84] Nikunj C. Oza. Aveboost2: Boosting for noisy data. In Roli et al. [178], pages 31–40. ISBN: 978-3-540-22144-9.

[85] Jyrki Kivinen and Manfred K. Warmuth. Boosting as entropy projection. In *In Proc. 12th Annu. Conference on Comput. Learning Theory*, pages 134–144. ACM Press, 1999.

[86] Ludmila I. Kuncheva. Error bounds for aggressive and conservative. In Terry Windeatt and Fabio Roli, editors, *Multiple Classifier Systems*, volume 2709 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2003.

[87] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[88] Robert E. Schapire. Theoretical views of boosting and applications. In *Proceedings of the 4th European Conference on Computational Learning Theory*, pages 1–10, 1999.

[89] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[90] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting,. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

[91] L. I. Kuncheva, M. Skurichina, and R. P. W. Duin. An experimental study on diversity for bagging and boosting with linear classifiers. *Information Fusion*, 3(4):245 – 258, 2002.

[92] Sherif Hashem. *Optimal Linear Combination of Neural Networks*. PhD thesis, School of Industrial Engineering, Purdue University, West Lafayette, Indiana, 1993.

[93] Sherif Hashem and Bruce Schmeiser. Improving model accuracy using optimal linear combinations of trained neural networks. *IEEE Transactions on Neural Networks*, 6:792–794, 1995.

[94] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[95] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(6):1289–1301, 1994.

[96] Daniel Jimenez, Thomas Darm, Bill Rogers, and Nicolas Walsh. Locating anatomical landmarks for prosthetics design using ensemble neural networks. In *Proceedings of the 1997 International Conference on Neural Networks, IJCNN'97*, 1997.

[97] Daniel Jimenez. Dynamically weighted ensemble neural networks for classification. In *Proceedings of the 1998 International Joint Conference on Neural Networks, IJCNN98*, pages 753–756, 1998.

[98] Nayer M. Wanas and Mohamed S. Kamel. Decision fusion in neural network ensembles. In *Proceedings of the 2001 International Joint Conference on Neural Networks, IJCNN01*, volume 4, pages 2952–2957, 2001.

[99] Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–75, 1994.

[100] Lei Xu, Adam Krzyzak, and Ching Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):418–435, 1992.

[101] Ron Kohavi and Foster Provost. Glossary of terms. *Machine Learning*, 30:271–274, 1998. Kluwer Academic Publishers, Boston, Manufactured in The Netherlands.

[102] Hossein Tahani and James M Keller. Information fusion in computer vision using the fuzzy integral. *IEEE Transactions on Systems, Man and Cybernetic*, 20:733–741, 1990.

[103] Sung-Bae Cho and Jin H. Kim. Combining multiple neural networks by fuzzy integral for robust classification. *IEEE Transactions on System, Man, and Cybernetics*, 25(2):380–384, 1995.

[104] Paul D. Gader, Magdi A. Mohamed, and James M. Keller. Fusion of handwritten word classifiers. *Pattern Recogn. Lett.*, 17(6):577–584, 1996.

[105] Michio Sugeno. *Fuzzy automata and decision processes*, chapter Fuzzy measures and fuzzy integrals: A survey, pages 89–102. Elsevier, 1977.

[106] Arvind S. Kumar, Samit K. Basu, and Kantial .L. Majumder. Robust classification of multispectral data using multiple neural networks and fuzzy integral. *IEEE Trans. on Geoscience and Remote Sensing*, 35(3):787–790, 1997.

[107] Ronald Robert Yager and Dimitar P. Filev. *Fuzzy Sets, Neural Networks, and Soft Computing,*, chapter On a Flexible Structure for Fuzzy Systems Models, pages 1–28. Van Nostrand Reinhold, New York, 1994.

[108] Hans-Jürgen Zimmermann and Peter Zysno. Decision and evaluations by hierarchical aggregation of information. *Fuzzy Sets and Systems*, 10(3):243–260, 1984.

[109] Raghu Krishnapuram and Joonwhoan Lee. Fuzzy-set-based hierarchical networks for information fusion in computer vision. *Neural Networks*, 5(2):335–350, 1992.

[110] Christian Jutten, Anne Guerin-Dugue, Carlos Aviles-Cruz, Jean Luc Voz, and Dominique Van Cappel. Basic research esprit project number 6891: Elena, enhanced learning for evolutive neural architectures, 1995. Anonymous FTP: ftp://ftp.dice.ucl.ac.be/pub/neural-nets/ELENA/databases.

[111] Youngjik Lee, Sang-Hoon Oh, and Myung Won Kim. The effect of initial weights on premature saturation in back-propagation learning. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, volume i, pages 765–770 vol.1, Jul 1991.

[112] Javier E. Vitela and Jaques Reifman. The causes for premature saturation with backpropagation training. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, volume 3, pages 1449–1453 vol.3, Jun-2 Jul 1994.

[113] Ali A. Ghorbani and Kiarash Owrangh. Stacked generalization in neural networks: Generalization on statistically neutral problems. In *Proceedings of the International Joint conference on Neural Networks, IJCNN 2001, Washington D.C., USA.*, pages 1715–1720. IEEE, 2001.

[114] Kai Ming Ting and Ian H. Witten. Stacked generalizations: When does it

work? In *International Joint Conference on Artificial Intelligence proceedings, Volume 2.*, pages 866–873, 1997.

[115] Kai Ming Ting and Ian H. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, 1999.

[116] Kai Ming Ting and Ian H. Witten. Stacking bagged and dagged models. In *In Proc. 14th International Conference on Machine Learning*, pages 367–375. Morgan Kaufmann, 1997.

[117] Agapito Ismael Ledezma Espino. *Aprendizaje Automático en Conjuntos de Clasificadores Heterogéneos y Modelado de Agentes.* PhD thesis, Escuela Politecnica Superior, Universidad Carlos III de Madrid, 2004.

[118] Marcos Faúndez-Zanuy. Nonlinear speech processing: Overview and possibilities in speech coding. In Gérard Chollet, Anna Esposito, Marcos Faúndez-Zanuy, and Maria Marinaro, editors, *Summer School on Neural Networks*, volume 3445 of *Lecture Notes in Computer Science*, pages 15–42. Springer, 2004.

[119] Marcos Faúndez Zanuy. Study of a committee of neural networks for biometric hand-geometry recognition. In Joan Cabestany, Alberto Prieto, and Francisco Sandoval Hernández, editors, *Computational Intelligence and Bioinspired Systems, 8th International Work-Conference on Artificial Neural Networks, IWANN 2005, Vilanova i la Geltrú, Barcelona, Spain, June 8-10, 2005, Proceedings*, volume 3512 of *Lecture Notes in Computer Science*, pages 1180–1187. Springer, 2005.

[120] Steven J. Nowlan and Geoffrey E. Hinton. Evaluation of adaptive mixtures of competing experts. In *NIPS-3: Proceedings of the 1990 conference on Advances in neural information processing systems 3*, pages 774–780, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

[121] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer, August 2006.

[122] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. Technical Report AIM-1440, Massachusetts Institute of Technology, 1993.

[123] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[124] Steve R. Waterhouse and Anthony J. Robinson. Classification using hierarchical mixtures of experts. In *Proceedings of the 1994 IEEE Workshop Neural Networks for Signal Processing*, pages 177–186, 1994.

[125] Ke Chen, Lei Xu, and Huisheng Chi. Improved learning algorithms for mixture of experts in multiclass classification. *Neural Networks*, 12(9):1229–1252, 1999.

[126] Elena Prades-Carceller. Construcción de un conjunto de Redes Neuronales

mediante Mezcla de Redes Expertas y Radial Basis Functions. Proyecto final de carrera de ingeniería informática (plan 2001), Universistat Jaume I, 2008.

[127] Sherif Hashem. Optimal linear combinations of neural networks. *Neural Networks*, 10(4):599–614, 1997.

[128] Conferences ranking. Web published, http://www.cs-conference-ranking.org/conferenceranking/topicsii.html, February 2009.

[129] Excellence in Research for Australia. Ranked conference list. web, 2010. http://www.arc.gov.au/era/era_journal_list.htm.

[130] Joachim Utans, Jonh Moody, Steve Rehfuss, and Hava Siegelmann. Input variable selection for neural networks: application to predicting the u.s. business cycle. In *Proceedings of the IEEE/IAFE Computational Intelligence for Financial Engineering*, pages 118–122, Apr 1995.

[131] Javier Lorenzo-Navarro, Francisco Mario Hernández-Tejera, and Juan Méndez-Rodríguez. Selección de atributos mediante una medida basada en información mutua. In *Proceedings of Conferencia de la Asociación Española para la Inteligencia Artificial, CAEPIA*, pages 469–478, 1997.

[132] Igor V. Tetko, Vsevolod Yu. Tanchuk, and Alexander I. Luik. Simple heuristic methods for input parameter estimation in neural networks. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 425–430, 1994.

[133] Igor V. Tetko, Alessandro E. P. Villa, and David J. Livingstone. Neural network studies, 2. variable selection. *Journal of Chemical Information and Computer Sciences*, 36(4):794–803, 1996.

[134] Lisa M. Belue and Kenneth W. Bauer. Determining input features for multilayer perceptrons. *Neurocomputing*, 7(2):111 – 121, 1995.

[135] Luisa Devena. Automatic selection of the most relevant features to recognize objects. In *Proceedings of the International Conference on Artificial Neural Networks*, volume 2, pages 1113–1116, 1994.

[136] Wael El-Deredy and Neil M. Branston. Identification if relevant features in HMR tumor spectra using neural networks. In *Proceedings of the 4th International Conference on Artificial Neural Networks*, pages 454–458, 1995.

[137] Jianchang Mao, K. Mohiuddin, and A.K. Jain. Parsimonious network design and feature selection through node pruning. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Computer Vision & Image Processing*, volume 2, pages 622–624 vol.2, Oct 1994.

[138] Andries P. Engelbrecht and Ian Cloete. A sensitivity analysis algorithm for pruning feedforward neural networks. In *IEEE International Conference in Neural Networks*, volume 1, pages 1274–1277, 1996.

[139] A. Bowles. Machine learns which features to select. In *Proceedings of the 5th*

Australian Joint Conference on Artificial Intelligence, pages 127–132, 1992.

[140] Younès Bennani and Fabrice Bossaert. A neural network based variable selector. In *Proceedings of the Artificial Neural Network in Engineering*, pages 425–430, 1995.

[141] Roberto Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5:537–550, 1994.

[142] Carlos Eduardo Pedreira, Leonardo Macrini, and Elaine S. Costa. Input and data selection applied to heart disease diagnosis. In *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, volume 4, pages 2389–2393 vol. 4, 31 2005-Aug. 4 2005.

[143] Mohammad Mahdi Rezaei Yousefi, Masoud Mirmomeni, and Caro Lucas. Input variables selection using mutual information for neuro fuzzy modeling with the application to time series forecasting. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2007, Celebrating 20 years of neural networks, Orlando, Florida, USA, August 12-17, 2007*, pages 1121–1126. IEEE, 2007.

[144] Paul J. Werbos. Backpropagation: past and future. In *IEEE International Conference on Neural Networks, 1988*, pages 343–353 vol.1, Jul 1988.

[145] Tautvydas Cibas, Françoise Fogelman Soulié, Patrick Gallinari, and Sarunas Raudys. Variable selection with neural networks. *Neurocomputing*, 12:223–248, 1996.

[146] Y.K Kim and J.B Ra. Weight value initialization for improving training speed in the backpropagation network. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, pages 2306–2401, 1991.

[147] Hisashi Shimodaira. A weight value initialization method for improving learning performance of the back propagation algorithm in neural networks. In *ICTAI*, pages 672–675, 1994.

[148] G. Palubinskas. Data-driven weight initialization of back-propagation for pattern recognition. In *Proceedings of the International Conference on Artificial Neural Networks*, volume 2, pages 851–854, 1994.

[149] Yoshio Hirose, Koichi Yamashita, and Shimpei Hijiya. Back-propagation algorithm which varies the number of hidden units. *Neural Netw.*, 4(1):61–66, 1991.

[150] Takio Kurita, Hideki Asoh, Shinji Umeyama, Shotaro Akaho, and Akitaka Hosomi. A structural learning by adding independent noises to hidden units. In *Proceedings of the IEEE International Conference on Neural Networks, IEEE World Congress on Computational Intelligence 1994*, volume 1, pages 275–278 vol.1, Jun-2 Jul 1994.

[151] Matthias Rychetsky, Stefan Ortmann, and Manfred Glesner. Pruning and

regularization techniques for feed forward nets applied on a real world data base. In Michael Heiss, editor, *Proceedings of the International ICSC / IFAC Symposium on Neural Computation (NC 1998), September 23-15, 1998, Vienna, Austria*, pages 603–609. ICSC Academic Press, International Computer Science Conventions, Canada / Switzerland, 1998.

[152] Jyh-Shyan Lin, Shih-Chung B. Lo, Akira Hasegawa, Matthew Freedman, and Seong K. Mun. Reduction of false positives in lung nodule detection using a two-level neural classification. *IEEE Transactions on Medical Imaging*, 15(2):206–217, Apr 1996.

[153] L. Zhang, W. Qian, R. Sankar, D. Song, and R. Clark. A new false positive reduction method for mccs detection in digital mammography. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing 2001*, volume 2, pages 1033–1036, 2001.

[154] Georgia D. Tourassi, Nevine H. Eltonsy, James H. Graham, Carey E. Floyd jr., and Adel S. Elmaghraby. Feature and knowledge based analysis for reduction of false positives in the computerized detection of masses in screening mammography. In *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, pages 6524–6527, Jan. 2005.

[155] Kenji Suzuki, Mark Epstein, Ivan Sheu, Ryan Kohlbrenner, Don C. Rockey, and Abraham H. Dachman. Massive-training artificial neural networks for cad for detection of polyps in ct colonography: False-negative cases in a large multicenter clinical trial. In *Proceedings of the 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro, 2008.*, pages 684–687, May 2008.

[156] Kaspersky Labs. Spam evolution: January 2010. Web published. http://www.securelist.com/en/analysis?pubid=%20204792103.

[157] Minoru Sasaki and Hiroyuki Shinnou. Spam detection using text clustering. In *Proceedings of the International Conference on Cyberworlds, 2005*, pages 316–319, November 2005.

[158] Chun-Chao Yeh and Soun-Jan Chiang. Revisit bayesian approaches for spam detection. In *Proceedings of the 9th International Conference for Young Computer Scientists, ICYCS 2008, Zhang Jia Jie, Hunan, China, November 18-21, 2008*, pages 659–664. IEEE Computer Society, 2008.

[159] Zhen Yang, Xiangfei Nie, Weiran Xu, and Jun Guo. An approach to spam detection by naive bayes ensemble based on decision induction. In *Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications*, pages 861–866, Washington, DC, USA, 2006. IEEE Computer Society.

[160] Bo Yu and Dong hua Zhu. Combining neural networks and semantic feature

space for email classification. *Know.-Based Syst.*, 22(5):376–381, 2009.

[161] Hung Nguyen, W.T. Hung, B.S. Thornton, E. Thornton, and W. Lee. Classification of microcalcifications in mammograms using artificial neural networks. In *Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Societ*, volume 20. IEEE, 1998.

[162] Michael E. Mavroforakis, Harris V. Georgiou, Dionisis Cavouras, Nikos Dimitropoulos, and Sergios A. Theodoridis. Mammographic mass classification using textural features and descriptive diagnostic data. In *Digital Signal Processing, 2002. DSP 2002. 2002 14th International Conference on*, volume 1, pages 461–464 vol.1, 2002.

[163] Yuan Chen and Chein-I Chang. A new application of texture unit coding to mass classification for mammograms. In *Proceedings of the International Conference on Image Processing, 2004.*, pages 3335–3338, 2004.

[164] R. J. Nandi, Asoke K. Nandi Nandi, R. Rangayyan, and Diane Scutt. Genetic programming and feature selection for classification of breast masses in mammograms. In *Proceedings of the 28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS '06*, pages 3021–3024, New York, USA, August 2006. IEEE.

[165] Konrad Bojar and Mariusz Nieniewski. New features for classification of cancerous masses in mammograms based on morphological dilation. In *Visual Information Engineering, 2008. VIE 2008. 5th International Conference on*, pages 111–116, 29 2008-Aug. 1 2008.

[166] Gi-Yeul Sung, Dong-Min Kwak, and Joon Lyou. Neural network based terrain classification using wavelet features. *Journal of Intelligent and Robotic Systems*, 59(3-4):269–281, 2010.

[167] H. Altay Guvenir, Burak Acar, Gulsen Demiroz, and Ayhan C Ekiny. A supervised machine learning algorithm for arrhythmia analysis. In *In Proceedings Computers in Cardiology Conference*, pages 433–436, 1997.

[168] Robert Siegler. Three aspects of cognitive development. *Cognitive Psychology*, 8:481–520, 1976.

[169] Bob Evans and Doug Fisher. Overcoming process delays with decision tree induction. *IEEE Expert: Intelligent Systems and Their Applications*, 9(1):60–66, 1994.

[170] Richard S. Forsyth. PC/BEAGLE User's Guide.

[171] Kenta Nakai and Minoru Kanehisa. Expert sytem for predicting protein localization sites in gram-negative bacteria. *Structure, Function, and Genetics*, 11:95–110, 1991.

[172] Kenta Nakai and Minoru Kanehisa. A knowledge base for predicting protein localization sites in eukaryotic cells. *Genomics*, 14:897–911, 1991.

[173] Paul Horton and Kenta Nakai. A probablistic classification system for predicting the cellular localization sites of proteins. *Intelligent Systems in Molecular Biology*, 14:109–115, 1996.

[174] V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10:262–266, 1989.

[175] S.B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. De Jong, S. Dzeroski, S.E. Fahlman, D. Fisher, R. Hamann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, R.S. Michalski, T. Mitchell, P. Pachowicz, Y. Reich H. Vafaie, W. Van de Welde, W. Wenzel, J. Wnek, and J. Zhang. The monk's problems - a performance comparison of different learning algorithms. Technical Report CS-CMU-91-197, Carnegie Mellon University, December 1991.

[176] James A. Anderson and Edward Rosenfeld, editors. *Neurocomputing: foundations of research*. MIT Press, Cambridge, MA, USA, 1988.

[177] Marvin L. Minsky and Seymour A. Papert. *Perceptrons: expanded edition*. MIT Press, Cambridge, MA, USA, 1988.

[178] Fabio Roli, Josef Kittler, and Terry Windeatt, editors. *Multiple Classifier Systems, 5th International Workshop, MCS 2004, Cagliari, Italy, June 9-11, 2004, Proceedings*, volume 3077 of *Lecture Notes in Computer Science*. Springer, 2004.