UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

This thesis is submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Ph.D.)

Deep Learning for i-Vector Speaker and Language Recognition

Author:

Omid Ghahabi Esfahani

Thesis advisor:

Prof. Francisco Javier Hernando Pericás

TALP Research Center
Department of Signal Theory and Communications
Technical University of Catalonia, UPC

Barcelona, April 2018

Abstract

Over the last few years, i-vectors have been the state-of-the-art technique in speaker and language recognition. Recent advances in Deep Learning (DL) technology have improved the quality of i-vectors but the DL techniques in use are computationally expensive and need speaker or/and phonetic labels for the background data, which are not easily accessible in practice. On the other hand, the lack of speaker-labeled background data makes a big performance gap, in speaker recognition, between two well-known cosine and Probabilistic Linear Discriminant Analysis (PLDA) i-vector scoring techniques. It has recently been a challenge how to fill this gap without speaker labels, which are expensive in practice. Although some unsupervised clustering techniques are proposed to estimate the speaker labels, they cannot accurately estimate the labels.

This thesis tries to solve the problems above by using the DL technology in different ways, without any need of speaker or phonetic labels. In order to fill the performance gap between cosine and PLDA scoring given unlabeled background data, we have proposed an impostor selection algorithm and a universal model adaptation process in a hybrid system based on Deep Belief Networks (DBNs) and Deep Neural Networks (DNNs) to discriminatively model each target speaker. In order to have more insight into the behavior of DL techniques in both single and multi-session speaker enrollment tasks, some experiments have been carried out in both scenarios. Experiments on the National Institute of Standard and Technology (NIST) 2014 i-vector challenge show that 46% of this performance gap, in terms of minDCF, is filled by the proposed DL-based system. Furthermore, the score combination of the proposed DL-based system and PLDA with estimated labels covers 79% of this gap.

In the second line of the research, we have developed an efficient alternative vector representation of speech by keeping the computational cost as low as possible and avoiding phonetic labels, which are not always accessible. The proposed vectors will be based on both Gaussian Mixture Models (GMMs) and Restricted Boltzmann Machines (RBMs) and will be referred to as GMM-RBM vectors. The role of RBM is to learn the total speaker and session variability among background GMM supervectors. This RBM, which will be referred to as Universal RBM (URBM), will then be used to transform unseen supervectors to the proposed low dimensional vectors. The use of different activation functions for training the URBM and different transformation functions for extracting the proposed vectors are investigated. At the end, a variant of Rectified Linear Unit (ReLU) which is referred to as Variable ReLU (VReLU) is proposed. Experiments on the core test condition 5 of the NIST Speaker Recognition Evaluation (SRE) 2010 show that comparable results with conventional i-vectors are achieved with a clearly lower computational load in the vector extraction process.

Finally, for the Language Identification (LID) application, we have proposed a DNN architecture to model effectively the i-vector space of four languages, English, Spanish, German, and Finnish, in the car environment. Both raw i-vectors and session variability compensated i-vectors are evaluated as input vectors to DNN. The performance of the proposed DNN architecture is compared with both conventional GMM-UBM and i-vector/Linear Discriminant Analysis (LDA) systems considering the effect of duration of signals. It is shown that the signals with duration between 2 and 3 sec meet the accuracy and speed requirements of this application, in which the proposed DNN architecture outperforms GMM-UBM and i-vector/LDA systems by 37% and 28%, respectively.

Resumen

En los últimos años, los i-vectores han sido la técnica de referencia en el reconocimiento de hablantes y de idioma. Los últimos avances en la tecnología de Aprendizaje Profundo (Deep Learning. DL) han mejorado la calidad de los i-vectores, pero las técnicas DL en uso son computacionalmente costosas y necesitan datos etiquetados para cada hablante y/o unidad fonética, los cuales no son fácilmente accesibles en la práctica. La falta de datos etiquetados provoca una gran diferencia de los resultados en el reconocimiento de hablante con i-vectors entre las dos técnicas de evaluación más utilizados: distancia coseno y Análisis Lineal Discriminante Probabilístico (PLDA). Por el momento, sigue siendo un reto cómo reducir esta brecha sin disponer de las etiquetas de los hablantes, que son costosas de obtener. Aunque se han propuesto algunas técnicas de agrupamiento sin supervisión para estimar las etiquetas de los hablantes, no pueden estimar las etiquetas con precisión.

Esta tesis trata de resolver los problemas mencionados usando la tecnología DL de diferentes maneras, sin necesidad de etiquetas de hablante o fonéticas. Con el fin de reducir la diferencia de resultasos entre distancia coseno y PLDA a partir de datos no etiquetados, hemos propuesto un algoritmo selección de impostores y la adaptación a un modelo universal en un sistema híbrido basado en Deep Belief Networks (DBN) y Deep Neural Networks (DNN) para modelar a cada hablante objetivo de forma discriminativa. Con el fin de tener más información sobre el comportamiento de las técnicas DL en las tareas de identificación de hablante en una única sesión y en varias sesiones, se han llevado a cabo algunos experimentos en ambos escenarios. Los experimentos utilizando los datos del National Institute of Standard and Technology (NIST) 2014 i-vector Challenge muestran que el 46% de esta diferencia de resultados, en términos de minDCF, se reduce con el sistema propuesto basado en DL. Además, la combinación de evaluaciones del sistema propuesto basado en DL y PLDA con etiquetas estimadas reduce el 79% de esta diferencia.

En la segunda línea de la investigación, hemos desarrollado una representación vectorial alternativa eficiente de la voz manteniendo el coste computacional lo más bajo posible y evitando las etiquetas fonéticas, Los vectores propuestos se basan tanto en el Modelo de Mezcla de Gaussianas (GMM) y en las Máquinas Boltzmann Restringidas (RBM), a los que se hacer referencia como vectores GMM-RBM. El papel de la RBM es aprender la variabilidad total del hablante y de la sesión entre los supervectores del GMM genérico. Este RBM, al que se hará referencia como RBM Universal (URBM), se utilizará para transformar supervectores ocultos en los vectores propuestos, de menor dimensión. Además, se esttuida el uso de diferentes funciones de activación para el entrenamiento de la URBM y diferentes funciones de transformación para extraer los vectores propuestos. Finalmente, se propone una variante de la Unidad Lineal Rectificada (ReLU) a la que se hace referencia como Variable ReLU (VReLU). Los experimentos sobre los datos de la condición 5 del test de la NIST Speaker Recognition Evaluation (SRE) 2010 muestran que se han conseguido resultados comparables con los i-vectores convencionales, con una carga computacional claramente inferior en el proceso de extracción de vectores.

Por último, para la aplicación de Identificación de Idioma (LID), hemos propuesto una arquitectura DNN para modelar eficazmente en el entorno del coche el espacio i-vector de cuatro idiomas: inglés, español, alemán y finlandés. Tanto los i-vectores originales como los i-vectores propuestos son evaluados como vectores de entrada a DNN. El rendimiento de la arquitectura DNN propuesta se compara con los sistemas convencionales GMM-UBM y i-vector/Análisis Discriminante Lineal (LDA) considerando el efecto de la duración de las señales. Se muestra que en caso de señales con una duración entre 2 y 3 se obtienen resultados satisfactorios en cuanto a precisión y resultados, superando a los sistemas GMM-UBM y i-vector/LDA en un 37% y 28%, respectivamente.

Acknowledgements

First of all, I would like to thank Prof. Javier Hernando for his excellent supervision. This thesis work would not have been accomplished without his valuable guidance, encouragement, and support. I learned, from whom, not only how to do research correctly and fulfill scientific contributions, but also many precious life lessons. I will never forget our conference trips to Joensuu, Las Palmas de Gran Canaria, and San Francisco along with the thesis discussions. I feel very grateful and fortunate for being able to spent this time under his tutorship.

I am grateful then to my ex-supervisor, from my former university in Tehran, Prof. M.H. Savoji who interested me in Speech Technology and stood at the beginning of this adventure.

I would like to thank Martin Zelenak and Jordi Luque for their great supports at the beginning of this work, and to Diego Lendoiro and Carlos Nistal for providing the excellent technical support. I am very grateful to Climent Nadeu, Antonio Bonafonte, and Asuncion Moreno for their helpful advice and collaborations during these years. I would like to thank all other colleagues from the TALP research center, specifically Anna Raboshchuk, Pooyan Safari, Martin Wolf, Rupayan Chakraborty, Abraham Woubie, Miquel India, Enric Monte, and Adrian Rodriguez-Fonollosa, for the enjoyable research environment, and the colleagues from other research groups in my office D5-120, Jaime Gallego, Marc Maceira, Marc Torrellas, Pau Bellot, David Varas, Jordi Pont, Carles Ventura, Guillem Palou, as well as all others who I might have forgotten to mention. I would like to give special thanks to Jaime Gallego and Lucia Maio who were always supporting me in difficult situations along this experience. Thanks also to all other friends being beside me during these years, specifically Javad Zolfaghari, Vahid Joroughi, and Parastoo Nasr.

I finally want to express my deep gratitude to my parents Nasrin and Mehdi, my sister Elnaz, and my brother Amir for their continuous encouragement, love, and support.

Contents

List of Acronyms

ASR	Automatic Speech Recognition
CD	Contrastive Divergence
CDF	Cumulative Distribution Function
CMN	Cepstral Mean Normalization
CMVN	Cepstral Mean and Variance Normalization
CNN	Convolutional Neural Network
DBM	Deep Boltzmann Machine
DBN	Deep Belief Network
DCF	Detection Cost Function
DCT	Discrete Cosine Transform
DET	Detection Error Tradeoff
DL	Deep Learning
DNN	Deep Neural Network
EER	Equal Error Rate
EM	Expectation-Maximization
FAR	False Acceptance Rate
FBE	Filter Bank Energy
FF	Frequency Filtering
FRR	False Rejection Rate
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
JFA	Joint Factor Analysis
LDA	Linear Discriminant Analysis
LFCC	Linear Frequency Cepstral Coefficient
LID	Language Identification
LPC	Linear Predictive Coefficient

LPR	Log Posterior Ratio
LSTM	Long Short-Term Memory
MAP	Maximum a Posteriori
MFCC	Mel-Frequency Cepstral Coefficient
minDCF	minimum DCF
NAP	Nuisance Attribute Projection
NIST	National Institute of Standard and Technology
PCA	Principal Component Analysis
PDF	Probability Density Function
PLDA	Probabilistic Linear Discriminant Analysis
PLP	Perceptual Linear Predictive
RBM	Restricted Boltzmann Machine
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SDC	Shifted Delta Cepstrum
SRE	Speaker Recognition Evaluation
SVM	Support Vector Machine
UBM	Universal Background Model
UDBN	Universal DBN
URBM	Universal RBM
VAD	Voice Activity Detection
VReLU	Variable ReLU
WCCN	Within-Class Covariance Normalization

List of figures

List of tables

Chapter 1

Introduction

Speaker recognition is the process of automatically identifying who is speaking, i.e., speaker identification, or verifying the speaker identities being claimed by individuals, i.e., speaker verification. All the recognition process is only based on the speech signals captured from the speakers. In other word, it is supposed that everyone has a unique voice which could be used as an identity rather than or in addition to other identities like fingerprint, face, iris, etc. In some applications, it is more convenient to use the voice as an identity, e.g., access to the bank accounts through the mobile apps when the luminance of the environment is not suitable for using face or iris. Similarly, language recognition refers to an automatic process through which the language spoken in a speech signal is determined or verified. Several technologies are shared between speaker and language recognition. Hence, the proposed ideas in one application can be also used in another. This thesis focuses mainly on speaker verification and partially on language identification.

Historically, both applications have faced many challenges over the time. On the other hand, demands for having higher accuracy, faster recognition, and more robustness against changing the environment have led to non-stop investigations as in other applications. In the following, the motivations and the objectives of the thesis are first described and then the outline of the following chapters is briefly given.

1.1 Motivation

The success use of Deep Learning (DL) in a large variety of signal processing applications, particularly in speech processing (e.g., Mohamed, Yu, & Deng, 2010; Dahl, Yu, Deng, & Acero, 2012; Mohamed, Dahl, & Hinton, 2012; Hinton et al., 2012; Senior, Sak, & Shafran, 2015), has inspired the community to make use of DL techniques in speaker recognition as well. Both generative approaches, like Restricted Boltzmann Machine (RBM) and Deep Belief Network (DBN), and discriminative ones, like Deep Neural Network (DNN), have been used for this purpose. A possible use of DL techniques in speaker recognition is to combine them with the state-of-the-art i-vector (Dehak, Kenny, Dehak, Dumouchel, & Ouellet, 2011)—a compact representation of characteristics of a speech signal which is widely used in both text-independent speaker and language recognition tasks. Two kinds of combination have been considered. DL techniques have been used in the i-vector extraction process (Lei, Scheffer, Ferre, & Mclaren, 2014; Kenny, Gupta, Stafylakis, Ouellet, & Alam, 2014; Mclaren, Lei, & Ferre, 2015; Richardson, Reynolds, & Dehak, 2015a; Liu et al., 2015) or applied on i-vectors as a backend (Stafylakis, Kenny, Senoussaoui, & Dumouchel, 2012b; Senoussaoui, Dehak, Kenny, Dehak, & Dumouchel, 2012; Stafylakis, Kenny, Senoussaoui, & Dumouchel, 2012a; Novoselov, Pekhovsky, Kudashev, Mendelev, & Prudnikov, 2015; Pekhovsky, Novoselov, Sholokhov, & Kudashev, 2016; Villalba, Brümmer, & Dehak, 2017).

DL technology has been used in the i-vector extraction algorithm in two ways. First, a DNN has been used for acoustic modeling rather than the typical Gaussian Mixture Model (GMM) (Lei et al., 2014; Kenny et al., 2014; W. M. Campbell, 2014; Richardson et al., 2015a; Garcia-Romero, Zhang, McCree, & Povey, 2014; Liu et al., 2015). Second, conventional spectral features have been replaced or appended by the so-called DNN bottleneck features and then a DNN or a GMM has been used as an acoustic model (Mclaren et al., 2015; Richardson et al., 2015a; Liu et al., 2015). It has been shown that the best results are obtained when spectral features are appended by bottleneck features and a GMM is used as an acoustic model (Mclaren et al., 2015; Richardson et al., 2015a; Lozano-Diez et al., 2016). However, the main problem is that the use of DNN as either an acoustic model or bottleneck feature extractor increases highly the computational cost of the i-vector extraction process. Moreover, in both cases phonetic labels are required for DNN training, which are not always accessible.

Besides, after i-vector computation, DL techniques can be used for different purposes. For example, different combinations of RBMs have been proposed in (Stafylakis et al., 2012b; Senoussaoui et al., 2012) to classify i-vectors and in (Stafylakis et al., 2012a) to learn speaker and channel factor subspaces. RBMs in (Novoselov, Pekhovsky, Simonchik, & Shulipa, 2014) and DNNs in (Isik, Erdogan, & Sarikaya, 2015) have been used to increase the discrimination power of i-vectors. A nonlinear Probabilistic Linear Discriminant Analysis (PLDA) is simulated in (Villalba et al., 2017) using a tied variational autoencoder architecture. In (Novoselov et al., 2015; Pekhovsky et al., 2016), a combination of RBM, autoencoder, and PLDA is proposed for speaker and channel variability compensation, which shows some improvements compared to using only PLDA. All of these techniques have been proposed somehow as an alternative backend to the powerful PLDA (Prince & Elder, 2007; Kenny, 2010). Nevertheless, as for PLDA, a large amount of speaker labeled background data is required. Usually, a large number of different speakers with several speech utterances each are necessary for these techniques to work efficiently. Access to the speaker labeled data is costly and in some cases almost impossible. One of the recent challenges in speaker recognition, which was organized by the National Institute of Standard and Technology (NIST), has been how to achieve a comparable performance with PLDA when no labeled background data is available (NIST, 2014). Although some unsupervised automatic labeling techniques have been proposed (Khoury, El Shafey, Ferras, & Marcel, 2014; Novoselov, Pekhovsky, & Simonchik, 2014), they cannot appropriately estimate the true labels and also they assume that there are several samples from a same speaker in the background data which could not be true in reality.

Another possible use of DL is to represent the speaker characteristics of a speech signal with a single low dimensional vector using a DL architecture, rather than the traditional i-vector algorithm. These vectors are often referred to as speaker embeddings which could be computed by supervised or unsupervised techniques. The DL architectures in the supervised techniques are usually trained given the speaker-labeled background data. Typically, the inputs of the neural networks are a sequence of feature vectors and the outputs are speaker classes. Different architectures, activation functions, and training procedures have been proposed (e.g., Variani, Lei, McDermott, Lopez Moreno, & Gonzalez-Dominguez, 2014; Liu et al., 2015; Wang, Qian, & Yu, 2017; Bhattacharya, Alam, & Kenny, 2017; Snyder, Garcia-Romero, Povey, & Khudanpur, 2017). The experimental results have shown that, in most cases, the bigger improvements are obtained on the shorter signals compared to traditional i-vectors (Bhattacharya et al., 2017; Snyder et al., 2017), which implies that DL technology can model better the speaker characteristics of a short-duration speech signal than the traditional signal processing techniques. However, the need of speaker labels for training the network is one of the disadvantages of these techniques. Moreover, speaker embeddings extracted from hidden layer outputs are not so compatible with PLDA backend as the posterior distribution of hidden layer outputs are usually not truly Gaussian. On the other hand, only a few works have used unsupervised techniques for extraction of speaker embeddings (e.g., Vasilakakis, Cumani, & Laface, 2013). The background data in these techniques is free of any kind of labels, which can be considered one of the advantages of these techniques. Although some success has been shown for supervised speaker embeddings, mainly for very short utterances, still no significant improvement is reported for unsupervised speaker embeddings.

As in speaker recognition, DL architectures have been used in the i-vector extraction process for language recognition as well (e.g., Song, Hong, et al., 2015; Richardson, Reynolds, & Dehak, 2015b) with the cost of increasing the computational time which is important for real-time applications. Moreover, DL has been applied after i-vector computation for classification (Matějka et al., 2012; Matejka et al., 2014; Ferrer, Lei, McLaren, & Scheffer, 2016b). However, DNNs with only one hidden layer are mainly used, keeping this question open how well other DL architectures will perform for this purpose. There have been also some efforts to discriminate different languages by using DL directly on the feature vectors (e.g., Lopez-Moreno et al., 2014; Gonzalez-Dominguez, Lopez-Moreno, Sak, Gonzalez-Rodriguez, & Moreno, 2014). Processing the feature vectors with DL leads usually to higher accuracy but with much more computational cost in both training and test.

1.2 Objectives

As discussed in the previous section, the main question will be how one can take advantage of the recent advances in DL while decreasing the cost of labeled background data. In this thesis, two major objectives are followed in speaker recognition and one in language recognition to answer the aforementioned question. For language recognition, we will deal with an extra challenge which is working with short utterances spoken in the car environment. The more concrete aims are given as follows.

I. To make use of deep architectures for backend i-vector classification in order to fill the performance gap between the cosine (unlabeled-based) and PLDA (labeled-based) scoring baseline systems given unlabeled background data. We take advantage of unsupervised learning of DBNs to train a global model referred to as Universal DBN (UDBN) and DNN supervised learning to model each target speaker discriminatively. To provide a balanced training, an impostor selection algorithm and to cope with few training data, a UDBN-adaptation process is proposed. We explore particularly hybrid deep architectures with different number of layers for both single and multi-session speaker enrollment tasks. The preliminary experiments are performed on NIST Speaker Recognition Evaluation (SRE) 2006 (NIST, 2006) to show the effect of each contribution. Taking advantage of the conclusions obtained on the preliminary experiments, another set of experiments are carried out on the newer and more challenging database NIST 2014 i-vector challenge (NIST, 2014). Experimental results performed on this challenge show that the proposed DL-based system fills 46% of the performance gap between cosine and oracle PLDA scoring systems in terms of minimum DCF (minDCF) which is similar to the PLDA scoring results obtained with unsupervised estimated labels. The score combination of the proposed DL-based system and PLDA with estimated labels fills 79% of this gap.

II. To develop an efficient framework for vector representation of speech by keeping the computational cost as low as possible and avoiding speaker and phonetic labels. In order to achieve this goal, a global RBM referred to as Universal RBM (URBM) is trained given background GMM supervectors. The URBM tries to learn the total session and speaker variability among background supervectors. It will then be used to transform unseen supervectors to lower dimensional vectors which will be referred to as GMM-RBM vectors. We investigate the effect of the type of the activation function for training the URBM and the type of the transformation function for GMM-RBM vector extraction. At the end, a variation of Rectified Linear Unit (ReLU), which will be referred to as Variable ReLU (VReLU), is proposed for training the URBM, and then a linear function is used for transformation in the vector extraction stage. The core condition of NIST SRE 2006 (NIST, 2006) is used for the development and the core condition 5 of NIST SRE 2010 (NIST, 2010) with much bigger background data is used for the test and evaluation. The experiments on the evaluation set shows that the proposed GMM-RBM vectors achieve comparable performance with conventional i-vectors while a clearly lower computational cost is required for vector extraction. The conclusion is valid with both cosine and PLDA scoring. Moreover, the combination of GMM-RBM vectors and i-vectors at the score level improves the performance more.

III. To make use of deep architectures for backend i-vector classification for Language Identification (LID) in intelligent vehicles. In this scenario, LID systems are evaluated using words or short sentences recorded in cars in four languages. As the use of DNNs in the i-vector extraction process is computationally expensive for both acoustic modeling and bottleneck feature extraction, we will use the conventional i-vectors in this task in which the computational time is important. Instead, we will explore the use of DNNs only for i-vector language classification. As opposed to (Matějka et al., 2012; Matejka et al., 2014; Ferrer et al., 2016b) in which neural networks with only one hidden layer are used for this purpose, we will explore DNNs with different architectures. Additionally, both raw i-vectors and channel-compensated i-vectors are considered as inputs to DNNs. In order to have the highest accuracy with the minimum response time of the system, signals with different durations from less than 2 sec to higher than 3 sec with the average duration of 3.8 sec are analyzed. The performances of the proposed DNN architectures are compared with both frame-based GMM-Universal Background Model (UBM) and i-vector baseline systems.

1.3 Outline

The thesis is organized as follows.

Chapter 2 reviews briefly both speaker and language recognition tasks as well as the main techniques, from feature extraction to classification, used in the state-of-the-art. It describes also the DL techniques which have been used in this thesis and summarizes some recent developments in the state-of-the-art techniques using DL.

Chapter 3 describes our proposed algorithms and techniques regarding to the first objective mentioned in the previous section. It contains the proposed DNN adaptation technique, impostor selection algorithms, and the general process for creating the discriminative target speaker models based on hybrid DBN-DNNs. Experimental results are given on both NIST SRE 2006 and NIST 2014 i-vector challenge datasets.

Chapter 4 reports on our proposed technique for vector representation of speech waveform using RBMs for speaker recognition. The use of different activation functions for training the URBM and different transformation functions for extracting the proposed vectors are investigated. The effective VReLU is also proposed in this chapter. Evaluation experiments are performed on the core test condition 5 of NIST SRE 2010.

Chapter 5 describes our proposed DNN architecture to model effectively the i-vector space of short utterances spoken in four languages: English, Spanish, German, and Finnish. Both raw i-vectors and session variability compensated i-vectors are evaluated as input vectors to DNNs. The performance of the proposed DNN architecture is compared with both conventional GMM-UBM and i-vector/LDA systems considering the effect of the duration of signals.

Finally, Chapter 6 concludes the thesis and gives some hints for the future work.

Chapter 2

State of the Art

This section gives first a brief overview on the state-of-the-art techniques in both speaker and language recognition from frontend to backend, including speech characterization, acoustic modeling, session compensation and scoring techniques. Afterwards, it will describe the deep learning techniques which have been used in this work. Then some recent developments in the state-of-the-art techniques using Deep Learning (DL) are summarized.

2.1 Speaker and Language Recognition

The history of speaker recognition, recognizing who is speaking, dates back around four decades. It uses the speech acoustic features that have been found to differ between individuals. These acoustic patterns reflect both anatomy (e.g., size and shape of the throat and mouth) and learned behavioral patterns (e.g., voice pitch and speaking style). Two main branches can be considered for speaker recognition, namely speaker identification and speaker verification. Speaker identification can be described as a multi-class classification problem. Given a test utterance, it is to be identified which is the speaker of that utterance, among a set of enrolled speakers. The assumption of whether the test speaker belongs to the enrollment set gives places to two different identification problems, closed-set and open-set, the latter being more difficult as a threshold should be defined as well. In speaker verification, on the other hand, an unknown speaker claims an identity, and the system is to verify if the claim

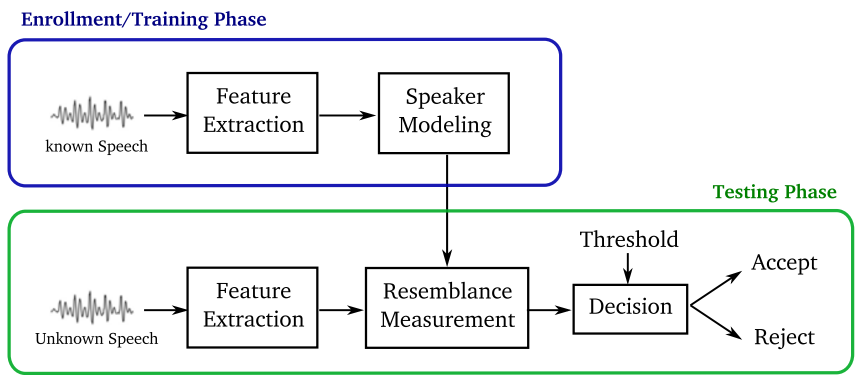


Figure 2.1: Block diagram of a basic speaker verification system.

is true. Speaker recognition can be categorized into two main tasks, namely text-dependent and text-independent. The text-dependent speaker verification requires the speaker saying exactly a given text, password, or sequence of numbers, whereas the text-independent speaker verification is based on free speech. In principle, the text-dependent task is more accurate and needs less amount of training and testing data compared to the text-independent one. However, the text-independent speaker verification is more convenient for users as they can speak freely. Moreover, in some applications only the text-independent task is applicable. As this thesis investigates over the text-independent speaker verification, only this specific task will be followed from now on.

As it is shown in Fig 2.1, speaker verification involves two main phases: the training phase in which the target speakers are enrolled and the testing phase in which a decision about the identity of the speaker is taken. From training point of view, speaker models can be classified into generative and discriminative. Generative models such as Gaussian Mixture Model (GMM) (D. Reynolds & Rose, 1995) estimate the feature distribution within each speaker. Discriminative models such as Support Vector Machine (SVM) (W. Campbell, Campbell, Reynolds, Singer, & Torres-Carrasquillo, 2006) and Deep Neural Network (DNN) , in contrast, model the boundary between speakers.

Language recognition, on the other hand, is mainly applicable for Language Identification (LID). As in speaker recognition, each target language is first modeled. Then an utterance with unknown language is compared with all target languages and the one with highest score is selected. If the task is an open-set identification, a threshold should also be defined for detecting unknown languages. A wide range of approaches has been proposed for modeling the characteristics of languages. The two most effective ones are: the acoustic-phonetic approach and the phonotactic approach (Ambikairajah, Li, Wang, Yin, & Sethu, 2011; Li, Ma, & Lee, 2013). The acoustic-phonetic approach is motivated by the observation that languages differ at a very fundamental level in terms of phones and frequencies of these phones occurring (i.e., the phonetic differences between languages). More importantly, it is assumed that the phonetic characteristics could be captured by some form of spectral-based features. Then the distribution of each language in the feature space is modeled. The phonotactic approach is motivated by the belief that a spoken language can be characterized by its lexical-phonological constraints. Hence, the lexical-phonological rules of each language are taken into account in the phonotactic level to connect phonemes and to form words. From a system development point of view, the acoustic-phonetic approach requires only the speech utterances and the language labels, while the phonotactic approach requires the phonetic transcription of speech, which could be expensive to obtain. Moreover, the acoustic-phonetic approaches are generally much faster in the test phase, which is important in real-time applications. In this thesis, we have only focused on the acoustic-phonetic approach because of its less complexity in both train and test phases and also because it has many techniques in common with the speaker recognition task, which is the first part of the thesis.

2.1.1 Speech Characterization

In almost any kind of pattern recognition system, one of the basic steps is the extraction of features from the raw data. In the context of speaker recognition, features obtained from the speech signal attempt to reflect the discriminative speaker information. A standard in the field is to use short-term acoustic features derived from the speech spectrum. The spectrum of the speech is closely related to the physiology of the human vocal tract, an important discriminating factor. By far, the most popular one is the Mel-Frequency Cepstral Coefficient (MFCC) (Furui, 2004), which has showed to perform well in speech processing tasks. Apart from MFCCs, other features like Linear Predictive Coefficient (LPC), Linear Frequency Cepstral Coefficient (LFCC), Perceptual Linear Predictive (PLP) coefficients, and frequency filtered filter-bank energies or in short Frequency Filtering (FF) coefficients (Nadeu, Hernando, & Gorricho, 1995; Nadeu, Macho, & Hernando, 2001) are used as well. In speaker recognition, the first and the second order time derivatives called delta and delta-delta coefficients are usually obtained to assist the recognition. However, delta-delta coefficients are shown not to be such effective in speaker recognition (Fauve, Matrouf, Scheffer, Bonastre, & Mason, 2007). The delta energy is also commonly added to the feature vectors.

For language recognition, however, only the short-term spectral features are not enough to represent the characteristics of a language. Therefore, a longer duration of speech is usually taken into consideration. Shifted Delta Cepstrum (SDC) coefficients (Torres-carrasquillo, Singer, Kohler, & Deller, 2002), which capture the speech dynamics over a wider range of speech frames than the first and second order MFCC derivatives, are commonly used.

It is worth noting that for both speaker and language recognition applications, only the speech parts of the signals are useful for discrimination. Hence, the non-speech frames like silence, noise, and music should be removed before modeling. Usually, energy-based Voice Activity Detection (VAD) algorithms are used either in the signal level or after feature extraction. In order to have a more robust recognition system against the environment variation, speech features are usually normalized afterwards.

In this section, the most popular feature type, i.e., MFCC, is first explained, and then FF and SDC features which are used, respectively, for speaker and language recognition experiments in this thesis will be described. Then a short description on feature normalization will be given at the end.

Mel Frequency Cepstral Coefficients

MFCCs are short-term representations of a sound spectrum. The sampled speech waveform is assumed to be almost stationary over short time intervals of 20 to 30 msec. The MFCC feature extraction procedure involves a sliding analysis window along the speech signal. For each window placement, the speech is pre-emphasized and the power spectrum is computed. A filter bank of triangular weighting filters is then used to compute the average energy around the center frequency of each triangle. Filters are distributed on a Mel scale, which approximates the behavior of the human auditory system. Finally, MFCCs are defined as the Discrete Cosine Transform (DCT) of the logarithms of the the filter bank energies. More details can

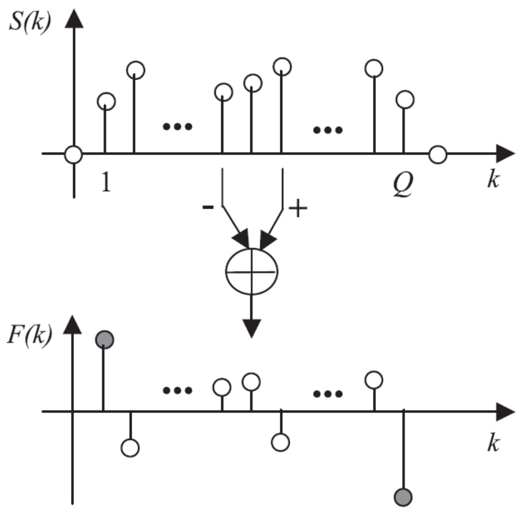


Figure 2.2: FF coefficient computation with the filter $z - z^{-1}$. $S(k)$ is a sequence of filter bank energies and $F(k)$ are the resulting FF coefficients (after (Nadeu et al., 2001)).

be found in literature (Kinnunen & Li, 2010; Hansen & Hasan, 2015).

Frequency Filtering Coefficients

FF coefficients (Nadeu et al., 1995, 2001) are computed in the same fashion as in MFCC but replacing the final DCT of the logarithmic Filter Bank Energys (FBEs) by the following filter,

$$H(z) = z - z^{-1} \tag{2.1}$$

In summary, as it is shown in Fig. 2.2, every FBE coefficient is replaced by the difference between two adjacent FBE coefficients. At the end, the first and the last resulting coefficients are discarded. More details can be found in (Nadeu et al., 2001). These features are not only computationally simpler than MFCCs, they are uncorrelated, they have frequency meaning, and they have shown a performance equal to or better than MFCCs in both speech (Nadeu et al., 2001) and speaker recognition (Hernando & Nadeu, 1997; Luque Serrano, 2012).

Shifted Delta Cepstrum Coefficients

MFCC features are typically computed for each short frame of speech. Usually, the first and the second order derivative vectors are added to take into account the

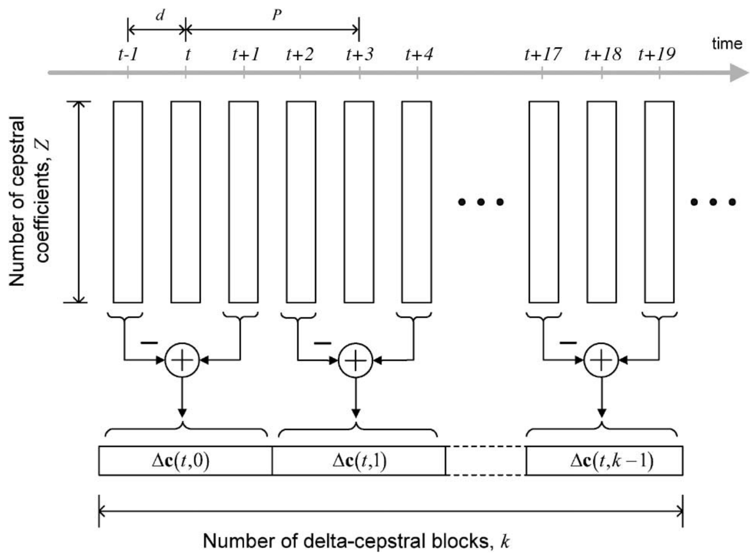

d
P
t-1
t
t+1
t+2
t+3
t+4
t+17
t+18
t+19
time
Number of cepstral coefficients, Z
−
Δc(t,0)
Δc(t,1)
Δc(t,k−1)
Number of delta-cepstral blocks, k

Figure 2.3: SDC computation at frame t for $(z, d, p, k) = (7, 1, 3, 7)$ (after (Li et al., 2013)).

short-term speech dynamics as well. As it was mentioned before, the consideration of speech dynamics over a wider range of speech frames could be useful for language recognition. Hence, SDC coefficients have been proposed (Torres-carrasquillo et al., 2002). SDC features are defined by four parameters $\{z, d, p, k\}$, where z is the number of static cepstral coefficients computed at each frame, d represents the time delay and advance for the delta computation, k is the number of delta-cepstral blocks whose delta-cepstral coefficients are stacked to form the final feature vector, and p is the time shift between consecutive blocks. Figure 2.3 shows an example of SDC computation for parameters $\{7, 1, 3, 7\}$ (Li et al., 2013). More details can be found in (Torres-carrasquillo et al., 2002; Ambikairajah et al., 2011; Li et al., 2013).

Feature Normalization

Feature normalization strategies are employed in both speaker and language recognition systems to compensate for the effects of environmental mismatch. Typically, the mean of the cepstral coefficients is removed in order to avoid nonlinear effects due to the session variability. Optionally, the variance of the cepstral coefficients can be also normalized to unit over a sliding window or over the whole utterance. Sometimes, the shape of the cepstral coefficient distribution is also taken into consideration. Among the several techniques, which have been proposed for this purpose, Cepstral Mean Normalization (CMN), Cepstral Mean and Variance Normalization

(CMVN), and feature warping are more commonly in use (Hansen & Hasan, 2015). In this thesis, we have mainly used feature warping (Pelecanos & Sridharan, 2001).

Feature warping attempts to map the distribution of each individual feature to a Gaussian distribution over a time interval based on the Cumulative Distribution Function (CDF). The method assumes that the components of the feature vector are independent and are processed individually as a separate stream. CDF matching is performed over a sliding window of size N and only the central frame of the window is warped. The features in a given window are sorted in ascending order. If the given component value x in the central frame has the rank r $(1 \leq r \leq N)$, the warped value $\hat{x}$ should satisfy (Pelecanos & Sridharan, 2001; Xiang, Chaudhari, Navrátil, Ramaswamy, & Gopinath, 2002),

$$(r - 1/2)/N = \int_{-\infty}^{\hat{x}} f(z)dz \tag{2.2}$$

where the left side is the approximated CDF value of x, the right side is the CDF value of $\hat{x}$, and $f(z)$ is the Probability Density Function (PDF) of a standard normal distribution (Pelecanos & Sridharan, 2001; Xiang et al., 2002).

2.1.2 GMM-UBM Approach

A GMM is a weighted sum of M Gaussian densities as given by,

$$p(\boldsymbol{x}|\lambda) = \sum_{i=1}^{M} w_i g(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{2.3}$$

where $\boldsymbol{x}$ is a D-dimensional feature vector, i is the index of the ith Gaussian mixture, $g(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ are Gaussian mixtures defined as,

$$g(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{D/2} |\boldsymbol{\Sigma}_i|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_i)' \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_i)\right\}, \tag{2.4}$$

and w_i, $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the weight, the mean vector, and the covariance matrix of the ith Gaussian density, respectively. A GMM is actually defined by these three sets of parameters as $\lambda = \{w_i, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\}_{i=1}^{M}$. The sum of the Gaussian mixture weights equals to 1. The components of the feature vectors are supposed to be decorrelated and, therefore, the covariance matrices $\boldsymbol{\Sigma}_i$ are usually considered diagonal. The GMM parameters are estimated using the Expectation-Maximization (EM) algorithm as in (D. Reynolds & Rose, 1995).

As the amount of the enrollment data for each speaker is usually few, it is not so efficient to train a GMM for each speaker from scratch. Therefore, a global GMM, which is referred to as Universal Background Model (UBM), is first trained using a large number of utterances, and then the UBM is adapted to a few amount of data of each speaker (D. A. Reynolds, Quatieri, & Dunn, 2000). UBM is also known as a speaker and language independent model since it is usually trained with hundreds of hours of speech data spoken by thousands of speakers in different languages. However, UBM can be sometimes gender dependent for more efficiency. The adaptation is typically performed using the Maximum a Posteriori (MAP) estimation which includes two steps. First, the sufficient statistics, which are known as Baum-Welch statistics, are calculated given the new feature vectors $\boldsymbol{u} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T\}$ and the UBM as follows,

$$\mathcal{N}_i(\boldsymbol{u}) = \sum_{t=1}^{T} Pr(i|x_t, \lambda_{ubm}) \tag{2.5}$$

$$\mathcal{F}_i(\boldsymbol{u}) = \sum_{t=1}^{T} Pr(i|x_t, \lambda_{ubm}) x_t \tag{2.6}$$

where $\mathcal{N}_i(\boldsymbol{u})$ and $\mathcal{F}_i(\boldsymbol{u})$ are the zeroth and the first order statistics, respectively, and $Pr(i|x_t, \lambda_{ubm})$ is the a posteriori probability for the Gaussian mixture i calculated as follows,

$$Pr(i|x_t, \lambda_{ubm}) = \frac{w_i g(\boldsymbol{x}_t|\boldsymbol{\mu}_i^{ubm}, \boldsymbol{\Sigma}_i^{ubm})}{\sum_{k=1}^{M} w_i g(\boldsymbol{x}_t|\boldsymbol{\mu}_k^{ubm}, \boldsymbol{\Sigma}_k^{ubm})} \tag{2.7}$$

Second, the adapted parameters are obtained by the combination of the new statistics for speaker a and the UBM parameters. Usually, only the mean vectors are adapted as follows,

$$\boldsymbol{\mu}_i^a = \alpha_i \left[\frac{\mathcal{F}_i(\boldsymbol{u})}{\mathcal{N}_i(\boldsymbol{u})} \right] + (1 - \alpha_i)\boldsymbol{\mu}_i^{ubm} \tag{2.8}$$

where $\boldsymbol{\mu}_i^a$ are the adapted mean vectors for speaker a and α_i are defined as,

$$\alpha_i = \frac{\mathcal{N}_i(\boldsymbol{u})}{\mathcal{N}_i(\boldsymbol{u}) + r} \tag{2.9}$$

where r is a fixed relevance factor. Equation 2.9 implies that when $\mathcal{N}_i \to \infty$, $\alpha_i \to 1$ and when $\mathcal{N}_i \to 0$, $\alpha_i \to 0$, i.e., the adapted model relies more on the UBM than the new statistics for those Gaussian mixtures where few data is available and vice versa. At the end, the scores are defined in a log likelihood ratio form as follows,

$$\Lambda(\boldsymbol{u}) = \log p(\boldsymbol{u}|\lambda_a) - \log p(\boldsymbol{u}|\lambda_{ubm}) \tag{2.10}$$

where $\log p(\boldsymbol{u}|\lambda) = \frac{1}{T}\sum_{t=1}^{T} \log p(\boldsymbol{x}_t|\lambda)$ and $p(\boldsymbol{x}_t|\lambda)$ is obtained as in Eq. 2.3.

2.1.3 Supervector and i-Vector

The computation of the decision score as in Eq. 2.10 is costly since it should be computed per each frame and every time twice, once against the adapted model and another time against the UBM. Hence, it would be more convenient and computationally efficient if each sequence of feature vectors is converted to a fixed length vector and the decision score is computed only once. GMM supervectors are one of these kind of fixed length vectors which are obtained by stacking the D-dimensional mean vectors of the M-mixture adapted GMM (W. Campbell, Sturim, & Reynolds, 2006). For the speaker a, a GMM supervector is represented as,

$$\boldsymbol{s}^a = (\boldsymbol{\mu}_1^a, \boldsymbol{\mu}_2^a, ..., \boldsymbol{\mu}_M^a)^t \tag{2.11}$$

where t refers to a transpose operation.

It is possible to compare two supervectors based on their distance, but it is commonly more efficient to classify them by SVMs (W. Campbell, Sturim, & Reynolds, 2006; Dehak & Chollet, 2006; K. Lee, You, Li, Kinnunen, & Zhu, 2008). This leads to a hybrid classifier in which the generative GMM-UBM is used to create supervectors as feature vectors for the discriminative SVM.

As the train and test speech utterances are usually spoken in different sessions, e.g., the use of different microphones or different channels for transferring the speech signal, some session variability compensation techniques are required, in addition to the compensation techniques in the feature domain as described in section 2.1.1, for having a higher recognition accuracy and a more robust system. Two commonly in use session compensation techniques in the supervector domain are the Nuisance Attribute Projection (NAP) (W. Campbell, Sturim, Reynolds, & Solomonoff, 2006; Solomonoff, Campbell, & Boardman, 2005) and Within-Class Covariance Normalization (WCCN)(Hatch, Kajarekar, & Stolcke, 2006).

Another successful technique is the Joint Factor Analysis (JFA) (Kenny, 2006) which models the supervectors as a linear combination of the speaker and channel components. However, as in SVM based techniques, the session variability compensation is carried out in the supervector domain which is very high dimensional. Therefore, a big memory space is required for the training of the compensation matrices and it is computationally expensive. It is proposed in (Dehak, Kenny, et al., 2011) to first reduce the dimension of the supervectors throw an effective factor analysis technique and then to perform session compensation in the lower dimensional space. It is supposed that the supervector $\boldsymbol{s}^a$ can be modeled as follows (Dehak, Kenny, et al., 2011),

$$\boldsymbol{s}^a = \boldsymbol{s}^{ubm} + \boldsymbol{T\nu} \tag{2.12}$$

where $\boldsymbol{s}^{ubm}$ is the speaker- and session-independent mean supervector, typically from the UBM, $\boldsymbol{T}$ is the total variability matrix, and $\boldsymbol{\nu}$ is a vector of latent variables. The posterior distribution of $\boldsymbol{\nu}$ is conditioned on the Baum-Welch statistics of the given speech utterance. The mean of this posterior distribution is referred to as i-vector $\boldsymbol{\omega}$ and computed as follows,

$$\boldsymbol{\omega} = \left(\boldsymbol{I} + \boldsymbol{T}^t \boldsymbol{\Sigma}^{-1} \mathcal{N}(\boldsymbol{u}) \boldsymbol{T}\right)^{-1} \boldsymbol{T}^t \boldsymbol{\Sigma}^{-1} \tilde{\mathcal{F}}(\boldsymbol{u}) \tag{2.13}$$

where $\mathcal{N}(\boldsymbol{u})$ is a diagonal matrix containing the zeroth order Baum-Welch statistics, $\tilde{\mathcal{F}}(\boldsymbol{u})$ is a supervector of the centralized first order statistics, and $\boldsymbol{\Sigma}$ is a diagonal covariance matrix initialized by $\boldsymbol{\Sigma}_{ubm}$ and updated during the factor analysis training. The $\boldsymbol{T}$ matrix is trained using the EM algorithm given the Baum-Welch statistics from the background speech utterances. More details can be found in (Dehak, Kenny, et al., 2011).

2.1.4 i-Vector Backends

The preliminary scoring technique for i-vectors is cosine (Dehak, Dehak, glass, Reynolds, & Kenny, 2010; Dehak, Kenny, et al., 2011),

$$score^{(cosine)}(\boldsymbol{\omega}_1, \boldsymbol{\omega}_2) = \frac{\boldsymbol{\omega}_1^t \boldsymbol{\omega}_2}{\|\boldsymbol{\omega}_1\| \times \|\boldsymbol{\omega}_2\|} \tag{2.14}$$

where $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ are the target and the test i-vectors and $\|\boldsymbol{\omega}\|$ denotes the norm of the i-vector $\boldsymbol{\omega}$ computed as $\sqrt{\omega_1^2, \omega_2^2, ..., \omega_n^2}$. If the speaker labels for the background speech utterances are not available, the cosine scoring gives as such a reasonable accuracy. However, given the speaker labels it is more effective if a session variability compensation technique is applied before scoring. Linear Discriminant Analysis (LDA), WCCN, or a combination of them is usually used (Dehak, Kenny, et al., 2011). It should be noted that speaker labels are costly and are not always accessible. Probabilistic Linear Discriminant Analysis (PLDA) (Prince & Elder, 2007) is a more effective technique when speaker labels are available for the background data. These three commonly in use techniques are described briefly as follows.

Linear Discriminant Analysis

In LDA, the feature vectore, i-vectors in this case, are transformed usually to a lower dimensinal feature space in which the classification task become easier. LDA finds the orthogonal directions in the current feature space which are more effective in discriminating the classes (Hansen & Hasan, 2015). Given the background i-vectors and the corresponding class labels, e.g., speaker or language labels, LDA tries to maximize the between-class covariance matrix while minimizing the within-class covariance matrix defined as follows,

$$\boldsymbol{S}_b = \frac{1}{S}\sum_{s=1}^{S}\left(\bar{\boldsymbol{\omega}}_s - \bar{\boldsymbol{\omega}}\right)\left(\bar{\boldsymbol{\omega}}_s - \bar{\boldsymbol{\omega}}\right)^t \tag{2.15}$$

$$\boldsymbol{S}_w = \frac{1}{S}\sum_{s=1}^{S}\frac{1}{n_s}\sum_{i=1}^{n_s}\left(\boldsymbol{\omega}_{s,i} - \bar{\boldsymbol{\omega}}_s\right)\left(\boldsymbol{\omega}_{s,i} - \bar{\boldsymbol{\omega}}_s\right)^t \tag{2.16}$$

where $\boldsymbol{\omega}_{s,i}$ is the ith i-vector of speaker s, n_s is the total number of i-vectors belonging to speaker s, S is the total number of speakers in the background set of i-vectors, and $\bar{\boldsymbol{\omega}}_s$ and $\bar{\boldsymbol{\omega}}$ are the speaker-dependent and speaker-independent mean i-vectors, obtained on each class and on the whole background data, respectively,

Within-Class Covariance Normalization

WCCN is a normalization technique which was mainly used for improving the robustness of SVM based speaker recognition (Hatch et al., 2006; Hatch & Stolcke, 2006). The within-class covariance matrix $\boldsymbol{S}_w$ is first calculated as in eq. 2.16. Then the projection WCCN matrix is computed using the Cholesky factorization of $\boldsymbol{S}_w^{-1}$ such that,

$$\boldsymbol{S}_w^{-1} = \boldsymbol{A}_{WCCN}\boldsymbol{A}_{WCCN}^t \tag{2.17}$$

$$\hat{\boldsymbol{\omega}} = \boldsymbol{A}_{WCCN}^t\boldsymbol{\omega} \tag{2.18}$$

where $\hat{\boldsymbol{\omega}}$ is the projected form of i-vector $\boldsymbol{\omega}$.

It is worth noting that unlike LDA, the WCCN projection conserves the direction and the dimension of the feature space (Hansen & Hasan, 2015).

Probabilistic Linear Discriminant Analysis

In PLDA, scoring is performed along with the session variability compensation. It assumes that each i-vector can be decomposed as,

$$\boldsymbol{\omega} = \boldsymbol{m} + \boldsymbol{\Phi}\boldsymbol{\zeta} + \boldsymbol{\varepsilon} \tag{2.19}$$

where $\boldsymbol{m}$ is a global offset, the columns of $\boldsymbol{\Phi}$ are eigenvoices, $\boldsymbol{\zeta}$ is a latent vector having a standard normal prior, and the residual vector $\boldsymbol{\varepsilon}$ is normally distributed with zero mean and the full covariance matrix $\boldsymbol{\Sigma}$. The model parameters are estimated from a large collection of speaker-labeled background data using the EM algorithm as in (Prince & Elder, 2007).

Given the two i-vectors $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ involved in a trial for a speaker verification task, there could be two hypotheses $\mathcal{H}_s$ and $\mathcal{H}_d$ indicating that two i-vectors $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ belong to the same or different speakers, respectively. The verification score can now be computed as the log-likelihood ratio of these two hypotheses as,

$$score^{(plda)}(\boldsymbol{\omega}_1, \boldsymbol{\omega}_2) = \log \frac{p(\boldsymbol{\omega}_1, \boldsymbol{\omega}_2 | \mathcal{H}_s)}{p(\boldsymbol{\omega}_1 | \mathcal{H}_d) p(\boldsymbol{\omega}_2 | \mathcal{H}_d)} \tag{2.20}$$

Supposing that the i-vectors are generated from a Gaussian distribution and the global offset, i.e., $\boldsymbol{m}$, is removed, the log-likelihood ratio is computed in a closed-form solution resulting in,

$$score^{(plda)}(\boldsymbol{\omega}_1, \boldsymbol{\omega}_2) = \boldsymbol{\omega}_1^t Q \boldsymbol{\omega}_1 + \boldsymbol{\omega}_2^t Q \boldsymbol{\omega}_2 + 2\boldsymbol{\omega}_1^t P \boldsymbol{\omega}_2 \tag{2.21}$$

where P and Q are square matrices represented in terms of within and between covarience matrices as,

$$Q = \boldsymbol{\Sigma}_b^{-1} - \left(\boldsymbol{\Sigma}_b - \boldsymbol{\Sigma}_w \boldsymbol{\Sigma}_b^{-1} \boldsymbol{\Sigma}_w\right)^{-1}, \tag{2.22}$$

$$P = \boldsymbol{\Sigma}_b^{-1} \boldsymbol{\Sigma}_w \left(\boldsymbol{\Sigma}_b - \boldsymbol{\Sigma}_w \boldsymbol{\Sigma}_b^{-1} \boldsymbol{\Sigma}_w\right)^{-1} \tag{2.23}$$

where $\boldsymbol{\Sigma}_w = \boldsymbol{\Phi}\boldsymbol{\Phi}^t$ and $\boldsymbol{\Sigma}_b = \boldsymbol{\Phi}\boldsymbol{\Phi}^t + \boldsymbol{\Sigma}$. To speed up the scoring process, the above formulas are usually summarized to be calculated in the PLDA lower dimensional space. More details can be found in (Garcia-Romero & Espy-Wilson, 2011). As proposed in (Kudashev, Novoselov, Pekhovsky, Simonchik, & Lavrentyeva, 2016), for simplicity, the LDA within and between covariance matrices can also be used in eqs. 2.22 and 2.23.

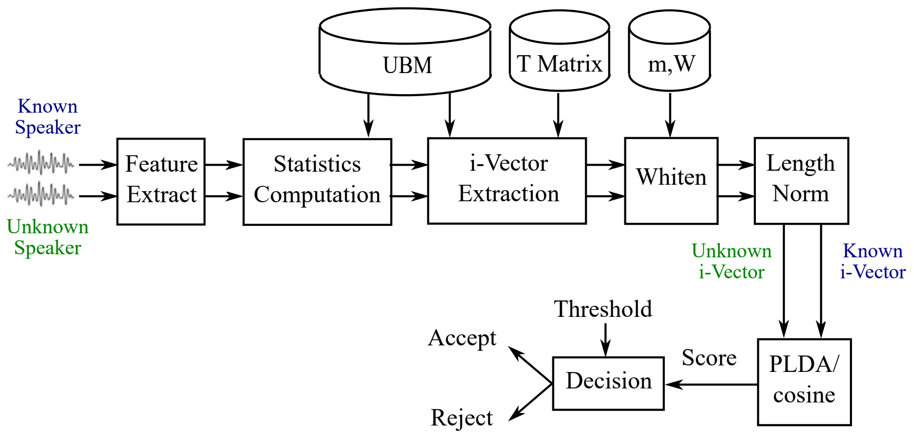


Figure 2.4: Block-diagram of a typical i-vector/PLDA speaker verification system.

It is shown (Garcia-Romero & Espy-Wilson, 2011) that the length normalization ($\boldsymbol{\omega} \leftarrow \frac{\boldsymbol{\omega}}{\|\boldsymbol{\omega}\|}$) helps the Gaussianity of i-vectors which leads to a comparable performance to a more complicated PLDA, which is referred to as heavy-tailed PLDA (Kenny, 2010; Matějka et al., 2011). As proposed in (Greenberg et al., 2014), in an i-vector baseline system, i-vectors are first globally whitened as,

$$\boldsymbol{\omega}' = \boldsymbol{H}\boldsymbol{\omega} \tag{2.24}$$

$$\boldsymbol{H} = \boldsymbol{V}\left(\boldsymbol{D} + \epsilon\right)^{-1/2}\boldsymbol{V}^t \tag{2.25}$$

where $\boldsymbol{H}$ is the whitening matrix, $\boldsymbol{V}$ is the matrix of eigenvectors obtained on the covariance matrix of the background i-vectors, $\boldsymbol{D}$ is the diagonal matrix of the corresponding eigenvalues, and ϵ is a very small constant regularization factor. After whitening, i-vectors are length-normalized. If the target speaker enrollment has been a single-session task, i.e., one utterance available per each target speaker, test and target i-vectors are compared directly using either cosine or PLDA scoring. Otherwise, the available i-vectors per each target speaker are first averaged and a single i-vector is obtained. In case of cosine scoring, averaged target i-vectors are length-normalized again before scoring but in case of PLDA, the experiments have shown that length-normalization of the averaged target i-vectors will worsen the results (Greenberg et al., 2014). Moreover, it is observed (Greenberg et al., 2014) that for PLDA training, excluding those background i-vectors obtained on utterances with speech duration shorter than 30 msec will improve the results. The block-diagram of Fig. 2.4 summarizes a typical state-of-the-art system for speaker verification.

2.2 Deep Learning

DL refers to a branch of machine learning techniques which attempts to learn high level features from data. Since 2006 (Hinton & Salakhutdinov, 2006; Hinton, Osindero, & Teh, 2006), DL has become a new area of research in many applications of machine learning and signal processing. From training point of view, DL techniques can be divided in three main groups, supervised, unsupervised, and a hybrid of supervised and unsupervised (Deng & Yu, 2014). Supervised techniques need labeled data for training, e.g., phonetic, speaker, or language labels in speech processing, and they are usually intended to directly provide discriminative power for pattern classification purposes, often by characterizing the posterior distributions of classes conditioned on the input data. Labeled data is not easily accessible and, in most cases, is usually expensive which can be considered as a disadvantage of these techniques. DNN, Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM) are some well-known techniques in this group (Deng & Yu, 2014). In contrary, unsupervised techniques take advantage of a large amount of unlabeled data, which are usually easily accessible, to capture high-order correlations in observed data. Restricted Boltzmann Machine (RBM), autoencoder, Deep Belief Network (DBN), and Deep Boltzmann Machine (DBM) are some examples in this group (Deng & Yu, 2014). Hybrid techniques are typically more effective since they take advantage of both labeled and unlabeled data, which are usually a few and a large amount in practice, respectively. A DBN-DNN is a well-known architecture of this type (Deng & Yu, 2014).

Various DL architectures have been used in speech processing (e.g., Hinton et al., 2012; Z.-H. Ling, Deng, & Yu, 2013; X.-L. Zhang & Wu, 2013; Senior et al., 2015; Sainath et al., 2015; Nugraha, Liutkus, & Vincent, 2016; Sainath et al., 2017). DNN, DBN, and RBM are three main architectures we have used in this thesis.

2.2.1 Deep Neural Networks

DNNs are feed-forward neural networks with multiple hidden layers between input and output layers (Fig. 2.5a). The hidden unit values in each hidden layer are computed as follows,

$$h_j^l = f(b_j^l + \sum_i h_i^{l-1} w_{ij}^l) \tag{2.26}$$

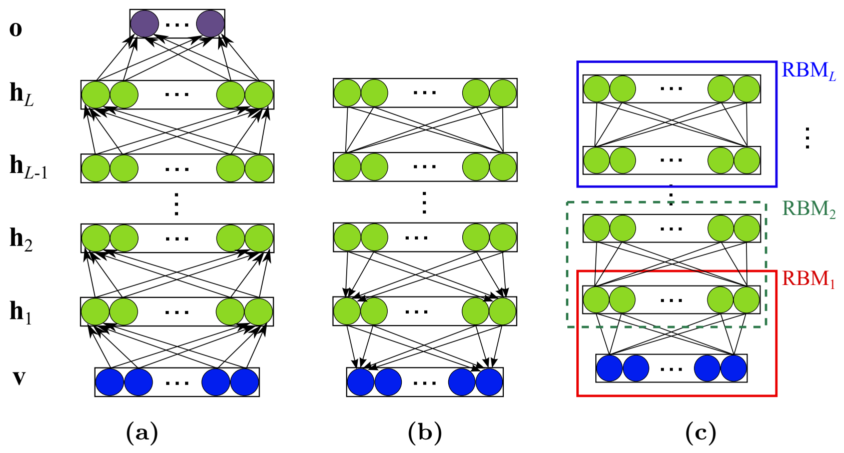


Figure 2.5: (a) DNN, (b) DBN, and (c) DBN training/DNN pre-training.

where h_j^l and b_j^l are, respectively, the posterior probability and the bias term of the jth hidden unit in the lth hidden layer, w_{ij}^l is the connection weight between jth hidden unit in layer l and ith hidden unit in layer $l-1$, and $f(.)$ is an activation function. Note that layers $l=0$ and $l=L+1$ are visible (i.e., input) and output layers, where L is the number of hidden layers in the network.

Two commonly in use hidden units are sigmoid and Rectified Linear Unit (ReLU). The names come from the type of the activation function used,

$$sigmoid: \quad f(x) = (1+\exp(-x))^{-1} \tag{2.27}$$

$$ReLU: \quad f(x) = \begin{cases} x & x > 0 \\ 0 & x \le 0. \end{cases} \tag{2.28}$$

DNNs are trained using a discriminative back-propagation algorithm given the class labels of the input vectors. The training algorithm tries to minimize a loss function between the class labels $\boldsymbol{\ell} = (\ell_1, \ell_2, ..., \ell_K)$ and the outputs $\boldsymbol{o} = (o_1, o_2, ..., o_K)$. For a classification purpose, the cross-entropy is often used as the loss function,

$$\mathcal{H}(\boldsymbol{\ell}, \boldsymbol{o}|\boldsymbol{v}, \theta) = -\sum_{k=1}^{K} \ell_k \log(o_k) \tag{2.29}$$

where $\boldsymbol{v}$ is the input vector and θ is the DNN parameters. The softmax is commonly used as the activation function at the output layer for classification (Z. Ling et al., 2015),

$$p(o_k|\boldsymbol{v}) = \frac{\exp(b_k^{L+1} + \sum_i h_i^L w_{ik}^{L+1})}{\sum_{j=1}^{K} \exp(b_j^{L+1} + \sum_i h_i^L w_{ij}^{L+1})}. \tag{2.30}$$

where $p(o_k|\boldsymbol{v})$ denotes the probability that class k is the correct class. The number of the output units will be equivalent to the number of the classes. The label of each class is defined as a binary vector, with the same length as the output layer, in which all the components are zero except the component corresponding to that class. As the Eq. 2.30 implies, the softmax function guarantees that each output unit will have a probability between 0 and 1 and the sum of the output probabilities is 1.

In traditional neural networks, the parameters are initialized with small random numbers and the sigmoid or hyperbolic tangent ($tanh$) functions are usually used for activation. However, the classification accuracy of the network decreases by increasing the number of hidden layers. The problem is that both sigmoid and $tanh$ functions suffer from the gradient vanishing, i.e., the gradient decreases when the absolute value of the input of these functions increases. In this way, the parameters of the lower layers will only slightly change. This leads to a performance degradation.

Two main solutions have been proposed for the gradient vanishing problem, either the use of better initialization techniques (Larochelle, Bengio, Louradour, & Lamblin, 2009; Dumitru, Manzagol, Bengio, Bengio, & Vincent, 2009; Erhan et al., 2010; Yu, Deng, Seide, & Li, 2016) or employing the activation functions which do not suffer from gradient vanishing, e.g., Rectified Linear (ReL) (Nair & Hinton, 2010; Zeiler et al., 2013; Maas, Hannun, & Ng, 2013). The first solution is chosen in this thesis. One of the efficient techniques for initialization is to initialize DNNs with DBN parameters, which is referred to as unsupervised pre-training and the final network is often called a hybrid DBN-DNN (Dahl et al., 2012; Deng & Yu, 2014). It has been empirically shown that this pre-training stage can set the weights of the network closer to an optimum solution than the random initialization (Larochelle et al., 2009; Dumitru et al., 2009; Erhan et al., 2010).

2.2.2 Deep Belief Networks

DBNs are generative models with multiple hidden layers of stochastic units above a visible layer, which represents a data vector (Fig. 2.5b). The top two layers are undirected and the other layers have top-down directed connections to generate the data. The main advantage of DBN is that it does not need labels for training and it is trained totally unsupervised. Hence, as it was mentioned in section 2.2.1, the combination of DBNs and DNNs could make a powerful model which benefits from both the large number of unlabeled data, which are usually accessible easily, and the fewer number of labeled data, which are usually costly to provide. In addition to the pre-training of DNNs, DBNs have been used for many other applications as well, e.g., non-linear dimensionality reduction (Hinton & Salakhutdinov, 2006; Salakhutdinov & Hinton, 2009), generating images (Susskind, Hinton, Movellan, & Anderson, 2008), and feature learning from images (F. J. Huang, Boureau, LeCun, et al., 2007), video sequences (Sutskever & Hinton, 2007) and audio signals (H. Lee, Largman, Pham, & Ng, 2009).

There is an efficient greedy layer wised algorithm to train DBN parameters (Hinton et al., 2006). In this case, DBN is divided in two-layer sub-networks and each one is treated as an RBM (Fig. 2.5c). When the first RBM, which is built on visible units, is trained, its parameters are frozen and the outputs are given to the RBM above as input vectors. This process is repeated until the top two layers are reached.

It is worth noting that the term DBN has previously been used in the literature for both unsupervised DBN, as it was explained in this section, and supervised hybrid DBN-DNN, as it was explained in section 2.2.1 (e.g., Mohamed, Dahl, & Hinton, 2009; Nair & Hinton, 2009; Mohamed et al., 2012; X.-L. Zhang & Wu, 2013). Nowadays, DBN just refers to the unsupervised network of Fig. 2.5b based on the definition given in (Deng & Yu, 2014).

RBMs are generative models constructed from two undirected layers of stochastic hidden and visible units (Fig. 2.6a). RBM training is based on a maximum likelihood criterion using the stochastic gradient descent algorithm (Hinton et al., 2006; Dahl et al., 2012). The gradient is estimated by an approximated version of the Contrastive Divergence (CD) algorithm which is called CD_1 (Hinton et al., 2006; Hinton, 2012).

Fig. 2.6b shows how an RBM is trained using CD_1. The connection weights $\boldsymbol{W}$ are randomly initialized and the visible and hidden bias terms ($\boldsymbol{a}$ and $\boldsymbol{b}$, respectively) are set to zero. Given the input vectors $\boldsymbol{v}$, the posterior probability of the hidden vector $\boldsymbol{h}$ is calculated and binarized based on random thresholds. Afterwards, input vectors are reconstructed given the binary values of the hidden layer. Then the

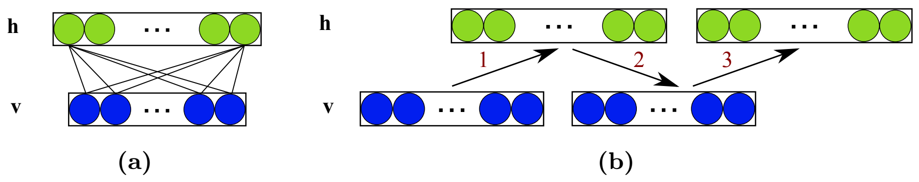


Figure 2.6: (a) RBM and (b) RBM training.

reconstructed inputs $\boldsymbol{v}_r$ are used to recalculate the posterior probabilities of hidden units. These three steps, marked in Fig. 2.6b, provide enough statistics to update the parameters of the network. From an algorithmic point of view, training an RBM with CD$_1$ where the input data is real-valued Gaussian distributed can be summarized as follows,

- Initialize Network Parameters $(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{a})$
- CD$_1$ Steps
  1. $\boldsymbol{h} = \sigma\left(\boldsymbol{b} + \boldsymbol{W}\boldsymbol{v}\right)$ (2.31)
  2. $\boldsymbol{v}_r = \boldsymbol{a} + \boldsymbol{W}^t\boldsymbol{h}'$ (2.32)
  3. $\boldsymbol{h}_r = \sigma\left(\boldsymbol{b} + \boldsymbol{W}\boldsymbol{v}_r\right)$ (2.33)
- Update Network Parameters
  1. $\Delta\boldsymbol{W} = \eta \times \left(\boldsymbol{v}\boldsymbol{h}^t - \boldsymbol{v}_r\boldsymbol{h}_r^t\right)^t$ (2.34)
  2. $\Delta\boldsymbol{a} = \eta \times \left(\boldsymbol{v} - \boldsymbol{v}_r\right)$ (2.35)
  3. $\Delta\boldsymbol{b} = \eta \times \left(\boldsymbol{h} - \boldsymbol{h}_r\right)$ (2.36)

where $\boldsymbol{h}_r$ is the reconstructed version of $\boldsymbol{h}$, $\boldsymbol{h}'$ is a binary vector randomly sampled from $\boldsymbol{h}$, η is the learning rate, and $\sigma(.)$ is the *sigmoid* function as in eq. 2.27.

Additionally, a momentum factor is used to smooth out the updates, and the weight decay regularization is used to penalize large weights. More theoretical and practical details can be found in (Hinton & Salakhutdinov, 2006; Hinton et al., 2006; Hinton, 2012).

In the training phase of all these three networks (DNN, DBN, and RBM), it is possible to update the parameters after processing each training example, but it is often more efficient to divide the whole input data (batch) into smaller size batches (minibatch) and to update the parameters by averaging the gradients over each minibatch. The parameter updating procedure is repeated when the whole available input data is processed. Each iteration is called an epoch.

2.2.3 Deep Learning in Speaker Recognition

Even though steps have been taken long ago to apply neural networks in speaker recognition (e.g., Oglesby & Mason, 1988, 1990; Rudasi & Zahorian, 1991; Farrell, Mammone, & Assaleh, 1994), recent advances in computing hardware, new DL architectures and training methods, and access to large amount of training data has inspired the research community to make use of DL technology again as in a large variety of other signal processing applications (e.g., Mohamed et al., 2012; Z.-H. Ling et al., 2013; X.-L. Zhang & Wu, 2013; Nair & Hinton, 2009; G. Huang, Lee, & Learned-Miller, 2012). DL techniques can be used in the frontend or/and backend of a speaker recognition system. The whole end-to-end recognition process can even be performed by a DL architecture.

Deep Learning Frontends

A possible use of DL in the frontend of a speaker recognition system is in the state-of-the-art i-vector algorithm (Dehak, Kenny, et al., 2011). The traditional i-vector approach consists in three main stages: Baum-Welch statistics collection, i-vector extraction, and PLDA backend. Recently, it is shown that if the Baum-Welch statistics are computed with respect to a DNN rather than a GMM or if bottleneck features are used in addition to conventional spectral features, a substantial improvement can be achieved (Lei et al., 2014; Kenny et al., 2014; Richardson et al., 2015a).

A variant of architectures have been used for acoustic modeling, bottleneck feature extraction, or both of them at the same time. Figure 2.7 shows a typical DNN architecture used for both Baum-Welch statistics computation and bottleneck feature extraction. The network is preliminary trained for acoustic modeling in Automatic Speech Recognition (ASR). The hidden layer before the last hidden layer is usually much smaller than the other hidden layers and considered as the bottleneck layer. The hidden unit values of this layer, given the input vectors, are referred to as bottleneck features. These features are usually highly correlated and, therefore, they

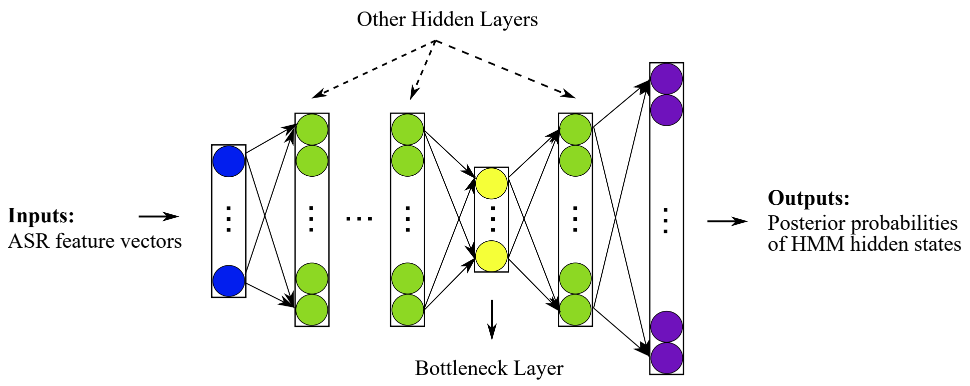


Figure 2.7: A typical DNN architecture used for acoustic modeling and bottleneck feature extraction in DNN based i-vector approach.

need some decorrelating, typically using Principal Component Analysis (PCA), before usage. The output layer represents the acoustic classes, which are typically the states of the Hidden Markov Model (HMM) in ASR. The input layer takes usually a concatenation of successive ASR feature vectors. ASR feature vectors are usually the log filter bank energies without any delta or delta delta coefficients. The activation function for the output layer is softmax, for the bottleneck layer is usually linear, and for the other hidden layers can be sigmoid, rectified linear, or other similar functions like tanh.

Given the DNN acoustic model, the zeroth and the first order Baum-Welch statistics are computed as follows,

$$\mathcal{N}_k(\boldsymbol{u}) = \sum_t p(o_k|x_t) \tag{2.37}$$

$$\mathcal{F}_k(\boldsymbol{u}) = \sum_t p(o_k|x_t)\hat{x}_t \tag{2.38}$$

where $p(o_k|x_t)$ is the posterior probability of kth output unit given the ASR feature vector x_t, and $\hat{x}_t$ is the speaker feature vector which can differ from x_t. Given the Baum-Welch statistics, the $\boldsymbol{T}$ matrix training and the i-vector extraction process will be the same as with GMM acoustic model. However, despite the GMM model, the non-speech frames have also been used in the training of the DNN acoustic model. Therefore, there will be two possible options for DNN statistics computation. The first option is to use an external VAD, discard non-speech frames, and use only the DNN output units corresponding to the speech states in HMM. The second option is not to use any external VAD, compute the statistics for all frames, and, like in the first option, to discard the output probabilities which correspond to non-speech states. The second option can be interpreted as a kind of a soft VAD rather than the hard VAD in the first option (Kudashev et al., 2016). It has been shown that the second option leads usually to a better performance in terms of the accuracy (Ferrer, Lei, McLaren, & Scheffer, 2014, 2016a). It is worth noting that for both options, the zeroth order statistics should be normalized by the sum over the output units corresponding to speech states.

Although the i-vector extraction using DNN acoustic models or bottleneck features leads to a higher accuracy in general, there are also some disadvantages. First of all, the use of DNN itself increases the computational cost of the statistics to a great extent. Moreover, the number of the output units is usually much higher than the number of Gaussian mixtures in the GMM. This means that the dimensions of supervectors will be much higher than the dimensions of GMM supervectors leading to higher computational cost in both $\boldsymbol{T}$ matrix training and i-vector extraction. Additionally, the language dependency of DNN acoustic models, the use of two different feature vectors for the computation of the zeroth and the first order statistics (e.q., 2.37 and 2.38), and the need of the phonetic labels for training the DNN acoustic model are other shortcomings of DNN based i-vector extraction approaches.

Another possible use of DL in the frontend is to represent the speaker characteristics of a speech signal with a single low dimensional vector using a DL architecture, rather than the traditional i-vector algorithm. These vectors are often referred to as speaker embeddings. Typically, the inputs of the neural network are a sequence of feature vectors and the outputs are speaker classes (Fig. 2.8). First, a deep architecture is trained using background feature vectors. Then the feature vectors of a given utterance are forward-propagated and the mean of the posterior probabilities of a particular hidden layer (Variani et al., 2014) or a PCA dimension reduced version of them (Liu et al., 2015), or a PCA dimension reduced version of the mean vectors (Vasilakakis et al., 2013) are considered as a new compact representation. Different architectures, activation functions, and training procedures have been proposed (e.g., Variani et al., 2014; Wang et al., 2017; Bhattacharya et al., 2017; Snyder et al., 2017). The experimental results have shown that, in most cases, the bigger improvements are obtained on the shorter signals compared to the traditional i-vectors (Bhattacharya et al., 2017; Snyder et al., 2017), which implies that DL technology can model better the speaker characteristics of a short-duration speech

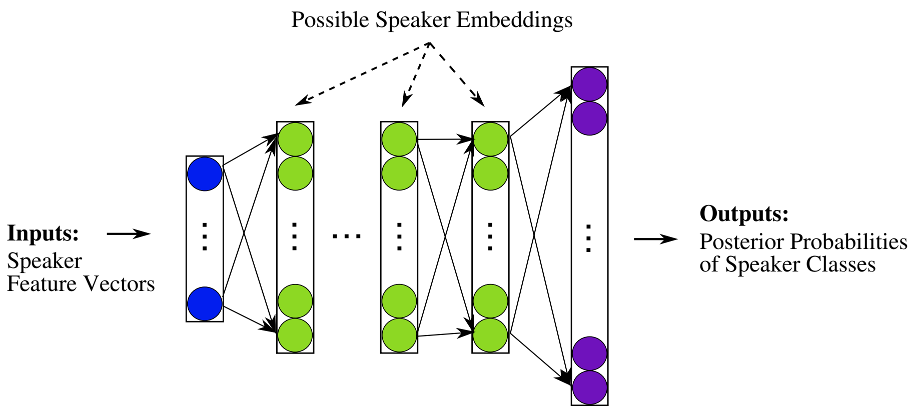


Figure 2.8: A typical DNN architecture used for speaker embedding extraction.

signal than the traditional signal processing techniques. This is important for real-time applications where the decision should be made in very few seconds, provided that the computational cost is still reasonable. However, the need of speaker labels for training the network is one of the disadvantages of these techniques. Moreover, speaker embeddings extracted from hidden layer outputs are not so compatible with PLDA backend because the posterior distribution of hidden layer outputs are usually not truly Gaussian.

Deep Learning Backends

One of the most effective backend techniques for i-vectors is PLDA (Prince & Elder, 2007; Kenny, 2010), which performs the scoring along with the session variability compensation. Usually, a large number of different speakers with several speech samples each are necessary for PLDA to work efficiently. Access to the speaker labeled data is costly and in some cases almost impossible. Moreover, the amount of the performance gain, in terms of accuracy, for short utterances is not as much as that for long utterances. These facts motivated the research community to look for DL based alternative backends. Several techniques have been proposed. Most of these approaches use the speaker labels of the background data for training, as in PLDA, and mostly with no significant gain compared to PLDA. For example, different combinations of RBMs have been proposed in (Stafylakis et al., 2012b; Senoussaoui et al., 2012) to classify i-vectors and in (Stafylakis et al., 2012a) to learn speaker and channel factor subspaces in a PLDA simulation. RBMs in (Novoselov, Pekhovsky, Simonchik, & Shulipa, 2014) and DNNs in (Isik et al., 2015) are used to increase the discrimination power of i-vectors given speaker-labeled background data. A nonlinear PLDA is simulated in (Villalba et al., 2017) using a tied variational autoencoder architecture. In (Novoselov et al., 2015; Pekhovsky et al., 2016), a combination of RBM, autoencoder, and PLDA is proposed for speaker and channel variability compensation, which shows some improvements compared to using only PLDA.

One of the recent challenges in speaker recognition, which was organized by the National Institute of Standard and Technology (NIST), has been how to achieve a comparable performance with PLDA when no labeled background data is available (NIST, 2014). Although some unsupervised automatic labeling techniques have been proposed (Khoury et al., 2014; Novoselov, Pekhovsky, & Simonchik, 2014), they cannot appropriately estimate the true labels and also they assume that there are several samples from a same speaker in the background data which could not be true in reality.

Deep Learning End-to-Ends

It would be interesting to train an end-to-end recognition system which is capable of doing multiple stages of signal processing with a unified DL architecture. In other words, the neural network will be responsible for the whole process from the feature extraction to the final similarity scores. However, working directly on the audio signals in the time domain is still computationally too expensive and, therefore, the current end-to-end DL systems take mainly the handcrafted feature vectors, e.g., MFCCs, as inputs. Recently, there have been several attempts to build an end-to-end speaker recognition system using DL. Nevertheless, most of these works have targeted text-dependent speaker recognition (e.g., Heigold, Moreno, Bengio, & Shazeer, 2016; S.-X. Zhang, Chen, Zhao, Li, & Gong, 2016; Miguel, Llombart, Ortega, & Lleida, 2017; Heo, Jung, Yang, Yoon, & Yu, 2017). In (Heigold et al., 2016), an LSTM was trained to distinguish same-speaker and different-speaker utterance pairs. Similar to (Heigold et al., 2016), another system for text-independent speaker verification was proposed in (Snyder et al., 2016), which is capable to handle variable length input through a temporal pooling layer. This new architecture composed of a feed-forward DNN which takes speech segment as input and maps it to a speaker embedding using a loss function inspired by PLDA. In that work, the frame-level features were

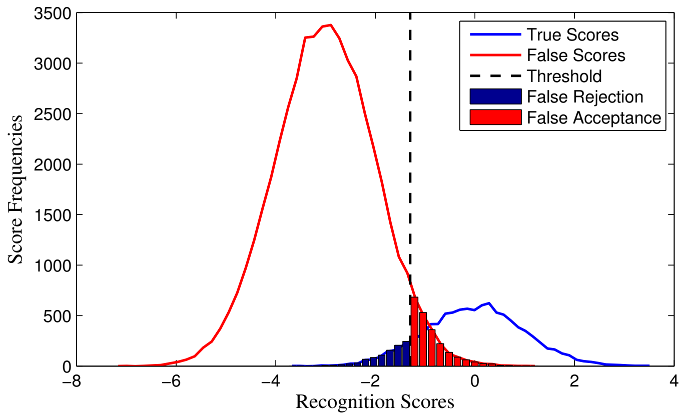


Figure 2.9: An example of the true and false score distributions and the definitions of false acceptance rate (FAR) and false rejection rate (FRR).

grouped into two categories, features of different utterances from the same speaker and features from different utterances of different speakers. Two speaker embeddings are built, each of which corresponding to one of these groups. If these two embeddings are from the same speaker then the loss function should be minimized and if they are from different speakers the loss should be maximized.

2.3 Evaluation Metrics

In the test phase of every speaker verification task, the system gives a real number score per each trial, showing the probability that the claimed speaker is true. Given a reference key, i.e., the true or false label for each trial, there will be two distributions of scores as shown in Fig. 2.9. One score distribution is for those trials which the target and test utterances are from the same speaker, i.e., true scores, and another one is for those trials which target and test speech signals are not from the same speaker, i.e., false scores. Given these two score distributions, there are three commonly in use metrics for speaker verification evaluation, namely Equal Error Rate (EER), the minimum DCF (minDCF), and the Detection Error Tradeoff (DET) curve. All of these metrics are obtained based on two main errors: False Acceptance Rate (FAR) and False Rejection Rate (FRR), called also false alarm and miss rates, respectively. FAR is the relative number of non-target trials accepted incorrectly and FRR is the relative number of target trials rejected incorrectly.

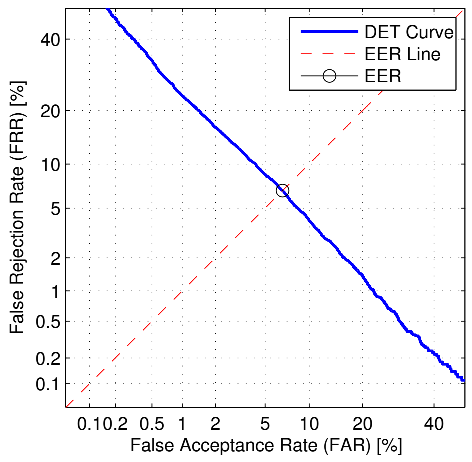


Figure 2.10: An example of DET curve with definition of EER.

As it is shown in Fig. 2.9, these two errors are dependent on the threshold based on which the decision is made. There is a threshold for which FAR and FRR become equal. These equal FAR and FRR are referred to as EER and it is shown typically in percentage. Detection Cost Function (DCF) is a weighted sum of FAR and FRR in terms of the decision threshold t,

$$DCF(t) = \alpha_1 FRR(t) + \alpha_2 FAR(t) \tag{2.39}$$

where the weights α_1 and α_2 are defined based on the application and the type of the evaluation. As we do not like to accept a non-target speaker incorrectly, usually the weight α_2 is much higher than α_1 is speaker recognition evaluations. The minimum value of $DCF(t)$ is referred to as minDCF and is used as one of the main metrics in speaker recognition. As it is shown in Fig. 2.9, there is always a trade-off between FRR and FAR. By increasing the threshold t, FRR increases and, correspondingly, FAR decreases. A DET curve (Fig. 2.10) is a plot of FRR versus FAR over all the operating thresholds. The closer plot to the origin shows a better performance of the system.

Usually, every two years the NIST organizes some Speaker Recognition Evaluation (SRE) to address the most recent challenges in speaker recognition. In this thesis, the NIST SRE 2004-2010 and the NIST 2014 i-vector challenge databases have been used.

Chapter 3

Deep Learning Backend for i-Vector Speaker Verification

The recent compact representation of speech utterances known as i-vector (Dehak, Kenny, et al., 2011) has become the state-of-the-art in the text-independent speaker recognition. There are two common scoring techniques to decide if two i-vectors belong to a same speaker namely cosine and Probabilistic Linear Discriminant Analysis (PLDA) (Prince & Elder, 2007; Kenny, 2010). PLDA scoring leads to a superior performance but with the cost of need to speaker-labeled background data. Moreover, it needs several samples for each background speaker spoken in different session conditions to work efficiently. One of the recent challenges in speaker recognition, which was organized by the National Institute of Standard and Technology (NIST), has been how to fill the performance gap between these two common scoring techniques when no labeled background data is available (NIST, 2014). Although there are some unsupervised automatic labeling techniques like those proposed in (Khoury et al., 2014; Novoselov, Pekhovsky, & Simonchik, 2014), they cannot appropriately estimate the true labels and also they assume that there are several samples from a same speaker in the background data which could not be true in reality. PLDA with estimated labels performs reasonably well (Khoury et al., 2014; Novoselov, Pekhovsky, & Simonchik, 2014), but the results are still far from that of PLDA with actual labels (Greenberg et al., 2014).

On the other hand, the success use of Deep Learning (DL) in speech processing, specifically in speech recognition (e.g., Mohamed et al., 2010; Dahl et al., 2012; Mohamed et al., 2012; Hinton et al., 2012; Senior et al., 2015), has inspired the community to make use of DL techniques in speaker recognition as well. Both generative approaches, like Restricted Boltzmann Machines (RBMs) and Deep Belief Networks (DBNs), and discriminative ones, like Deep Neural Networks (DNNs), have been used for this purpose. A possible use of DL techniques in speaker recognition is to combine them with the state-of-the-art i-vector approach. Two kinds of combination have been considered. DL techniques have been used in the i-vector extraction process (Lei et al., 2014; Kenny et al., 2014; Mclaren et al., 2015; Richardson et al., 2015a; Liu et al., 2015) or applied on i-vectors as a backend (Stafylakis et al., 2012b; Senoussaoui et al., 2012; Stafylakis et al., 2012a; Novoselov et al., 2015; Pekhovsky et al., 2016; Villalba et al., 2017).

In this Chapter, we make use of deep architectures for backend i-vector classification in order to fill the performance gap between the two cosine (unlabeled-based) and PLDA (labeled-based) scoring baseline systems given unlabeled background data. In order to reach this goal, we take advantage of unsupervised learning of DBNs to train a global model referred to as Universal DBN (UDBN) and DNN supervised learning to model each target speaker discriminatively. To provide a balanced training, an impostor selection algorithm and to cope with few training data, a UDBN-adaptation process is proposed.

The preliminary experiments are performed on NIST Speaker Recognition Evaluation (SRE) 2006 (NIST, 2006) to show the effect of each contribution. Taking advantage of the conclusions obtained on the preliminary experiments, another set of experiments are carried out on the new and more challenging database NIST 2014 i-vector challenge (NIST, 2014). The advantage of this database is that the i-vectors for the background, train, and test sets are provided by NIST and, therefore, the baseline systems will have the same performance for everyone to compare new proposed systems with. Three baseline classification techniques are considered: cosine, PLDA with estimated labels, and PLDA with actual labels, which is called also Oracle PLDA system in this thesis. Experimental results performed on NIST 2014 i-vector challenge show that the proposed DL-based system fills 46% of the performance gap between cosine and Oracle PLDA scoring systems in terms of minimum DCF (minDCF) which is similar to the PLDA scoring results obtained with unsupervised estimated labels. The score combination of the proposed DL-based system

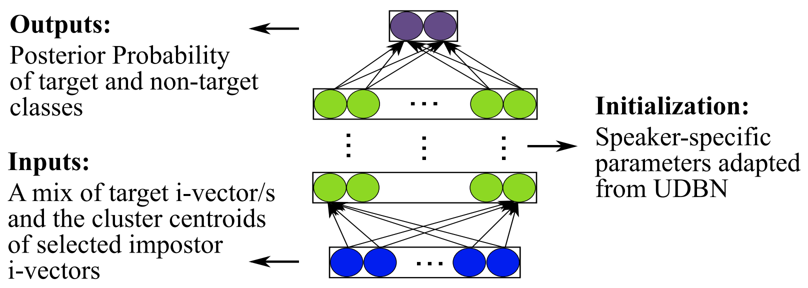

Outputs:
Posterior Probability of target and non-target classes
Inputs:
A mix of target i-vector/s and the cluster centroids of selected impostor i-vectors
Initialization:
Speaker-specific parameters adapted from UDBN

Figure 3.1: Proposed deep learning architecture for training of each speaker model.

and PLDA with estimated labels fills 79% of this gap.

The rest of the Chapter is organized as follows. Section 3.1 presents the proposed DL-based backend for i-vector classification. Section 3.2 describes the proposed impostor selection algorithms in order to have a balanced training. Section 3.3 shows how we will cope with the few amount of data for the training of each target model. Sections 3.4 and 3.5 discuss the experimental results obtained on NIST SRE 2006 and NIST 2014 i-vector challenge, respectively. Section 3.6 concludes the chapter.

3.1 Proposed Architecture

In this Chapter, DL technology is used as a backend in which a two-class hybrid DBN-DNN is trained for each target speaker to increase the discrimination between target i-vector/s and the i-vectors of other speakers (non-targets/impostors) (Fig. 3.1). Proposed networks are initialized with speaker-specific parameters adapted from a global model, which is referred to as Universal Deep Belief Network (UDBN). Then the cross-entropy between the class labels and the outputs is minimized using the back-propagation algorithm.

DNNs usually need a large number of input samples to be trained efficiently. As a general rule, deeper networks require more input data. In speaker recognition, target speakers can be enrolled with only one sample (single session task) or multiple samples (multi-session task). In both cases, the number of target samples is very limited. A network trained with such limited data is highly probable to be overfitted. On the other hand, the number of target and impostor samples will be highly unbalanced,

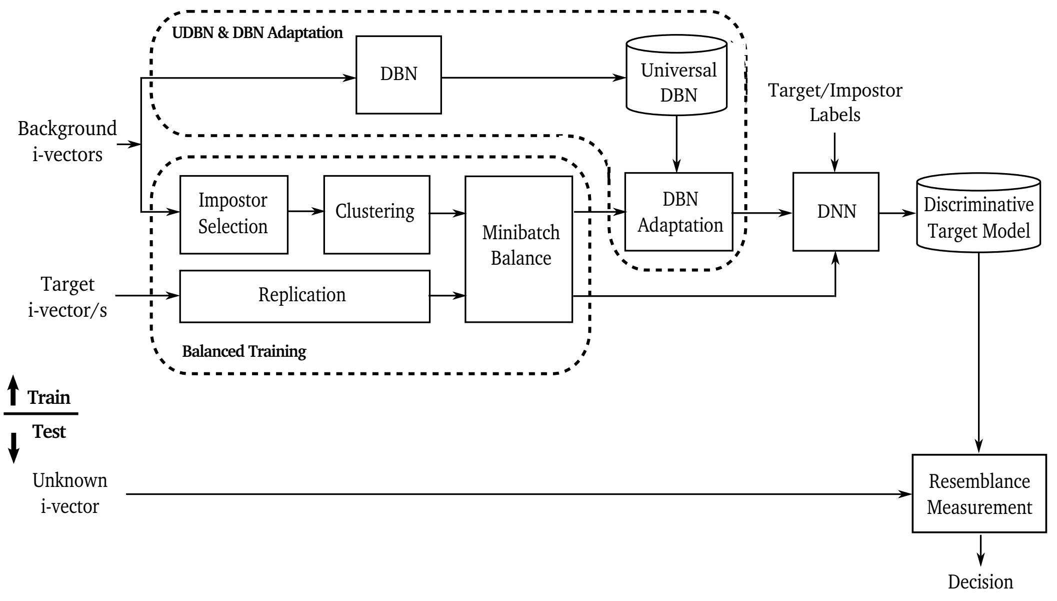


Figure 3.2: Block-diagram of the proposed DL-based backend on i-vectors for target speaker modeling.

i.e., one or some few target samples against thousands of impostor samples. Learning from such unbalanced data will result in biased DNNs towards the majority class. In other words, DNNs will have a much higher prediction accuracy over the majority class.

Fig. 3.2 shows the block diagram of the proposed approach to discriminatively model target speakers. Two main contributions are proposed in this Chapter to tackle the above problems. The balanced training block attempts to decrease the number of impostor samples and, on the contrary, to increase the number of target ones in a reasonable and effective way. The most informative impostor samples for target speakers are first selected by the proposed impostor selection algorithm. Afterwards, the selected impostors are clustered and the cluster centroids are considered as final impostor samples for each target speaker model. Impostor centroids and target samples are then divided equally into minibatches to provide balanced impostor and target data in each minibatch.

On the other hand, the DBN adaptation block is proposed to compensate the lack of input data. As DBN training does not need any labeled data, the whole background i-vectors are used to build a global model, which is referred to as UDBN.

The parameters of the UDBN are then adapted to the balanced data obtained for each target speaker. At the end, given the target/impostor labels, the adapted DBN and the balanced data, a DNN is discriminatively trained for each target speaker. These two contributions are described in more details in the following sections.

3.2 Balanced Training

As speaker models in the proposed method will be finally discriminative, they need both positive and negative data as inputs. Nevertheless, the problem is that the amount of positive and negative data are highly unbalanced in this case, which leads to biasing towards the majority class. Some of the straightforward ways to deal with unbalanced data problem are explored in (He & Garcia, 2009; Thai-Nghe, Gantner, & Schmidt-Thieme, 2010; Khoshgoftaar, Van Hulse, & Napolitano, 2010) (López, Fernández, García, Palade, & Herrera, 2013; Barua, Islam, Yao, & Murase, 2014). A commonly used method is data sampling. The data of the majority class is undersampled and, on the contrary, the data of the minority class is oversampled. The effectiveness of these techniques is highly dependent on the data structure.

In the proposed approach shown in Fig. 3.2, the amount of impostors is decreased in two steps, namely selection and clustering. On the other hand, the amount of target samples is increased by either replication or combination. After that, balanced target and impostor samples are distributed equally among minibatches.

3.2.1 Impostor Selection and Clustering

The objective is to decrease the large number of negative samples in a reasonable way. Our proposal has two main steps. First, only those impostor i-vectors which are more informative for the training dataset are selected. Informative impostor means, in this case, the impostor which is not only representative to a given target but also is statistically close to other targets in the dataset. For a real application, it makes sense to select those impostors who are globally close to all enrolled speakers. When the target speakers are changed, the selected impostors can be re-selected according to the new target dataset. Second, as the number of selected impostor samples is still high in comparison to the number of target ones, they are clustered by the k-means

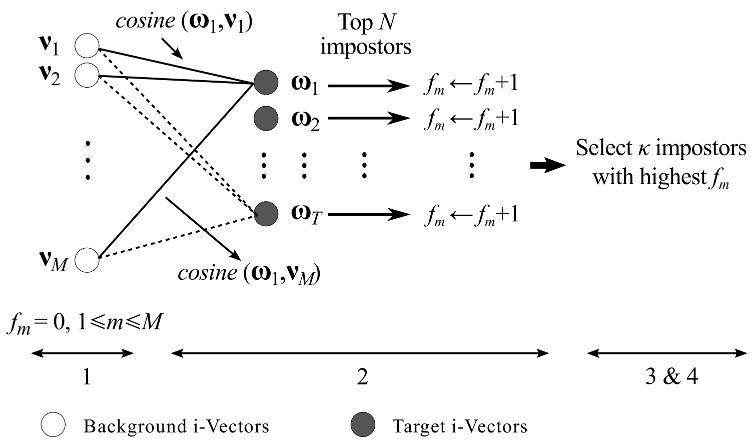


Figure 3.3: Steps of the proposed impostor selection algorithm.

algorithm using the cosine distance criterion. The centroids of the clusters are then used as the final negative samples.

The selection method is inspired from a data-driven background data selection technique proposed in (McLaren, Vogt, Baker, & Sridharan, 2010). In that technique given all available impostor supervectors, a Support Vector Machine (SVM) classifier is trained for each target speaker. The number of times each impostor is selected as a support vector, in all training SVM models, is called impostor support vector frequency (McLaren et al., 2010). Impostor examples with higher frequencies are then selected as the refined impostor dataset. However, SVM training for each target speaker would be computationally costly. Moreover, as our final discriminative models will be DNNs, it would not be worth to employ this technique as such. Instead, we have proposed to use cosine similarity as an efficient and a fast criterion for comparing i-vectors. We compare each target i-vector with all impostor i-vectors in the background dataset. Those N impostors which are close to each target i-vector are treated like support vectors in (McLaren et al., 2010). Then the κ impostors with the highest frequencies are selected as the most informative impostors.

The N and κ selected impostors are referred to as local and global selected impostors in this work. We will show in section 3.5 that the pooling of the global and local selected impostors, for each target model before clustering, will improve the results depending on the background data. However, the computational cost will be higher as the k-means algorithm should be performed individually for each target

Algorithm 1: Proposed target database-dependent impostor selection algorithm.

```
Input: Target i-vectors $\boldsymbol{\nu}_i$, $1 \leq i \leq \mathcal{I}$ and Background i-vectors $\boldsymbol{\omega}_m$,
       $1 \leq m \leq \mathcal{M}$
Output: Selected impostor i-vectors
1  Initialization: Set impostor frequencies $f_m \leftarrow 0$ for each $\boldsymbol{\omega}_m$
2  for each target i-vector $\boldsymbol{\nu}_i$ do
3      for each background i-vector $\boldsymbol{\omega}_m$ do
4          Compute $score_{i,m} = cosine\left(\boldsymbol{\nu}_i, \boldsymbol{\omega}_m\right)$
5      end
6      Select the $N$ background i-vectors with the highest scores
7      For the selected i-vectors $f_m \leftarrow f_m + 1$
8  end
9  Sort background i-vectors in descending order based on their $f_m$
10 Select the first $\kappa$ i-vectors as the final impostors
```

model while it is performed only once when only the global selected impostors are used. The parameters N and κ are determined experimentally. The whole algorithm is shown in Fig. 3.3 and can be summarized as in Algorithm 1 in which $cosine\left(\boldsymbol{\nu}_i, \boldsymbol{\omega}_m\right)$ is the cosine score between target i-vector $\boldsymbol{\nu}_i$ and the background i-vector $\boldsymbol{\omega}_m$ and $\mathcal{M}$ and $\mathcal{I}$ are the number of background and target i-vectors, respectively. Note that in case of multi-session target enrollment, the average of the available i-vectors per each target speaker will be considered in the algorithm above.

We have also proposed a similar algorithm in which the selection process is only dependent on the background data. A randomly selected subset from the background data is used in the Algorithm 1 rather than the target training database. In order to make the process statistically more reliable, the whole process is repeated several times and the impostor frequencies are accumulated over all iterations. We have shown that this algorithm performs similar to Algorithm 1, which uses the training target set in the selection process, when the background database is large enough. The full algorithm can be summarized as in Algorithm 2 in which $\mathcal{I}$ is an arbitrary size for B_1, typically chosen as the same size as the target dataset in Algorithm 1, and t_{max} is the number of iterations.

Algorithm 2: Proposed target database-independent impostor selection algorithm.

```
   Input: Background i-vectors $\boldsymbol{\omega}_m$, $1 \leq m \leq \mathcal{M}$
   Output: Selected impostor i-vectors
1  Initialization: Set impostor frequencies $f_m \leftarrow 0$ for each $\boldsymbol{\omega}_m$ and $t \leftarrow 1$
2  while $t \leq t_{max}$ do
3      Divide randomly the background i-vector dataset $B = \{\boldsymbol{\omega}_m\}$ into
         $B_1 = \{\boldsymbol{\nu}_i, 1 \leq i \leq \mathcal{I}\}$ and $B_2 = \{\boldsymbol{\chi}_j, 1 \leq j \leq \mathcal{M} - \mathcal{I}\}$, where
         $B_1 \cup B_2 = B$ and $B_1 \cap B_2 = \phi$
4      for each $\boldsymbol{\nu}_i \in B_1$ do
5          for each $\boldsymbol{\chi}_j \in B_2$ do
6              Compute $score_{i,j} = cosine\,(\boldsymbol{\nu}_i, \boldsymbol{\chi}_j)$
7          end
8          Select the $N$ i-vectors $\in B_2$ with the highest scores
9          For the selected i-vectors $f_m \leftarrow f_m + 1$
10     end
11     $t \leftarrow t + 1$
12 end
13 Sort background i-vectors $\in B$ in descending order based on their $f_m$
14 Select the first $\kappa$ i-vectors as the final impostors
```

3.2.2 Target Replication or Combination

In order to balance positive and negative samples, the number of target samples is increased as many as the number of impostor cluster centroids obtained in section 3.2.1. In the single session enrollment task, the i-vector of each target speaker is simply replicated as many as the number of cluster centroids. Replicated target i-vectors will not act exactly the same as each other in the pre-training process of DNNs due to the sampling noise created in RBM training (Hinton, 2012). Moreover, in both adaptation and supervised learning stages the replicated versions make the target and impostor classes having the same weights when the network parameters are being updated. In multi-session task, the available i-vectors of each target speaker can be combined, i.e., the average of every n i-vectors is considered as a new target i-vector.

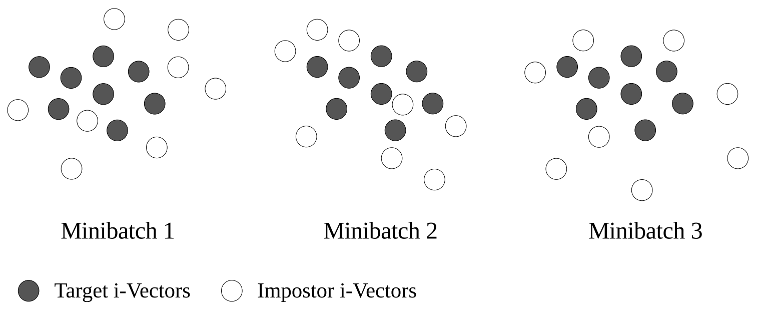


Figure 3.4: Balanced training for DNNs in multi-session speaker verification task. In each mini-batch the same target i-vectors but different impostors are shown to DNNs.

Once the number of positive and negative samples are balanced, they are distributed equally among minibatches. In other words, each minibatch contains the same number of impostors and targets. If target samples in the multi-session task are not combined, the same target samples but different impostor ones are shown to the network in each minibatch (Fig. 3.4). The optimum numbers of impostor clusters and minibatches will be determined experimentally in sections 3.4 and 3.5.

3.3 Universal DBN and Adaptation

Unlike DNNs, which need labeled data for training, DBNs do not necessarily need such labeled data as inputs. Hence, they can be used for unsupervised training of a global model referred to as UDBN. UDBN is trained by feeding background i-vectors from different speakers. The training procedure is carried out layer by layer using RBMs as described in section 2.2.2. As the input i-vectors are real-valued, a Gaussian-Bernoulli RBM (GRBM) (Hinton, 2012; Dahl et al., 2012) is used to train the connection weights between the visible and the first hidden layer units. The rest of the connection weights are trained with Bernoulli-Bernoulli RBMs.

It is shown that pre-training techniques can initialize DNNs better than simply random numbers (Larochelle et al., 2009; Dumitru et al., 2009; Erhan et al., 2010). However, when a few input samples are available, just pre-training may not be enough to achieve a good model. In this case, we have proposed to adapt UDBN parameters to the balanced data obtained for each target speaker. Adaptation is carried out by training a DBN which is initialized by the parameters of the UDBN

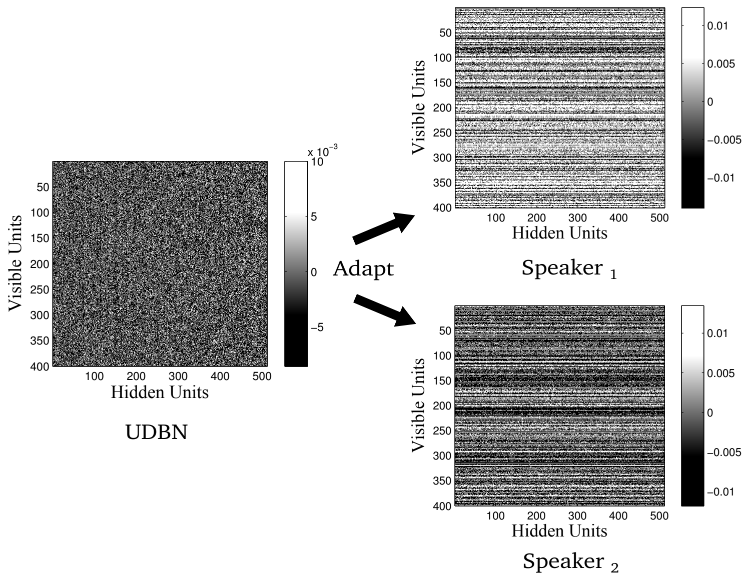


Figure 3.5: Comparison of the adapted connection weights between the visible and the first hidden units for two different speakers.

given the balanced data of each target speaker. Adapted DBNs are then used as an initialization for the final DNN target models. In order to avoid overfitting, only a few iterations will be considered for adaptation. It is supposed that UDBN can learn both speaker and channel variabilities from the background data. Therefore, UDBN will provide a more meaningful initial point for DBNs than a simple random initialization. The study in (Dumitru et al., 2009) has shown that pre-training is robust with respect to the random initialization seed. The use of UDBN parameters makes target models almost independent from the random seeds.

In order to facilitate the training of the networks specifically where more than one hidden layer is used, we normalize the UDBN parameters before adaptation. Normalization is carried out by simply scaling down the maximum absolute value of connection weights to 0.01. In this way, connection weights will have a dynamic range similar to that typically used for random initialization. Additionally, bias terms are multiplied by 0.01 to be closer to zero. This is because the bias terms are usually set to zero when the connection weights are randomly initialized. In this way, the same learning rates and the number of epochs tuned for random initialized DNNs can also be used for adapted DNNs in the supervised learning stage.

Fig. 3.5 shows the comparison of the adapted UDBN connection weights, between the input layer and the first hidden layer, for two different speakers. As it can be seen in this figure, speaker-specific initial points are set by the adaptation process for each DNN target model. Once the adaptation process is completed, a DNN is initialized with the adapted DBN parameters for each target speaker. Given target/impostor labels, the minibatch stochastic gradient descent back-propagation is then carried out for fine-tuning. The softmax and the logistic sigmoid will be the activation functions of the top label layer and the other hidden layers, respectively.

We have proposed to compute the output scores in Log Posterior Ratio (LPR) forms as,

$$\Lambda(\text{target}|\boldsymbol{\omega}) = \log P(\text{target}|\boldsymbol{\omega}) - \log P(\text{non-target}|\boldsymbol{\omega}) \tag{3.1}$$

where $P(\text{target}|\boldsymbol{\omega})$ and $P(\text{non-target}|\boldsymbol{\omega})$ are, respectively, the posterior probability of the target and non-target classes given the test i-vector $\boldsymbol{\omega}$. LPR computation helps to Gaussianize the true and false score distributions which can be useful for score fusion.

In addition, to make the fine-tuning process more efficient a momentum factor is used to smooth out the updates, and the weight decay regularization is used to penalize large weights.

3.4 Experiments on NIST SRE 2006

NIST SRE 2006 (NIST, 2006) is used to show the effect of each proposed contribution shown in Fig. 3.2 for both single and multi-session speaker verification tasks. In these experiments, we have built the whole system from scratch including Voice Activity Detection (VAD) and feature and i-vector extraction. Taking advantage of the conclusions of this section, the NIST 2014 i-vector challenge dataset (NIST, 2014) is used in section 3.5 to compare the performance of the proposed system with the most recent state-of-the-art baseline systems.

3.4.1 Baseline and Database

Two sets of experiments are performed is this section. The whole core test condition of SRE 2006 is used as a single session task and 8 conversation side training condition is used as the multi-session task. In both cases, training and test signals have approximately two-minute total speech duration. There are 816 target models and 51,068 trials in the single session and 699 target models and 31,080 trials in the multi-session task. Speech signals with the two-minute approximate duration from NIST SRE 2004 and 2005 are used as the background data containing 6063 speech signals from 1070 distinct speakers. It is worth noting that in case of NIST 2005 only the speech signals of those speakers who do not appear in NIST SRE 2006 are used.

Frequency Filtering (FF) features (Nadeu et al., 2001) are used in these experiments. FFs, like Mel-Frequency Cepstral Coefficients (MFCCs), are decorrelated version of log Filter Bank Energies (FBE) (Nadeu et al., 2001). It has been shown that FF features achieve a performance equal to or better than MFCCs (Nadeu et al., 2001). Features are extracted every 10 msec using a 30 msec Hamming window. The number of static FF features is 16 and along with delta FF and delta log energy, 33-dimensional feature vectors are built. Before feature extraction, speech signals are subject to an energy-based silence removal process. The gender-independent Universal Background Model (UBM) is represented as a diagonal covariance, 512-component Gaussian Mixture Model (GMM). All the i-vectors are 400-dimensional. The i-vector extraction process is carried out using ALIZE open source software (Larcher et al., 2013) with the minimum divergence algorithm (Kenny, Ouellet, Dehak, Gupta, & Dumouchel, 2008). Since the minimum divergence training algorithm is used for these experiments, i-vectors are already zero-mean unit-variance Normal-distributed and, therefore, no post-processing is carried out. UBM, $\boldsymbol{T}$ matrix, and PLDA parameters are trained using the same background data. PLDA baseline systems are gender-independent with a 250-dimensional speaker space. For PLDA experiments, i-vectors are length normalized. The performance is evaluated using Detection Error Tradeoff (DET) curves, the Equal Error Rate (EER), and the minDCF with $\alpha_1 = 0.1$ and $\alpha_2 = 0.99$ (eq. 2.39) defined as in (NIST, 2006).

3.4.2 Single Session Experiments

For DNN experiments, the size of hidden layers is set to 512. DNNs with up to three hidden layers are explored in all experiments. We do not go further than three layers because of few amount of data and increasing the computational complexity without more significant gain. The number of minibatches and the number of impostor

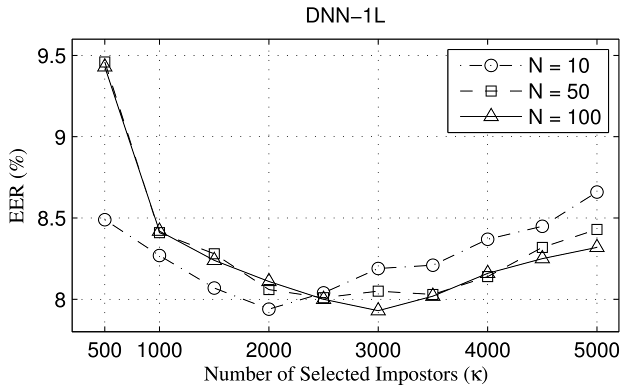


Figure 3.6: Parameter setting of the proposed impostor selection algorithm for one hidden layer DNNs. N and κ are, respectively, the number of local and global nearest impostor i-vectors to target i-vectors.

centroids are set experimentally to 3 and 12, respectively. Each minibatch will include four impostor centroids and four replicated target samples.

As a DNN baseline system, we train a DNN for each target speaker using the whole impostor background data and random initialization. In this case, the whole background i-vectors are clustered using the k-means algorithm and the centroids are considered as impostor samples. In this work, we use the uniform distribution $\mathcal{U}(0, 0.01)$ for random initialization as the experimental results showed that it achieves slightly better performance than the normal distribution $\mathcal{N}(0, 0.01)$ used in the prior work. We tune the parameters of the networks and keep them fixed in all other experiments. DNN-1L, DNN-2L, and DNN-3L are trained with the learning rates of 0.001, 0.005, and 0.08 and with the number of epochs of 30, 100, and 500, respectively. DNN-3L stands for a three hidden layer DNN. Momentum and weight decay are set, respectively, to 0.9 and 0.0012 for all DNNs.

Background i-vectors are extracted from the same speech signals used for training UBM and $\mathbf{T}$ matrix. The two parameters N and κ, the number of local and global selected impostors in the proposed impostor selection algorithm, need to be determined experimentally. For SRE 2006 experiments, we have used the first proposed impostor selection algorithm in which the target training data set is employed (Algorithm 1). Fig. 3.6 illustrates the variability of EER in terms of these two parameters for one hidden layer DNNs. The similar behavior can be observed for minDCF curves. DNN examples shown in this figure are initialized randomly. Based on this

Table 3.1: The effect of each proposed idea of Fig. 3.2 on the performance of the DNN systems. Results are obtained on the core test condition of NIST SRE 2006. The cosine and PLDA Baseline systems achieve (EER=7.18%, minDCF=324) and (EER=4.78%, minDCF=253), respectively.

Impostor Selection	**Adaptation**	**EER** (%)			**minDCF** ($\times 10^4$)		
		# Hidden Layers			# Hidden Layers		
		1	2	3	1	2	3
–	–	8.55	7.76	7.59	381	353	351
✓	–	8.06	7.12	7.09	360	327	326
–	✓	7.43	7.47	7.45	339	343	339
✓	✓	**6.81**	6.97	6.99	315	317	313
Fusion with cosine		6.83	**6.88**	**6.64**	**308**	**309**	**299**
Fusion with PLDA		4.98	5.03	**4.76**	253	**248**	**230**

figure, for DNN-1L we set N and κ to 10 and 2000, respectively. Similar curves are plotted for other networks and N is set to 10 for all of them and κ is set to 300 and 500 for DNN-2L and DNN-3L, respectively.

Experimental results showed that the main improvement due to the adaptation process comes from the adaptation of the connection weights between the input layer and the first hidden layer for all DNNs. The adaptation of the other layers has no significant impact on the performance. In order to decrease the probability of overfitting during the adaptation, a separate network is adapted to each minibatch and then the parameters of the obtained networks are averaged. For DNN-1L and DNN-2L we adapted all layers and for DNN-3L only the first two layers. The learning rate of adaptation is set to 0.001 and 0.0001 for the first and the second layers, respectively. The number of epochs for the first layer is set to 10, 20, and 15 for DNN-1L, DNN-2L, and DNN-3L, respectively. The number of epochs for the second layer is set, respectively, to 15 and 20 for DNN-2L and DNN-3L.

Table 3.1 summarizes the effect of each proposed contribution. In the first row of the table, DNNs are initialized randomly and the impostor cluster centroids are obtained on the whole background data. As it can be seen in this row, adding more hidden layers to the network improves the performance. However, they still work worse than the baseline system in which i-vectors are classified using cosine distance. EER and minDCF for the baseline system are 7.18% and 0.0324, respectively. Impostor selection improves the performance to a great extent for all the networks. We have tried global, local, and the pooling of global and local selected impostors before k-means clustering and the best performance was obtained by using only global selected impostors. The biggest improvement due to the adaptation process is observed in DNNs with one hidden layer. The best results are obtained using both impostor selection and adaptation techniques which show an 8-20% and 10-17% relative improvements in terms of EER and minDCF, respectively, compared to the baseline DNNs. The biggest relative improvements are achieved on DNN-1L. The last two rows of the table show the fusion of DNN systems with the cosine and PLDA (EER=4.78%, minDCF=0.0253) baseline systems. Scores of each system are first mean and variance normalized and then simply summed. The fusion of the cosine baseline and DNN systems improves the results and DNN-3L achieves the best results corresponding to an 8% relative improvement for both EER and minDCF in comparison to the cosine scoring baseline system. Nevertheless, only DNN-3L scores can improve the PLDA results specifically for minDCF by 9% relative improvement. We have also combined the scores of DNNs with different number of hidden layers, but no gain is observed.

The DET curve in Fig. 3.7 compares the best systems in all operating points. As it is shown in this figure, DNNs with one hidden layer achieve better results than the baseline and the combination of 3-layer DNNs with the baseline works the best in all operating points.

3.4.3 Multi-Session Experiments

The same configuration used for the single session task is also applied for the multi-session one. The number of minibatches is set to 3. In each minibatch, all 8 target i-vectors accompanying with 8 impostor cluster centroids are shown to the network. Therefore, the size of each minibatch and the total number of impostor clusters will be 16 and 24, respectively. As the combination of the i-vectors of each target speaker did not help the training of the networks, we replicated the target i-vectors in every minibatch as it was shown in Fig. 3.4. We train the networks with the same parameters tuned for the single session experiments.

Results are summarized in Table 3.2. Around 12% relative improvements are achieved in all DNNs employing impostor selection technique proposed in this work. With the same parameters obtained for the single session task, we re-selected the impostors for the new multi-session data set. The adaptation process improves the

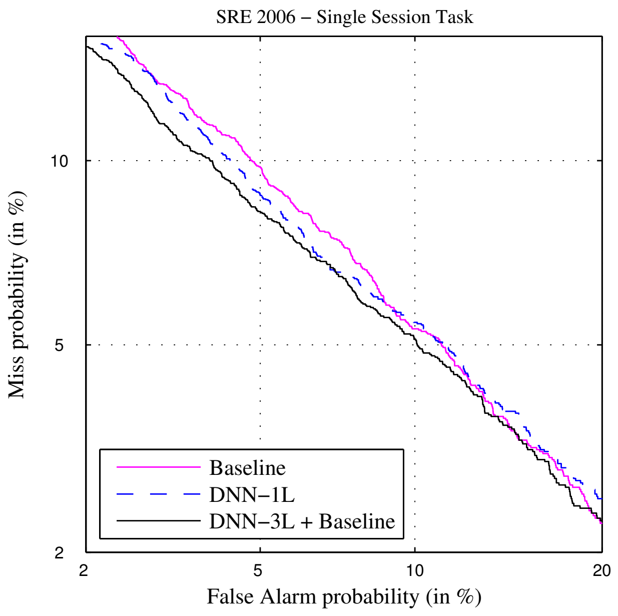


Figure 3.7: Comparison of the performance of the proposed DNN based systems with the baseline system (i-vector + cosine). DET curves are obtained on the core test condition of NIST SRE 2006.

performance up to 8%. As in the single session task, adaptation is more effective for one-hidden-layer DNNs. For all the networks, only the parameters of the first hidden layer are adapted because no more improvement was observed adapting the other layers. Adaptation is carried out by the learning rate of 0.001 for all DNNs and the number of epochs of 10, 10, and 25 for DNNs with one to three layers, respectively. The best results are obtained with DNN-3L when the two proposed techniques are combined. It shows more than 20% relative improvements of EER and minDCF in comparison to the baseline three-layer DNNs.

The proposed three-hidden-layer DNNs show a performance between the cosine (EER=4.2%, minDCF=0.0191) and PLDA (EER=2.27%, minDCF=0.0105) baseline systems, with more than 17% and 10% relative improvements in terms of EER and minDCF, respectively, compared to the cosine scoring. Fusion with the cosine baseline system improves the results in all cases, but no improvement is observed by combination with PLDA scores. Fusion is effective mostly on the minDCF which increase the improvement from 10% to 15%.

Fig. 3.8 compares the DET curves of the best results obtained in table 3.2. As it can be seen in this figure, DNN-3L outperforms clearly the baseline and the DNN-

Table 3.2: The effect of each proposed idea of Fig. 3.2 on the performance of the proposed DNN systems. Results are obtained on NIST SRE 2006, 8-session enrollment task. The cosine and PLDA Baseline systems achieve (EER=4.2%, minDCF=191) and (EER=2.27%, minDCF=105), respectively.

Impostor Selection	**Adaptation**	**EER** (%)			**minDCF** ($\times10^4$)		
		# Hidden Layers			# Hidden Layers		
		1	2	3	1	2	3
–	–	4.58	4.58	4.38	208	213	217
✓	–	4.02	4.07	3.86	183	201	194
–	✓	4.24	4.30	4.20	202	207	202
✓	✓	3.68	3.83	3.50	170	189	172
Fusion with cosine		**3.61**	**3.77**	**3.45**	**161**	**169**	**162**
Fusion with PLDA		2.46	2.62	2.36	111	121	112

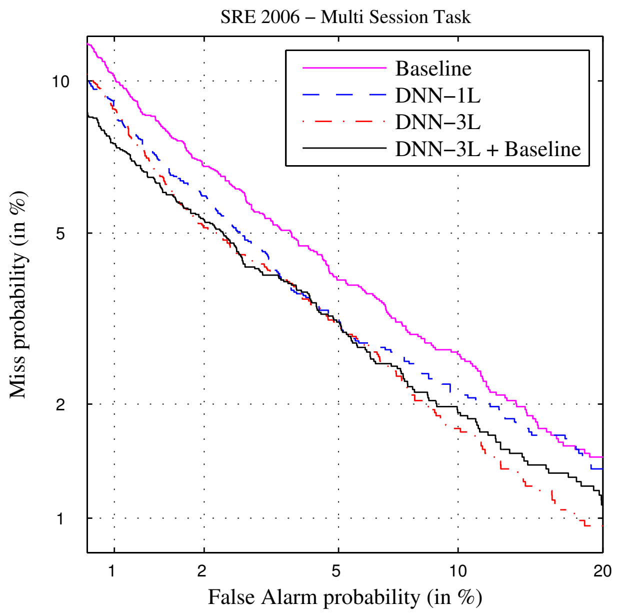


Figure 3.8: Comparison of the performance of the proposed DNN based systems with the baseline system (i-vector + cosine). DET curves are obtained on the 8-session enrollment task of NIST SRE 2006.

1L in all operating points. However, fusion with the baseline system improves the performance only for the operating points with lower false alarm probabilities.

3.5 Experiments on NIST 2014 i-Vector Challenge

The full database provided in the NIST 2014 speaker recognition i-vector challenge (NIST, 2014) is used for the experiments in this section. Rather than speech signals, i-vectors are given directly by NIST in this challenge to train, test, and develop the speaker recognition systems. This enables system comparison more readily with consistency in the front-end and in the amount and type of the background data (NIST, 2014). For this challenge, speaker recognition systems are evaluated in two phases: when the speaker labels of the background data are not known and when they are known to the systems. The cosine and PLDA scoring techniques are used by NIST as the baseline systems when unlabeled and labeled background data are available, respectively. The goal of this evaluation is to see how other techniques can fill the performance gap between these two baseline systems when no labeled background data is available.

3.5.1 Baseline and Database

Conventional telephone speech recordings from NIST SRE 2004 to 2012 are used to compute i-vectors for this challenge (Greenberg et al., 2014). Unlike NIST SRE 2006 experiments, in which the duration of speech signals for each i-vector was approximately 2 minutes, in this challenge i-vectors are extracted from speech utterances of varying duration with a mean of 39.6 seconds. Three sets of 600-dimensional i-vectors are provided: development, train, and test consisting of 36,572, 6530, and 9634 i-vectors, respectively. The number of target speaker models is 1306 and for each of them five i-vectors are available. Each target model will be scored against all the test i-vectors and, therefore, the total number of trials will be 12,582,004. Trials are divided by NIST into two randomly selected subsets: a progress subset (40%), and an evaluation subset (60%). The performance is evaluated using a minDCF with $\alpha_1 = 1$ and $\alpha_2 = 100$ (eq. 2.39) recommended by NIST in (NIST, 2014).

Three baseline systems are considered in this work for evaluation: cosine, PLDA with actual labels, and PLDA with estimated labels. In all of them, i-vectors are whitened and length normalized prior to evaluation and the average of the i-vectors per each target speaker is used as a single target model. For the cosine baseline system, the averaged target i-vectors are length normalized again while for the PLDA baseline systems it is shown that re-normalization affects the performance (Greenberg et al., 2014). Both PLDA systems are gender-independent with a 400-dimensional speaker space. In order to have the best PLDA with actual labels, those background i-vectors extracted from speech signals shorter than 30 seconds are discarded before PLDA training (Greenberg et al., 2014). For the PLDA with estimated labels, a two stage unsupervised clustering technique is used to estimate the speaker labels of the background data. The first stage of the clustering algorithm is similar to the Mean Shift based algorithm proposed in (Senoussaoui, Kenny, Stafylakis, & Dumouchel, 2014) and used successfully in NIST 2014 i-vector challenge (Novoselov, Pekhovsky, & Simonchik, 2014). In the second stage, the closer clusters obtained in the first stage are combined. In both stages, i-vectors are joined based on the cosine similarity considering a threshold which is set to 0.29 in our experiments as in (Novoselov, Pekhovsky, & Simonchik, 2014). At the end, only clusters contained no less than 4 and no more than 50 i-vectors are selected. As in (Novoselov, Pekhovsky, & Simonchik, 2014), those i-vectors with less than 20 seconds of speech are discarded before PLDA training in this case. It is possible to train a PLDA with the estimated labels and repeat the two stage unsupervised clustering algorithm with the PLDA similarity, but it would be time consuming, and no significant gain will be observed in practice. The experimental results for this baseline system show a comparable performance to those reported in (Novoselov, Pekhovsky, & Simonchik, 2014) and (Khoury et al., 2014).

3.5.2 Multi-Session Experiments

The same architecture of SRE 2006 multi-session experiments with some modification is used for these experiments. The size of hidden layers is set to 400. Each minibatch consists of 5 impostor centroids and 5 target samples. The total number of impostor centroids is 15 for each target model. Since DNN-1L and DNN-3L worked better than DNN-2L in SRE 2006 experiments, we only implement these two networks for NIST 2014 i-vector challenge. DNN-1L and DNN-3L are trained with the learning rates of 0.002 and 0.07 and with the number of epochs of 30 and 300, respectively. Momentum and weight decay are set, respectively, to 0.9 and 0.001 for all DNNs. For UDBN training, the learning rate and the number of epochs are set to 0.02 and 200 for GRBM, and to 0.06 and 120 for the rest of RBMs, respectively. Momentum, weight decay, and the minibatch size are set, respectively, to 0.9, 0.0002, and 100 for

Table 3.3: Performance comparison of the proposed DNN system with other baseline systems on NIST 2014 i-vector challenge.

Unlabeled Background Data	**Progress Set**		**Evaluation Set**	
	EER (%)	minDCF	EER (%)	minDCF
[1] cosine	4.78	0.386	4.46	0.378
[2] PLDA (Estimated Labels)	3.85	0.300	3.46	0.284
[3] Proposed DNN-1L	5.13	0.327	4.61	0.320
[4] Proposed DNN-3L	4.55	0.305	4.11	0.300
Fusion [2] & [4]	**2.99**	**0.260**	**2.70**	**0.243**
Labeled Background Data				
[5] PLDA (Actual Labels)	2.23	0.226	2.01	0.207
Fusion [2] & [5]	2.04	0.220	1.85	0.204
Fusion [4] & [5]	2.13	0.221	2.00	0.196
Fusion [2] & [4] & [5]	**1.88**	**0.204**	**1.74**	**0.190**

all RBMs. For DNN-3L we adapted only the first two layers. The learning rate and the number of epochs of adaptation are set, respectively, to 0.001 and 10 for the first layer and to 0.0001 and 20 for the second layer.

As the background dataset is big enough for these experiments, we have used the impostor selection algorithm which is only dependent on the background data (Algorithm 2, section 3.2.1). The results will be slightly better if we use the training data set in the selection algorithm, but the system may not be robust enough to unseen data. As in SRE 2006 experiments, we have tried global, local, and the pooling of global and local selected impostors before k-means clustering and the best performance was obtained by pooling. For global impostor selection, κ and N are set to 4,500 and 100 for both DNN-1L and DNN-3L, respectively. The algorithm is iterated 20 times. Afterwards, the global selected impostors are pooled with 500 local impostors for each target speaker before k-means clustering.

Table 3.3 compares the performance of the proposed DNN systems with other baseline systems in terms of minDCF and EER and Figs. 3.9 and 3.10 compares them in all operating points in terms of DET curves. Circles in the figures show the operating points corresponding to minDCFs. It is worth noting that in the NIST 2014 i-vector challenge the performance of the systems were evaluated only in terms of minDCF. However, we have also included EERs in the table for better comparison. As it can be seen in the table, the proposed DNN-3L performs better

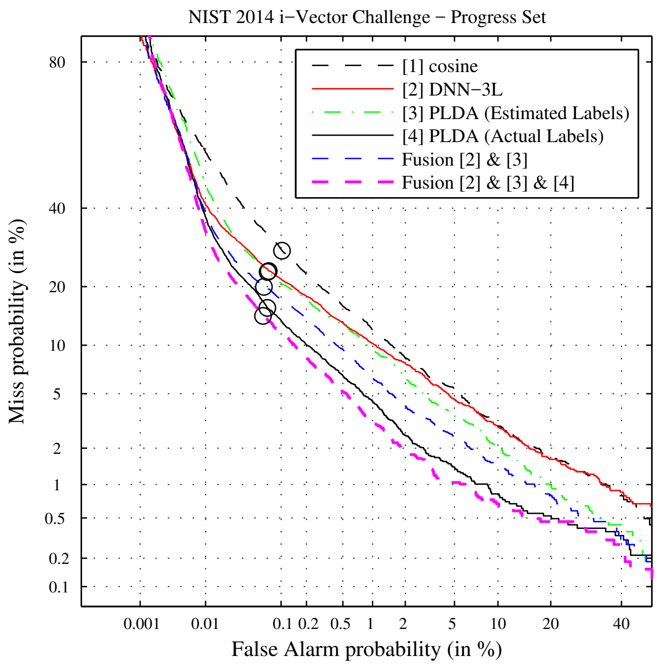


Figure 3.9: Comparison of the performance of the proposed DNN-3L system with other baseline systems on the progress set of NIST 2014 i-vector challenge.

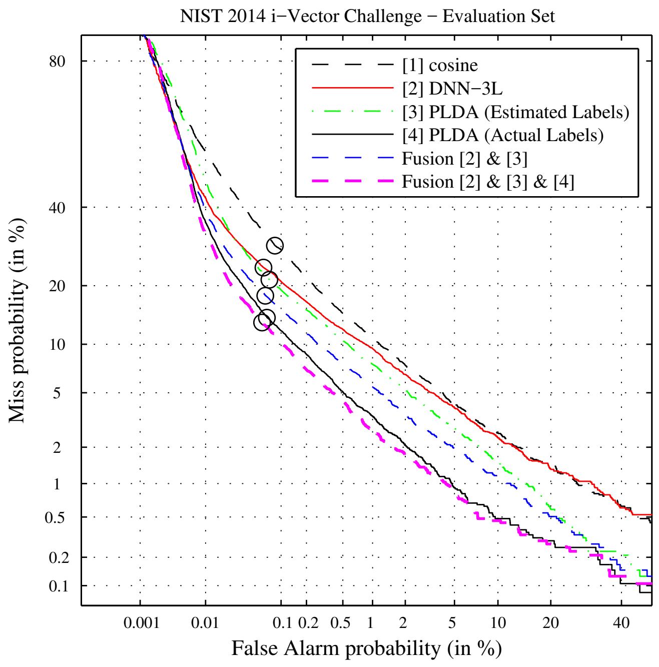


Figure 3.10: Comparison of the performance of the proposed DNN-3L system with other baseline systems on the evaluation set of NIST 2014 i-vector challenge.

than DNN-1L, as it was concluded from SRE 2006 experiments. The proposed DNN-3L system achieves comparable performance to PLDA with estimated labels in terms of minDCF (with 21% relative improvement compared to cosine scoring), but lower performance in terms of EERs. In other words, as it is shown in Figs. 3.9 and 3.10, the proposed DNN-3L system performs closer to PLDA with actual labels than to cosine for lower False Alarm (FA) probabilities. The proposed DNN and PLDA with actual labels achieve the same performance for FA probability around 0.01, and for lower than 0.01 the proposed DNN system outperform the PLDA with actual labels. For higher security purposes, it is more important in speaker recognition to have better performance in lower FA probabilities.

The interesting point is that the combination of the DNN-3L and PLDA with estimated labels in the score level improves the results to a great extent in all operating points. The resulting relative improvement compared to cosine baseline system is 36% in terms of minDCF on the evaluation set. This improvement with no use of background labels is considerable compared to 45% relative improvement which can be obtained by PLDA with actual labels. The score fusion is carried out using BOSARIS toolkit (Brummer & Villiers, 2011). The combination weights are trained on the progress trial set and used for the evaluation set.

As it can be seen in Table 3.3 in both cases of DNN-3L and PLDA with estimated labels, the combination with PLDA improves the results. This improvement is higher in terms of EER for PLDA with estimated labels and in terms of minDCF for DNN-3L systems. Nevertheless, the combination of all three systems achieves the best performance, corresponding to 8% relative improvement, in terms of minDCF, compared to the PLDA with actual labels.

3.6 Conclusion

A hybrid architecture based on DBN and DNN has been proposed in this work to discriminatively model each target speaker for i-vector speaker verification. The main objective has been to fill the performance gap between the cosine and the Oracle PLDA scoring systems when no labeled background data is available. Two main contributions have been proposed to make DNNs more efficient in this particular task. Firstly, the most informative impostor i-vectors have been selected and clustered to provide a balanced training. Secondly, each DNN has been initialized with the speaker specific parameters adapted from a global model, which has been referred to as UDBN. In order to have more insight into the behavior of these techniques in both single and multi-session speaker enrollment tasks, the experiments have been carried out in both scenarios. Experiments were performed on NIST SRE 2006, mainly for development, and on NIST 2014 i-vector challenge, mainly for evaluation. It was shown that the proposed hybrid system fills approximately 46% of the performance gap between the cosine and the Oracle PLDA scoring systems in terms of minDCF. Although the proposed system still does not outperform the baseline PLDA with estimated labels, their score fusion is highly effective and covers 79% of this gap. The reason that the proposed system still does not outperform the baseline PLDA system could be that it does not explicitly compensate the session variability as it is carried out in PLDA. Thus, it is expected that adding some explicit session modeling to the proposed hybrid model could improve the performance, but it has been beyond the scope of this thesis.

Chapter 4

RBMs for Vector Representation of Speech

Over the last few years, i-vectors have shown a great performance not only in speaker recognition but also in other applications (e.g., Xia & Liu, 2012; Bahari, McLaren, Van Hamme, & Van Leeuwen, 2012). Motivated by the success use of Deep Learning (DL) in other speech processing applications, DL techniques have also been used in the i-vector extraction process in two ways. First, a Deep Neural Network (DNN) has been used for acoustic modeling rather than the typical Gaussian Mixture Models (GMMs) (Lei et al., 2014; Kenny et al., 2014; W. M. Campbell, 2014; Richardson et al., 2015a; Garcia-Romero et al., 2014; Liu et al., 2015). Second, conventional spectral features have been replaced or appended by the so-called DNN bottleneck features and then a DNN or a GMM has been used as an acoustic model (Mclaren et al., 2015; Richardson et al., 2015a; Liu et al., 2015). The main problem is that the traditional i-vector extraction itself is computationally expensive for real-time applications and the use of DNN as either an acoustic model or bottleneck feature extractor increases highly the computational cost. Moreover, in both cases phonetic labels are required for DNN training, which are not always accessible.

The aim of this chapter is to develop an efficient alternative vector representation of speech by keeping the computational cost as low as possible and avoiding phonetic labels, which are not always accessible. The proposed vectors will be based on both GMM and Restricted Boltzmann Machine (RBM) and will be referred to as

GMM-RBM vectors. The role of RBM is to learn the total speaker and session variability among background GMM supervectors. This RBM, which will be referred to as Universal RBM (URBM), will then be used to transform unseen supervectors to the proposed low dimensional vectors. The use of different activation functions for training the URBM and different transformation functions for extracting the proposed vectors are investigated. At the end, a variant of Rectified Linear Unit (ReLU) which is referred to as Variable ReLU (VReLU) is proposed.

The core condition of NIST SRE 2006 (NIST, 2006) is used for the development and the core condition 5 of NIST SRE 2010 (NIST, 2010) with much bigger background data is used for the test and evaluation. The experiments on the evaluation set shows that the proposed GMM-RBM vectors achieve comparable performance with traditional i-vectors while much lower computational cost is required for vector extraction. The conclusion is valid with both cosine and Probabilistic Linear Discriminant Analysis (PLDA) scoring. Moreover, the combination of GMM-RBM vectors and i-vectors at the score level improves the performance more.

The rest of the chapter is organized as follows. Section 4.1 describes the proposed GMM-RBM vectors. Section 4.2 investigate the effect of activation and transformation functions used, respectively, for URBM training and GMM-RBM vector extraction and discusses the database, baseline systems, and the experimental results. Section 4.3 concludes the chapter.

4.1 Proposed GMM-RBM Vectors

Recently, the advances in DL have improved the quality of i-vectors, but the DL techniques in use are computationally expensive and need phonetic labels for the background data. We propose in this section an alternative vector-based representation for speakers in a less computationally expensive manner with no use of any phonetic or speaker labels.

RBMs are good potentials for this purpose because they have good representational powers and they are unsupervised and computationally low cost. In this work, it is assumed that the inputs of RBM, i.e., visible units, are GMM supervectors and the outputs, i.e., hidden units, are the low dimensional vectors we are looking for. The RBM is trained given the background GMM supervectors and will be referred to

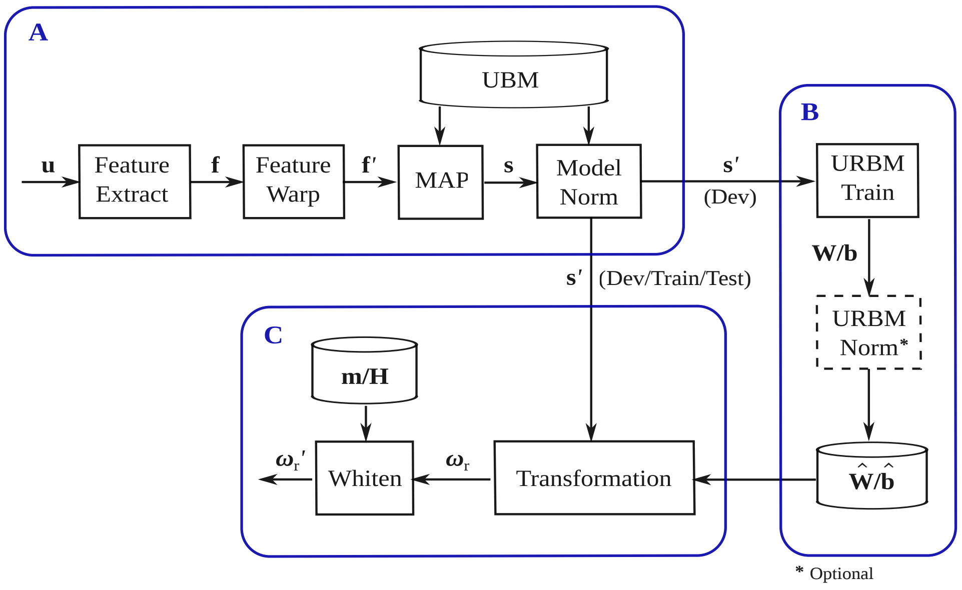


Figure 4.1: Block-diagram of the proposed GMM-RBM vector framework. $\boldsymbol{W}$ and $\boldsymbol{b}$ are the parameters of the Universal RBM (URBM), $\boldsymbol{m}$ is the global mean, and $\boldsymbol{H}$ is the whitening matrix obtained on the background GMM-RBM vectors.

as URBM. The role of the URBM is to learn the total session and speaker variability among the background supervectors. Different types of units and activation functions can be used for training the URBM which will be mentioned in section 4.1.2 and evaluated in section 4.2 for this application. After training the URBM, the visible-hidden connection weight matrix is used to transform unseen GMM supervectors to lower dimensional vectors which will be referred to as GMM-RBM vectors in this work.

Fig. 4.1 shows the block-diagram of the proposed framework. The whole process can be divided in three main stages detailed in the following sections. First, GMM supervectors are built from the warped spectral features given the Universal Background Model (UBM), and then are normalized using the UBM parameters. Second, the background GMM supervectors are used to train the URBM and then it can optionally be normalized to provide an appropriate transformation matrix from the supervectors to the proposed low dimensional vectors. Third, given the unseen GMM supervectors and the parameters of the URBM, GMM-RBM vectors are extracted.

4.1.1 GMM Supervector Preparation

As it is shown in Block A of Fig. 4.1, input speech signals are first characterized by spectral feature vectors. Afterwards, feature warping is applied to map the distribution of each individual feature to a Gaussian distribution over a time interval (see section 2.1.1). It will be shown in the experimental result section that feature warping has a high impact on the performance of the proposed GMM-RBM vectors.

Warped features are then modeled by a GMM adapted from the UBM. The mean vectors of each adapted GMM are stacked to build a supervector (see section 2.1.3). In order to increase the discrimination power, supervectors are model-normalized using the mean supervector and diagonal covariance matrix of the UBM ($\boldsymbol{s}_{ubm}$ and $\boldsymbol{\Sigma}_{ubm}$),

$$\boldsymbol{s}' = \boldsymbol{\Sigma}_{ubm}^{-1/2}(\boldsymbol{s} - \boldsymbol{s}_{ubm}) \tag{4.1}$$

Model normalization helps also having zero mean and unit variance for supervectors which is a prior assumption for the training of an RBM with real-valued inputs as it will be described in the next section.

4.1.2 Universal RBM

Normalized supervectors obtained on the background data are used to train the URBM (Block B of Fig. 4.1). The role of the URBM is to learn all session and speaker variability among background supervectors. The URBM parameters will then be used to transform unseen supervectors to lower dimensional GMM-RBM vectors. Different visible and hidden units, and activation functions can be used for training an RBM (Hinton, 2012). Since the inputs in this application are real-valued supervectors, the visible units will be Gaussian. However, sigmoid and ReLU can be used in the hidden layer during the training of the URBM. As it is mentioned in (Hinton, 2012) and proved by our experiments, training an RBM with both linear hidden and visible units is highly unstable. Therefore, pure linear hidden units are discarded in this work. Given the URBM parameters, any reasonable transformation function could be used to transform unseen supervectors. In this section, the problem

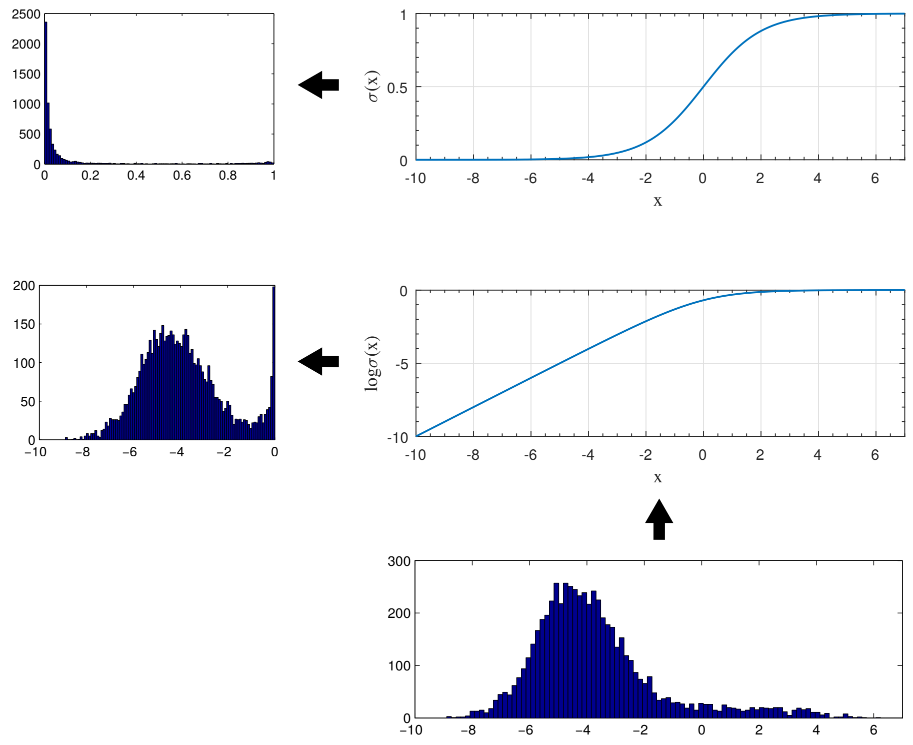


Figure 4.2: Histograms of the first hidden unit values before (bottom) and after (left) transformation with *sigmoid* and log *sigmoid* functions. URBM is trained with *sigmoid* hidden units.

of the use of the traditional sigmoid function for both activation and transformation is first addressed and a potential solution is proposed. Then in the next section a variant of ReLU, which will be referred to as VReLU, is proposed for this application. It will be shown in section 4.2 that the proposed VReLU does not suffer from the problems of sigmoid and ReLU.

Figure 4.2 shows the histograms of the posterior probabilities of the first hidden unit of the URBM before and after nonlinear transformations. The URBM is trained with traditional sigmoid activation function. The typical sigmoid function and the log sigmoid function are employed for the transformation. Other hidden units show also similar behaviors. As it can be seen in this figure, the posterior probability distribution of hidden units after *sigmoid* transformation will be compressed around zero and far from a Gaussian distribution which is ideal for the proposed GMM-RBM vectors. This fact degrades the performance significantly. The behavior is better in case of log sigmoid function since the most part of the distribution is transformed with linear part of the function, but still there is the same problem for

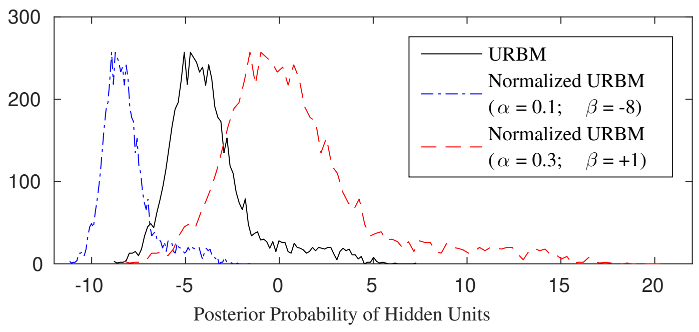


Figure 4.3: The histograms of the posterior probabilities of the first hidden unit of URBM and normalized URBM (with two different pairs of α and β) before nonlinear transformation. The histograms are obtained on the background dataset used for development.

values around zero after transformation. Although the whitening transformation on posterior probabilities afterwards corrects the distributions to some extent, still the performance will be low specifically for sigmoid transformation.

A potential solution can be changing the mean and variance of the posterior probability distributions, before transformation, somehow they fall in the active nonlinear parts of the transformation functions. This can be easily performed through URBM parameter normalization which we have proposed as follows,

$$\hat{\boldsymbol{W}} = \alpha \frac{\boldsymbol{W}}{\max\limits_{i,j} |w_{ij}|} \tag{4.2}$$

$$\hat{b_i} = \beta + \left(b_i - \bar{b}\right) \tag{4.3}$$

where $\boldsymbol{W}$ is the visible-hidden connection weights, $\boldsymbol{b}$ is the vector of hidden bias terms, α and β are two parameters to control, respectively, the variance and the mean of the posterior probability distributions of hidden units before nonlinear transformation, w_{ij} is the (i, j) element of $\boldsymbol{W}$, and b_i and $\bar{b}$ are the i^{th} element and the mean value of $\boldsymbol{b}$, respectively.

Fig. 4.3 shows how changing α and β can move the distribution of the posterior probabilities of hidden units to a desired interval. We will show in Section 4.2 that this movement will improve the quality of the GMM-RBM vectors when the URBM is trained with sigmoid hidden units.

4.1.3 Variable ReLU for URBM Training

Another alternative unit is ReLU. ReLU is a kind of linear unit for which the negative values are zeroed out. If the URBM is trained with ReLU and the inputs are transformed with linear function after training, none of the above problems will occur. However, as we will show in section 4.2, the problem will be that the distribution of posterior probabilities of hidden units will be asymmetric around the mean value, which is not appropriate for PLDA scoring. Therefore, we have proposed in this work a variant of ReLU, which is referred to as VReLU. In VReLU, the unit values less than the threshold τ are zeroed out, rather than the fixed threshold zero in ReLU. Threshold τ is randomly selected from a normal distribution $N(0,1)$ for each hidden unit and for each input sample in each training iteration. In fact, VReLU is defined as follows,

$$f(x) = \begin{cases} x & x > \tau \\ 0 & x \leq \tau \end{cases} \quad , \quad \tau \in N(0,1) \tag{4.4}$$

Figure 4.4 compares ReLU and VReLU with both positive and negative values of τ. It will be shown in section 4.2 that VReLU solves the asymmetric problem of the posterior probability distributions to a great extent and, therefore, it works better than ReLU when PLDA scoring is used.

The full training algorithm for RBM with sigmoid hidden units was given in section 2.2.2. In the following, we only explain the RBM training algorithm with the proposed VReLU. Figure 4.5 shows the training steps based on the CD$_1$ algorithm (Hinton et al., 2006; Hinton, 2012). The connection weights $\boldsymbol{W}$ are first randomly initialized from $N(0, 0.01)$ and the visible and hidden bias terms ($\boldsymbol{a}$ and $\boldsymbol{b}$, respectively) are set to zero. Given the normalized supervectors $\boldsymbol{s}'$, the posterior probability of the lower dimensional hidden vector $\boldsymbol{h}$ is calculated using eq. 4.4. Afterwards, supervectors are reconstructed given the hidden unit values. Then the reconstructed supervectors $\boldsymbol{s}'_r$ are used to recalculate the posterior probabilities of hidden units. These three steps, marked in Fig. 4.5, provide enough information to update the parameters of the network.

The training process is summarized as follows,

- Initialize Network Parameters $(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{a})$

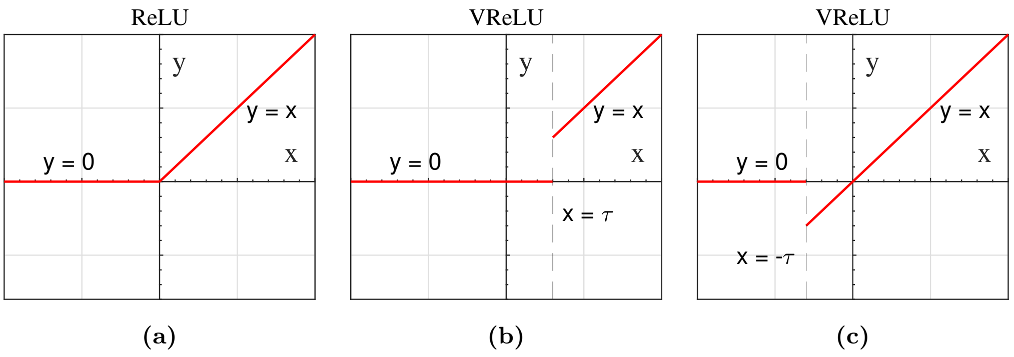


Figure 4.4: Comparison of ReLU and proposed VReLU. In each epoch, per each hidden unit and per each input sample, τ is randomly selected from a normal distribution with zero mean and unit variance. (b) and (c) show the two examples of VReLU when τ is positive and negative, respectively.

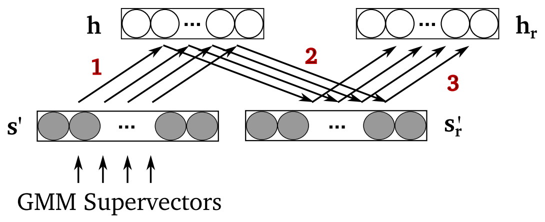


Figure 4.5: Training of the Universal RBM (URBM) given background GMM supervectors.

- CD$_1$ Steps

1. $\boldsymbol{h} = f\left(\boldsymbol{b} + \boldsymbol{W}\boldsymbol{s}'\right)$ (4.5)
2. $\boldsymbol{s}'_r = \boldsymbol{a} + \boldsymbol{W}^t\boldsymbol{h}$ (4.6)
3. $\boldsymbol{h}_r = f\left(\boldsymbol{b} + \boldsymbol{W}\boldsymbol{s}'_r\right)$ (4.7)

- Update Network Parameters

1. $\Delta\boldsymbol{W} = \eta \times \left(\boldsymbol{s}'\boldsymbol{h}^t - \boldsymbol{s}'_r\boldsymbol{h}^t_r\right)^t$ (4.8)
2. $\Delta\boldsymbol{a} = \eta \times \left(\boldsymbol{s}' - \boldsymbol{s}'_r\right)$ (4.9)
3. $\Delta\boldsymbol{b} = \eta \times \left(\boldsymbol{h} - \boldsymbol{h}_r\right)$ (4.10)

where η is the learning rate and $f(.)$ is the VReLU function calculated as in eq. 4.4.

Additionally, a momentum factor is used to smooth out the updates, and the weight decay regularization is used to penalize large weights. The parameters are updated after processing each minibatch and the updating procedure is repeated when all the minibatches are processed.

4.1.4 GMM-RBM Vector Extraction

Given the GMM supervectors from Block A and the URBM parameters from Block B of Fig. 4.1, the GMM-RBM vectors are extracted in Block C as follows,

$$\boldsymbol{\omega}_r = \boldsymbol{W}\boldsymbol{\Sigma}_{ubm}^{-1/2}(\boldsymbol{s} - \boldsymbol{s}_{ubm}) \tag{4.11}$$

As a linear transformation function is used, the hidden unit bias terms $\boldsymbol{b}$ can be easily discarded and only the visible-hidden connection weights $\boldsymbol{W}$ are used for transformation. If we reformulate eq. 4.11 based on zeroth and first order Baum-Welch statistics (see section 2.1.2, eqs. 2.5 and 2.6), we will have,

$$\boldsymbol{\omega}_r = \boldsymbol{W}\boldsymbol{\Sigma}_{ubm}^{-1/2}\mathcal{N}^{-1}(\boldsymbol{u})\tilde{\mathcal{F}}(\boldsymbol{u}) \tag{4.12}$$

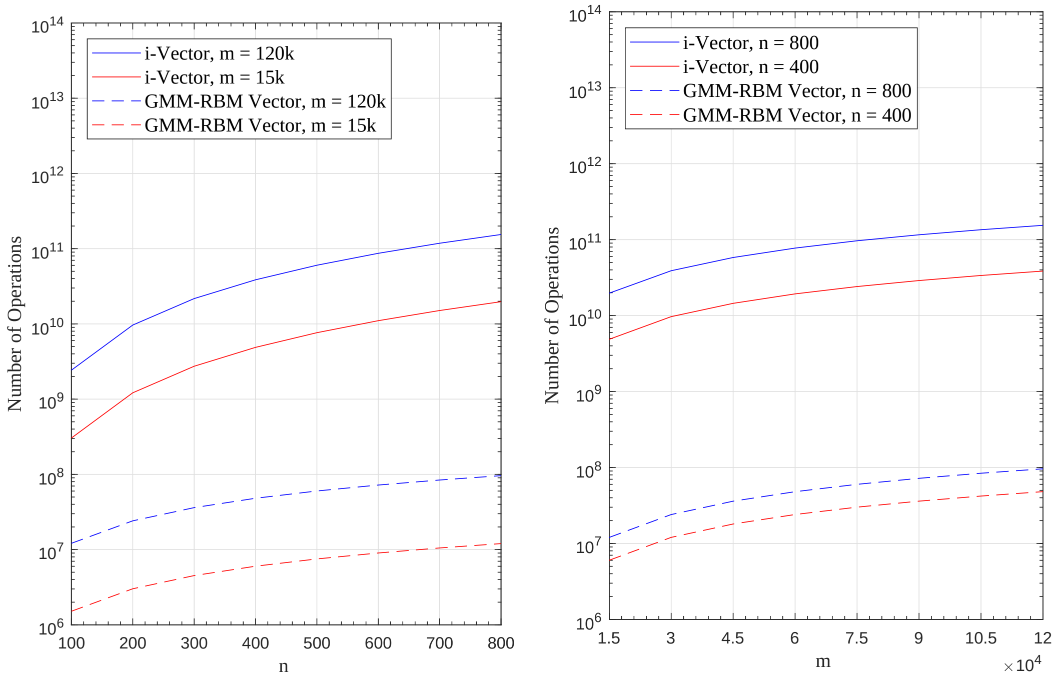
where the relevance factor in map adaptation can also be added to $\mathcal{N}(\boldsymbol{u})$.

Like in case of i-vectors, resulting GMM-RBM vectors are mean normalized and whitened using the mean vector $\boldsymbol{m}$ and the whitening matrix $\boldsymbol{H}$ obtained on the background data as in eq. 2.25.

4.1.5 Computational Load Compared to i-Vector

The comparison of equations 2.13 and 4.12 implies clearly that GMM-RBM vector extraction needs much less computational load. We compare the computational load in terms of the number of product operations required for extracting an i-vector and a GMM-RBM vector with the same size based on equations 2.13 and 4.12. Considering the computational cost for multiplication of two matrices $n \times m$ and $m \times k$ of order $\mathcal{O}(nmk)$ and for a matrix inversion of size $n \times n$ of order $\mathcal{O}(n^3)$, and this fact that $\mathcal{N}(u)$ is diagonal and $\boldsymbol{W}\boldsymbol{\Sigma}_{ubm}^{-1/2}$ in eq. 4.12 or $\boldsymbol{T}^t\boldsymbol{\Sigma}^{-1}$ in eq. 2.13 are computed offline, the minimum computational load of i-vector and GMM-RBM vector extraction will be $\mathcal{O}(n^3 + (2n^2 + 2n)m)$ and $\mathcal{O}((n + 1)m)$, respectively, in which n is the dimension of i-vector/GMM-RBM vector and m is the size of supervector.

Figure 4.6 compares the minimum computational load for extracting an i-vector and a GMM-RBM vector for different values of n and m. The figure implies that the number of the product operations required for extracting a GMM-RBM vector is about $10^{-6} - 10^{-8}$ compared to an i-vector which requires about $10^{-8} - 10^{-11}$



(a) **(b)**

Figure 4.6: Comparison of the number of product operations required for extracting an i-vector and a GMM-RBM vector in terms of (a) the size of i-vector/GMM-RBM vector n and (b) the size of supervector m.

operations. The computational load is of higher importance for online applications in which the frequency of vector extraction is high.

4.2 Experimental Results

The details of the database, the setup of the baseline and the proposed approaches, and the experimental results are given in this section. Baseline systems will be based on conventional i-vectors which are scored using either cosine or PLDA techniques. Proposed GMM-RBM vectors are build according to the block-diagrams of Fig. 4.1. The effect of feature warping, URBM normalization, the type of the activation and transformation functions, as well as the score combination for both cosine and PLDA techniques are shown in this section.

4.2.1 Baseline and Database

Two sets of database are used for the experiments. For development, the core test condition of the NIST 2006 SRE evaluation (NIST, 2006) is used. It includes 816 target models and 51,068 trials. In both the training and testing phases, the duration of speech in signals is approximately two minutes. The background data includes 6,063 speech files collected from NIST 2004 and 2005 SRE corpora. The same background data is used to train UBM, URBM, PLDA, $\boldsymbol{T}$ and whitening matrices.

For evaluation, the NIST 2010 SRE (NIST, 2010), core test-common condition 5, is used. It contains 2354 target speaker models and 30,373 trials involving normal vocal effort conversational telephone speech in training and test. The background data is collected form NIST SRE 2004-2008 and includes 37,600 speech utterances from which 18,140 signals are labeled for PLDA training.

Frequency Filtering (FF) features (Nadeu et al., 2001) (see section 2.1.1) are used in the experiments. Features are extracted every 10 msec using a 30 msec Hamming window. The number of static FF features is 16 and along with delta FF and delta log energy, 33-dimensional feature vectors are built. Before feature extraction, speech signals are subject to an energy-based silence removal process. After feature extraction, a 3-second sliding window is used for feature warping.

ALIZE open source software (Larcher et al., 2013) is used to build the i-vector baseline systems in which cosine and PLDA scoring techniques are employed. The dimension of i-vectors is 400 and PLDA size for development data is 250 and for evaluation 400. A gender-independent UBM is represented as a diagonal-covariance 512-component GMM.

In the proposed GMM-RBM vector framework, GMMs are adapted from the UBM by a relevance factor of 16. Only mean vectors are adapted. The dimension of supervectors is, therefore, $512 \times 33 = 16{,}896$. Two URBMs with the hidden layer sizes of 400 and 8000 are trained to create GMM-RBM vectors. The bigger one is trained only with sigmoid activation function and is used just for comparing the results with those reported in our prior work. URBMs with hidden layer size of 400 are trained with sigmoid, ReLU, and the proposed VReLU. The learning rate, the number of epochs, the minibatch size, the weight decay, and the momentum for the URBM, trained with VReLU, are set to 0.0014, 40, 50, 2×10^{-3}, and 0.9, respectively.

Table 4.1: The effect of feature warping and whitening of input GMM supervectors in the proposed GMM-RBM framework. The numbers in the parentheses indicate the dimensions of GMM-RBM vectors. Results are obtained on the **development** database with cosine scoring.

		Raw Features		Warped Features	
Input to RBM	Output of RBM	EER	minDCF	EER	minDCF
Whitened Supervectors	GMM-RBM Vector (8000)	7.58	0.0346	6.90	0.0331
Raw Supervectors	GMM-RBM Vector (8000)	7.92	0.0379	6.89	0.0323
Raw Supervectors	GMM-RBM Vector (400)	10.45	0.0475	8.08	0.0383

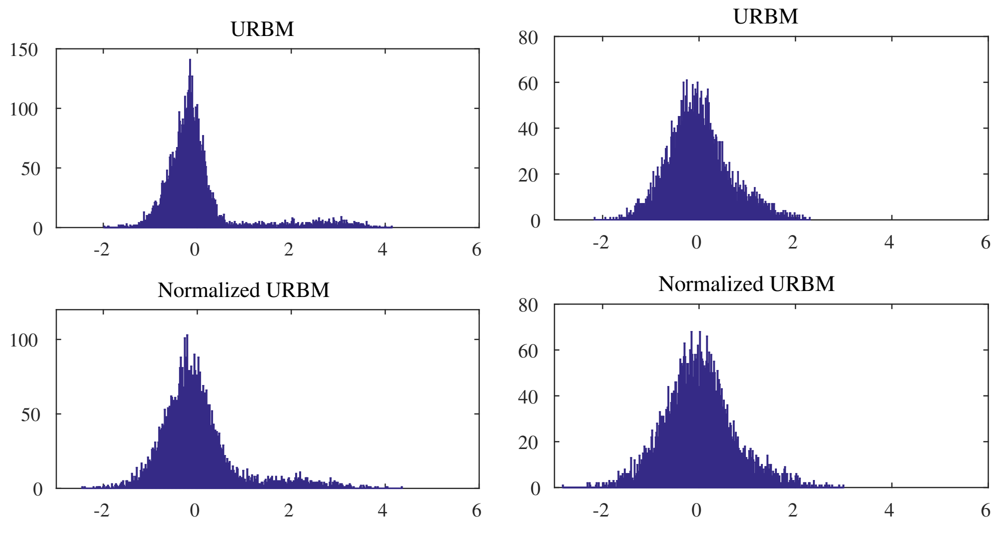
Performance is evaluated using EER and minDCF calculated with $\alpha_1 = 0.1, \alpha_2 = 0.99$ (eq. 2.39) for the development experiments (NIST, 2006) and $\alpha_1 = 0.001, \alpha_2 = 0.999$ for the evaluation experiments (NIST, 2010).

4.2.2 Results

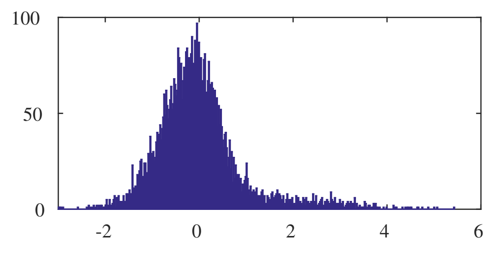
In addition to feature warping, it is also possible to normalize the supervectors through whitening before feeding them to the network. The results reported in Table 4.1 imply that when no feature warping is used, whitening in the supervector level helps. However, if feature vectors are warped, the whitening of supervectors is not effective anymore. Moreover, it is time and memory consuming. The best results are obtained when only feature warping is used.

Figure 4.7 shows the histograms of the first component of the GMM-RBM vectors obtained with a URBM, which is trained with sigmoid activation function. However, sigmoid, log sigmoid, and linear transformation functions are used for vector extraction. The histograms of other components show similar behaviors. For sigmoid and log sigmoid transformations, the histograms are presented for both URBM and normalized URBM parameters. The normalization parameters α and β in Eqs. 4.2 and 4.3 are set to 0.05 and -0.5, respectively. This is to move approximately the posterior distributions into the interval -2 and 2 (Fig. 4.3) corresponding to the active nonlinear parts of the sigmoid and log sigmoid transformation functions. For both sigmoid and log sigmoid, URBM normalization helps having more Gaussian-like histograms. As it will be shown later, this will increase the performance of GMM-RBM vectors when URBM is trained with sigmoid hidden units.

Figure 4.8 shows the same histograms for GMM-RBM vectors for which URBM

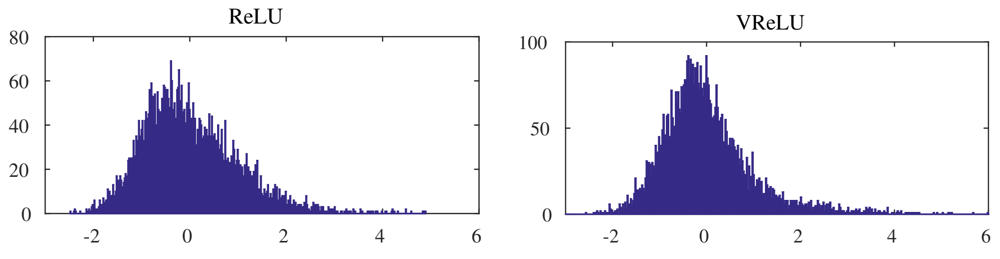


(a) (b)



(c)

Figure 4.7: Comparison of the histograms of the first component of the background GMM-RBM vectors obtained with *sigmoid* activation function and transformation functions of (a) *sigmoid*, (b) log *sigmoid*, and (c) linear.



(a) (b)

Figure 4.8: Comparison of the histograms of the first component of the background GMM-RBM vectors obtained with (a) ReLU and (b) the proposed VReLU activation functions and linear transformation function.

Table 4.2: The effect of the hidden unit types during the training of URBM and the transformation function for GMM-RBM vector extraction. Results are obtained on the **development** database with vectors of dimension 400. VReLU refers to the proposed Variable ReLU.

		cosine		PLDA	
Hidden Units	Transformation	EER (%)	minDCF	EER (%)	minDCF
sigmoid	sigmoid	13.55	0.0570	11.05	0.0517
	sigmoid (Normalized URBM)	8.67	0.0407	6.08	0.0338
	log sigmoid	8.08	0.0383	6.51	0.0316
	log sigmoid (Normalized URBM)	7.85	0.0366	6.28	0.0317
	linear	8.24	0.0382	5.86	0.0317
ReLU	linear	7.82	0.0372	5.58	0.0305
VReLU	linear	7.82	0.0373	5.52	0.0297

is trained with ReLU and VReLU and linear transformation is used in both cases. The figure implies that the histograms are asymmetric in case of ReLU which is due to the training process in which the hidden units are encouraged having positive values. On the other hand, the random threshold τ proposed in VReLU makes it possible having both positive and negative hidden values during the training process. This improves the histograms as shown in Fig. 4.8.

Table 4.2 compares the performance of GMM-RBM vectors extracted by different URBMs and transformation functions. The comparison is based on both cosine and PLDA scoring. As it was expected, the worse results are for sigmoid hidden units and transformation function. The URBM normalization improves significantly the performance of these vectors. The use of log sigmoid itself performs better than sigmoid as discussed for Figs. 4.2 and 4.7. URBM normalization improves also the performance in this case but the amount of improvement is not as much as for sigmoid transformation. If the URBM is trained with sigmoid hidden units and then the parameters are used for linear transformation of input supervectors, the performance will be worse than log sigmoid with cosine scoring but better with PLDA scoring. The use of ReLU for training the URBM and linear function for transformation, keeps the performance as good as log sigmoid with cosine scoring and improves the PLDA results obtained with sigmoid URBM and linear transformation. URBM trained with VReLU improves the PLDA results slightly more. We will show later that VReLU works better than ReLU on unseen evaluation set with both cosine and PLDA scoring.

Table 4.3: Performance comparison of proposed GMM-RBM vectors and conventional i-vectors on the **evaluation** set core test condition-common 5 of NIST SRE 2010. GMM-RBM vectors and i-vectors are of a same size of 400.

	cosine		PLDA	
	EER (%)	minDCF	EER (%)	minDCF
[1] i-Vector	6.270	0.05450	4.096	0.04993
[2] GMM-RBM Vector (Trained with ReLU)	6.638	0.06228	4.517	0.05085
[3] GMM-RBM Vector (Trained with VReLU)	6.497	0.06099	3.907	0.05184
Fusion [1] & [3]	**5.791**	**0.05238**	**3.814**	**0.04673**

Table 4.3 compares the performance of GMM-RBM vectors, which are obtained with URBMs trained with ReLU and VReLU, with traditional i-vectors on the evaluation set. The use of proposed VReLU shows better performance than the use of ReLU in both cosine and PLDA scoring. This fact implies that the variable threshold τ in VReLU has increased the generalization power of URBM in addition to the correction of the histograms. As in this table, the performance of the best GMM-RBM vectors is comparable to that of i-vectors for both cosine and PLDA scoring. This is a significant achievement since the computational load of GMM-RBM vector extraction is much less than the traditional i-vector extraction as discussed in section 4.1.5. At the end, the best results are achieved with score fusion of i-vectors and GMM-RBM vectors which shows about 7-7.5% and 4-6.5% relative improvements in terms of EER and minDCF, respectively, compared to i-vectors. For score fusion, BOSARIS toolkit (Brummer & Villiers, 2011) is used. The fusion weights are trained on the development set.

4.3 Conclusion

We have presented in this work a new vector representation of speech for text-independent speaker recognition. GMM supervectors have been transformed by the proposed Universal RBM (URBM) to lower dimensional vectors, referred to as GMM-RBM vectors. The role of URBM has been to learn the total speaker and session variability among background GMM supervectors. The use of different hidden units for the training of URBM and different transformation functions for the vector extraction are investigated. A variant of Linear Rectified Units (ReLU), which is referred to as variable ReLU (VReLU), is proposed. The variable threshold defined in these units corrects the histograms of GMM-RBM vectors and leads to higher generalization power of URBM. The experimental results on the core test-common condition 5 of NIST 2010 SRE show that the performance of GMM-RBM vectors is comparable with that of traditional i-vectors with both cosine and PLDA scoring but with much less computational load. Moreover, the best results are obtained by score fusion of GMM-RBM vectors and i-vectors.

Chapter 5

Deep Learning Backend for i-Vector Language Identification

Language Identification (LID) is the automatic process of identifying a language spoken in a speech utterance. LID systems use typically one of these two levels of information: acoustic-phonetic or phonotactic (Li et al., 2013; Ambikairajah et al., 2011). The acoustic-phonetic level statistically represents the characteristic phonemes of each language by a set of acoustic parameters, while the lexical-phonological rules of each language are taken into account in the phonotactic level to connect phonemes and form words.

Recent successful techniques in both acoustic-phonetic and phonotactic levels are typically based on i-vectors (Ferrer et al., 2016b; McCree & Garcia-Romero, 2015). As it was mentioned in the previous chapters, an i-vector is a compact representation of characteristics of a speech signal, which has been originally developed for speaker recognition (Dehak, Kenny, et al., 2011) and has also shown promising performance for LID (e.g., Dehak, Torres-Carrasquillo, Reynolds, & Dehak, 2011; González Martínez, Plchot, Burget, Glembek, & Matějka, 2011). Some post-processing techniques are usually required to compensate undesired session variability in the i-vector space. Linear Discriminant Analysis (LDA), Within-Class Covariance Normalization (WCCN), and Probabilistic Linear Discriminant Analysis (PLDA) (see section 2.1.4) are the most commonly used techniques in speaker recognition (Dehak, Kenny, et al., 2011; Prince & Elder, 2007; Kenny, 2010). However, some of these techniques may not be such effective for LID due to the limited number of language classes (González Martínez et al., 2011; Singer et al., 2011).

Like in speaker recognition, Deep Neural Networks (DNNs) have been used in the i-vector extraction process (e.g., Song, Hong, et al., 2015; Richardson et al., 2015b) or applied after i-vector computation as classifiers (Matějka et al., 2012; Matejka et al., 2014; Ferrer et al., 2016b). In (Matejka et al., 2014; Song, Cui, et al., 2015; Fer, Matejka, Grezl, Plchot, & Cemocky, 2015; Song, Hong, et al., 2015), DNN bottleneck features are used in the conventional i-vector extraction process, and in (Richardson et al., 2015b; Ferrer et al., 2016b), in addition to bottleneck features, DNNs are employed for acoustic modeling to extract Baum-Welch statistics. The highest gains are reported when DNN bottleneck features are used with conventional Universal Background Model (UBM) for i-vector extraction (Richardson et al., 2015b; Ferrer et al., 2016b).

In this chapter, we will focus on the application of the LID technology in intelligent vehicles. In this scenario, LID systems are evaluated using words or short sentences recorded in cars in four languages: English, Spanish, German, and Finnish. As the use of DNNs in the i-vector extraction process is computationally expensive for both acoustic modeling and bottleneck feature extraction, we will use the conventional i-vectors in this scenario, in which the computational time is important. Instead, we will explore the use of DNNs only for i-vector language classification. As opposed to (Matějka et al., 2012; Matejka et al., 2014; Ferrer et al., 2016b) in which neural networks with only one hidden layer are used for this purpose, we will explore DNNs with different architectures. Additionally, both raw i-vectors and channel-compensated i-vectors are considered as inputs to DNNs. In order to have the highest accuracy with the minimum response time of the system, signals with different duration from less than 2 sec to higher than 3 sec with the average duration of 3.8 sec are analyzed. The performances of the proposed DNN architectures are compared with both frame-based GMM-UBM and i-vector baseline systems.

The rest of the chapter is organized as follows. Section 5.1 describes the proposed DNN backend architecture for language i-vector classification. Section 5.2 explains the database and the scenario used for this particular task. Section 5.3 reports on the setup and details of the training of the proposed DNNs as well as the baseline systems used in this work. Section 5.4 discusses the experimental results. Section 5.5

concludes the chapter.

5.1 Proposed DNN Architecture

The successful use of DNNs for discriminating between target and impostor i-vectors in speaker verification (Chapter 3), motivated us to make use of DNNs for the LID multi-classification task as well. As few i-vectors are available for each target class in speaker recognition and, therefore, the amount of target and impostor i-vectors are highly unbalanced, DNNs need some tricks for training to be efficient. In this application, however, we do not have this problem.

Figure 5.1 shows the architecture of DNNs we have proposed in this work. The inputs are i-vectors and the outputs are the language class posteriors. The softmax and sigmoid are used as the activation functions of the internal and the output layers, respectively. In order to Gaussianize the output posterior distributions, we have proposed to compute the output scores in Log Posterior Ratio (LPR) forms as,

$$\Lambda(\mathcal{C}_i|\boldsymbol{\omega}) = \log P(\mathcal{C}_i|\boldsymbol{\omega}) - \log \sum_{j \neq i} P\left(\mathcal{C}_j|\boldsymbol{\omega}\right) \tag{5.1}$$

where $P(\mathcal{C}_i|\boldsymbol{\omega})$ is the posterior probability of ith language class $\mathcal{C}_i$ given the test i-vector $\boldsymbol{\omega}$. As the sum of the output posterior probabilities equals to 1 in softmax, eq. 5.1 can be re-written as,

$$\Lambda(\mathcal{C}_i|\boldsymbol{\omega}) = \log P(\mathcal{C}_i|\boldsymbol{\omega}) - \log\left(1 - P(\mathcal{C}_i|\boldsymbol{\omega})\right) \tag{5.2}$$

The Gaussian distribution of the output scores is important for being compatible with other LID systems for score fusion.

As the response time of the LID system is important in the car, the computational complexity of the classifier should also be taken into account. Therefore, we have proposed to choose the size of the first hidden layer as the lowest power of 2 greater than the input layer size. From the second hidden layer towards the output, the size of each layer will be half of the previous layer. For example, the configuration of a 3-hidden-layer DNN will be as 400-512-256-128-4, where 400 is the size of the input i-vectors and 4 is the number of language classes. It will be shown in section 5.4 that, in this way, we can decrease the computational complexity to a great extent while keeping the classification accuracy.

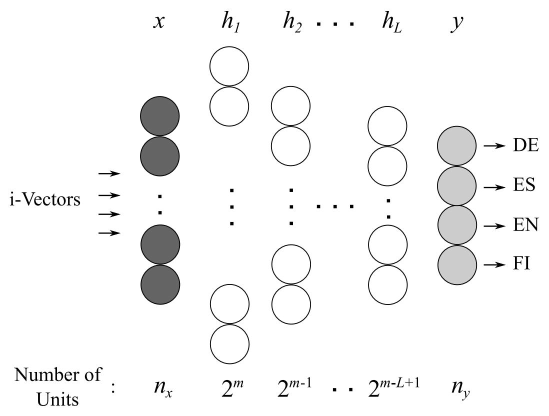


Figure 5.1: Proposed DNN architecture used for i-vector language identification (L denotes the number of hidden layers and $m = \lceil \log_2^{n_x} \rceil$).

As opposed to the speaker recognition (Chapter 3) where the pre-training step was helpful, neither Restricted Boltzmann Machine (RBM) nor discriminative pre-training have been effective for this task. This is not only for the the proposed architecture (Fig. 5.1), but also for other DNNs with hidden layers of the same size in our experiments. Therefore, no pre-training will be employed in this application.

Two forms of i-vectors are considered as inputs to DNNs, raw i-vectors and session-compensated i-vectors. LDA and WCCN are two commonly used techniques for session variability compensation among i-vectors. Although LDA performs better than WCCN for the LID application when cosine scoring is used, we will use only WCCN session-compensated i-vectors as the inputs to DNNs. This is because the number of the language classes is very few in this application and, therefore, the maximum number of meaningful eigenvectors will be also few (number of classes minus one). We implemented different DNN architectures with LDA-projected i-vectors as inputs but no gain was observed. The use of raw i-vectors is advantageous as no language-labeled background data is required.

5.2 Database and Scenario

The application is focused on the LID technology in vehicles where the response time of the system is crucial for user acceptance. Four languages have been chosen for the experiments: English, Finnish, German, and Spanish. The database has been recorded within the scope of the EU project SpeechDat-Car (LE4-8334) (Moreno et al., 2000). The database comprises utterances from 300 speakers recorded in 600 different sessions. Half of the speakers have been male and half female. They are equally distributed in three groups of ages between 18 and 65 years old and for each language, the speakers are chosen from five dialectal regions.

Four high quality audio channels have been recorded simultaneously. For this work, the close-talk microphone is selected and the sampling frequency is 16 kHz .

Several recording conditions were defined: car stopped by motor running, car in town traffic, car moving at a low speed with rough road conditions, and car moving at a high speed with good road conditions. For the experiments, the signals were taken from all these conditions with windows closed, roof window closed and radio off.

For each language, three sets of speakers were defined by selecting randomly from gender, age group and recording conditions: 150 speakers in the training set, 75 speakers in the development set and 75 speakers in the test set.

Three kinds of data have been selected: spontaneous sentences spoken as answers to specific questions, phonetically rich sentences which are read sentences from a phonetically balanced corpus, and phonetically rich words which are a set of words used to enrich the phonetic balance of the corpus.

Table 5.1 shows the number of utterances and the total signal duration in hours for each set. The training and development data are around 11 hours for Finnish and German and 6 hours for English and Spanish. Figure 5.2 shows the histograms of the duration of the utterances. It can be observed that there is a maximum around the first second, corresponding to the phonetically rich words. There is another maximum in the third second, related to phonetically rich sentences. Finally, the tail of the histogram is due to the longer spontaneous utterances.

5.3 Setup and Baseline

Speech signals are pre-emphasized with $\rho = 0.97$. Mel-Frequency Cepstral Coefficient (MFCC) features are extracted every 10 msec using a 25 msec Hamming window. Each feature vector consists of 8 MFCC coefficients obtained from a Mel filter bank of 24 filters. Before feature extraction, speech signals are subject to an energy-based

Table 5.1: Number of utterances and total signal duration for train, test and development sets.

Language	**Train**	**Development**	**Test**
German	6,954 (7h18m)	3,491 (3h43m)	3,490 (3h44m)
Spanish	4,755 (3h58m)	2,302 (1h56m)	2,188 (1h46m)
English	4,781 (3h57m)	2,393 (1h58m)	2,324 (1h53m)
Finnish	4,884 (7h18m)	2,489 (3h44m)	2,416 (3h32m)
Total	21,374 (22h32m)	10,675 (11h22m)	10,418 (10h55m)

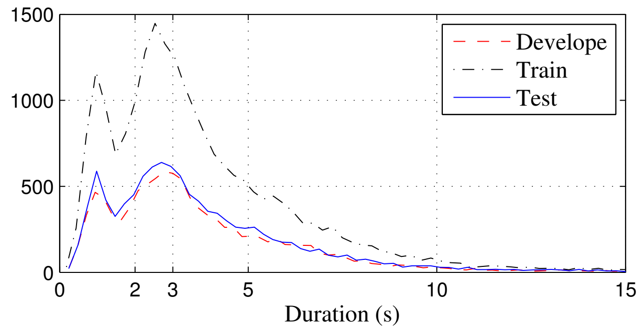


Figure 5.2: Histogram of signal duration.

silence removal process. Shifted Delta Cepstrum (SDC) coefficients are then created by a 8-1-3-5 configuration, spanning a duration of roughly 170 msec.

The size of i-vectors is set to 400 and $\boldsymbol{T}$ matrix is trained with 20 iterations using the same development dataset we have used for training UBM. The gender-independent UBM is represented as a diagonal covariance, 512-component GMM. The i-vector extraction process is carried out using the ALIZE open source software (Larcher et al., 2013) with the minimum divergence training algorithm. Since the speech signals are short in this application, both frame-based GMM-UBM and i-vector systems are used as the baselines in this work. The i-vector baseline system is the same as that proposed in (NIST, 2015). All the language i-vectors are centered and whitened. Afterwards, for each language, the average of the training i-vectors is considered as the average-language i-vector. Then the cosine distance between the average-language i-vector and test i-vectors is computed as final scores. If a language-labeled background dataset is available, as in this application, LDA or

WCCN can be also used for session variability compensation before scoring. As the number of the language classes is 4 in this application, the rank of LDA is set to 3.

For DNN experiments, the proposed architecture in section 5.1 is implemented with both raw and WCCN session-compensated input i-vectors. In both cases no normalization is applied on i-vectors prior to feeding to DNNs. The Proposed DNN architectures are trained with the learning rates of 0.07 and 0.04 and the number of epochs of 500 and 200 for raw and WCCN compensated i-vectors, respectively. Momentum and weight decay are set, respectively, to 0.9 and 0.001 for all DNNs.

LID systems are evaluated based on the total language identification error rate (LER) defined as,

$$LER = \frac{1}{N}\sum_{i=1}^{N} P_{miss}(L_i) \tag{5.3}$$

where $P_{miss}(L_i)$ is the probability that an utterance spoken with the target language L_i is misclassified, and N is the total number of languages.

5.4 Experimental Results

Table 5.2 summarizes the results for all the techniques in four categories based on the test signal durations: less than 2 sec, between 2 and 3 sec, more than 3 sec, and all durations. The first two categories are more interesting because the decision should be made fast in this application. The DNN results are reported based on the proposed architecture of Fig. 5.1 with 3 hidden layers (400-512-256-128-4). The network is trained with training signals of all durations. As it can be seen in this table, among i-vector baseline systems, i-vector + LDA outperforms the two others with a big difference in all categories. Both i-vector+DNN systems show superior performance compared to i-vector + LDA baseline system. However, except for the test signals with longer duration than 3 sec, DNNs with raw i-vectors perform better than with WCCN session-compensated i-vectors. This is an advantage where no language-labeled background data is available, e.g., (NIST, 2015). The frame-based GMM-UBM baseline system works better than other systems only for test signals shorter than 2 sec. However, the accuracy is still high in comparison to other categories.

Furthermore, Table 5.2 implies that the shortest test signals for which all the

Table 5.2: Comparison of LID systems for short signals recorded in car. Performance values are reported based on LER (%).

Duration of Test Signals (in sec)	$t < 2$	$2 \leqslant t < 3$	$t \geqslant 3$	All
Number of Samples	2,472	2,355	5,591	10,418
[1] GMM-UBM	**9.98**	4.56	4.70	6.02
[2] i-Vector + Cosine	17.28	6.58	5.00	8.09
[3] i-Vector + WCCN + Cosine	14.50	5.03	3.42	6.31
[4] i-Vector + LDA + Cosine	12.41	3.96	2.32	5.03
[5] i-Vector + WCCN + DNN	12.06	3.30	**2.30**	4.60
[6] i-Vector + DNN	11.01	**2.87**	2.58	**4.54**
Fusion [6] & [4]	11.63	3.41	**1.95**	4.48
Fusion [6] & [1]	10.20	3.04	2.49	4.41
Fusion [6] & [4] & [1]	11.12	3.37	**1.96**	**4.39**

techniques achieve adequate performance are between 2 and 3 sec. For these test signals, the proposed DNN architecture with raw input i-vectors achieves 37% and 27% relative improvements compared to GMM-UBM and i-vector + LDA baseline systems, respectively. In fact, the combination of i-vectors and the proposed DNN architecture meets the goal of this application, that is high accuracy and fast decision. For test signals longer than 3 sec, i-vector+WCCN+DNN system works the best and for signals with any duration, the i-vector+DNN system outperforms all other individual systems with 25% and 10% relative improvements comparing to GMM-UBM and i-vector+LDA baseline systems, respectively. Finally, the combination of the LID systems in the score level shows that a 16% relative improvement can be achieved for the signals with longer duration than 3 sec when the i-vector + LDA baseline and the i-vector + DNN systems are fused. Additionally, a slightly improvement is observed for the test signals with all durations when the i-vector + DNN system is combined with both GMM-UBM and i-vector + LDA baseline systems. For score fusion, the scores of different systems are simply summed.

Based on the results reported on Table 5.2, we can recommend the following LID systems for test signals of different durations: GMM-UBM for shorter than 2 sec, i-vector + DNN for longer than 2 sec and shorter than 3 sec, fusion of i-vector + LDA and i-Vector + DNN for longer than 3 sec, and fusion of i-vector + LDA, i-vector + DNN, and GMM-UBM for all durations.

Table 5.3 compares the performance and the size of the proposed DNN architec-

Table 5.3: Comparison of the proposed DNN architecture with some other architectures.

DNN Architecture	# Parameters	Duration of Test Signals			
		$t < 2$	$2 \leqslant t < 3$	$t \geqslant 3$	All
400-512-4	202k	11.74	3.51	2.63	4.79
400-512-512-4	458k	11.47	2.96	**2.39**	4.57
400-512-512-512-4	714k	11.11	3.42	2.62	4.74
400-512-256-4	329k	11.28	3.31	2.51	4.69
400-512-256-128-4	361k	**11.01**	**2.87**	2.58	**4.54**

ture with some other DNN architectures. As it can be seen, the proposed architecture with 3 hidden layers achieves the best accuracy in the first two categories. Among these DNN architectures, the 2-hidden-layer DNN with hidden layer size of 512 works slightly better than the proposed architecture for signals longer than 3 sec, but with the cost of bigger size and, consequently, higher computational complexity.

Figure 5.3 compares the proposed i-vector + DNN system with the i-vector + LDA baseline system in terms of Detection Error Tradeoff (DET) curves for test signals of all durations. DET curves are obtained for each language versus all other languages. In other words, each language is considered as the target class and all other languages as the non-target one. Each DET curve shows how well the target and non-target languages are distinguished by the LID system. As it can be seen in this figure, the proposed i-vector + DNN system outperforms the i-vector + LDA baseline system for all languages in all operating points, resulting in a 7-38% relative improvements in terms of Equal Error Rate (EER). The same conclusion can be observed for other test durations in Figs. 5.4, 5.5, and 5.6. In all of these figures, DNN system shows much higher accuracy than the baseline i-vector/LDA system for the Finnish language compared to other languages. This is while the number of training utterances has been the same or even less compared to other languages (Table 5.1). One reason for higher accuracy for Finnish could be this fact that DNN can extract and model higher order statistics underlying in Finnish i-vectors.

5.5 Conclusion

A DNN architecture has been proposed in this chapter as a backend for i-vector Language Identification (LID) of short utterances recorded in cars. The computational

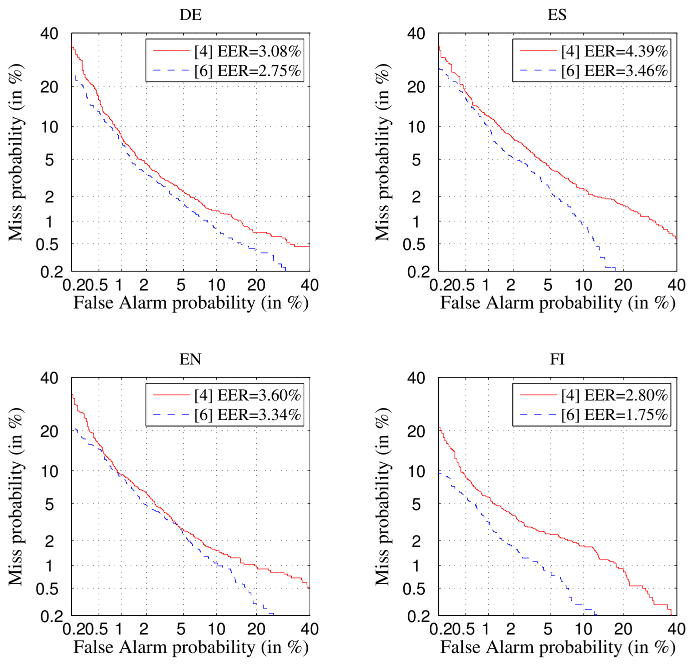


Figure 5.3: DET curve comparison of the proposed DNN system with the i-vector/LDA baseline system for test utterances of all durations.

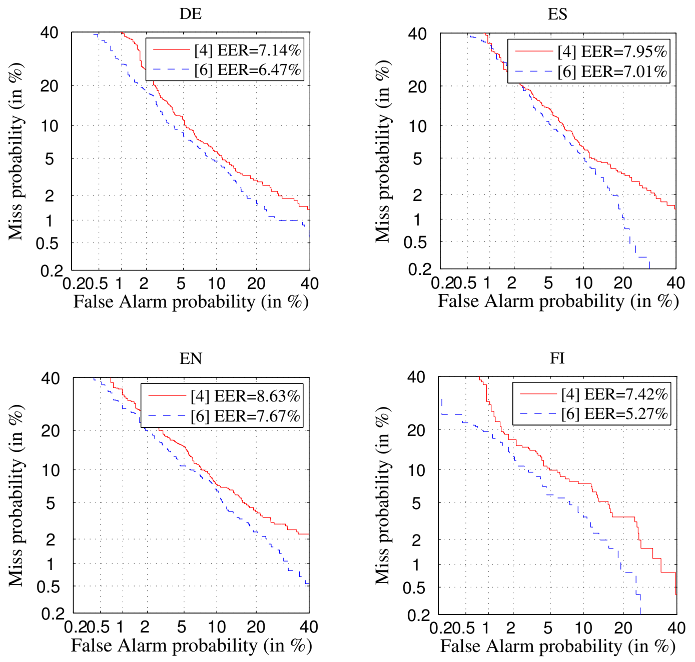


Figure 5.4: DET curve comparison of the proposed DNN system with the i-vector/LDA baseline system for test utterances shorter than 2 sec.

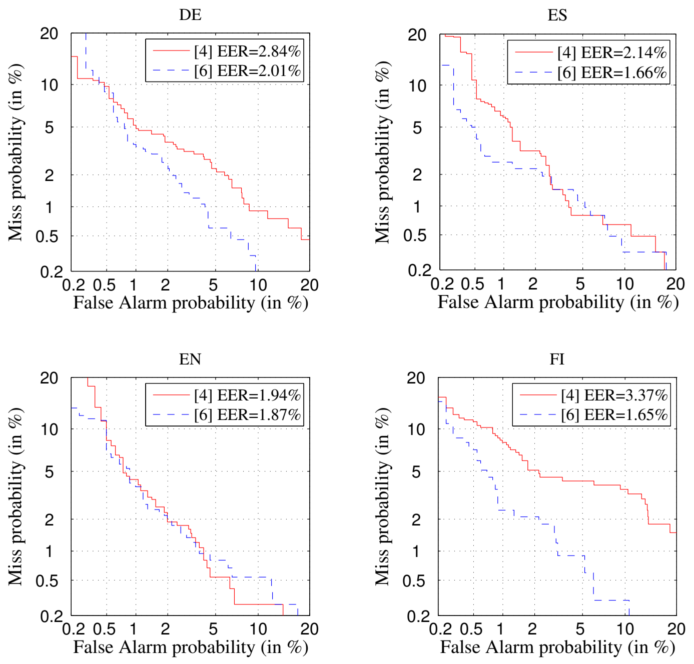


Figure 5.5: DET curve comparison of the proposed DNN system with the i-vector/LDA baseline system for test utterances between 2 and 3 sec.

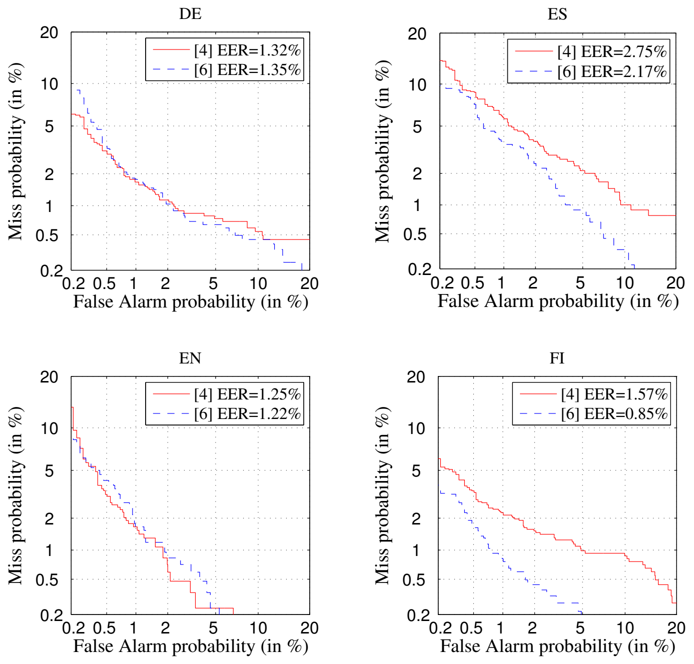


Figure 5.6: DET curve comparison of the proposed DNN system with the i-vector/LDA baseline system for test utterances longer than 3 sec.

complexity and the response time of the LID system have been taken into account for this application. In order to have the highest accuracy with the minimum response time of the system, signals with different durations from less than 2 sec to higher than 3 sec with the average duration of 3.8 sec have been analyzed. Each hidden layer in the proposed DNN architecture has the number of units half of the previous hidden layer, which is closer to the input. This way, the size and computational complexity of the network is decreased to a great extent while the accuracy is preserved. It has been shown that for test signals with durations between 2 and 3 sec the proposed DNN architecture with raw i-vectors as inputs outperforms GMM-UBM and i-vector/LDA baseline systems by 37% and 28%, respectively.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The main contributions of this thesis have been presented in three main works. In the first one, a hybrid architecture based on Deep Belief Network (DBN) and Deep Neural Network (DNN) has been proposed to discriminatively model each target speaker for i-vector speaker verification. The main objective has been to fill the performance gap between the cosine and the oracle Probabilistic Linear Discriminant Analysis (PLDA), estimated with actual labels, scoring systems when no labeled background data is available. Two main contributions have been proposed to make DNN more efficient in this particular task. Firstly, the most informative impostor i-vectors have been selected and clustered to provide a balanced training. Secondly, each DNN has been initialized with the speaker specific parameters adapted from a global model, which has been referred to as Universal DBN (UDBN). In order to have more insight into the behavior of these techniques in both single and multi-session speaker enrollment tasks, the experiments have been carried out in both scenarios. Experiments were performed on National Institute of Standard and Technology (NIST) Speaker Recognition Evaluation (SRE) 2006, mainly for development, and on NIST 2014 i-vector challenge, mainly for evaluation. It was shown that the proposed hybrid system fills approximately 46% of the performance gap between the cosine and the oracle PLDA scoring systems in terms of minimum DCF (minDCF). Although the proposed system still does not outperform the baseline PLDA with estimated labels, their score fusion is highly effective and covers 79% of this gap.

In the second work, we have presented a new vector representation of speech for text-independent speaker recognition. Gaussian Mixture Model (GMM) supervectors have been transformed by a Universal RBM (URBM) to lower dimensional vectors, referred to as GMM-RBM vectors. The role of URBM has been to learn the total speaker and session variability among background GMM supervectors. The use of different hidden units for training of URBM and different transformation functions for vector extraction are investigated. A variant of Rectified Linear Unit (ReLU), which is referred to as Variable ReLU (VReLU), is proposed. The variable threshold defined in these units corrects the histograms of GMM-RBM vectors and leads to higher generalization power of URBM. The experimental results on the core test-common condition 5 of NIST 2010 SRE show that the performance of GMM-RBM vectors is comparable with that of traditional i-vectors with both cosine and PLDA scoring but with much less computational load. Moreover, the best results are obtained by score fusion of GMM-RBM vectors and i-vectors.

In the third work, a DNN architecture has been proposed for i-vector Language Identification (LID) of short utterances recorded in cars. The computational complexity and the response time of the LID system is important in this application. In order to have the highest accuracy with the minimum response time of the system, signals with different duration from less than 2 sec to higher than 3 sec with the average duration of 3.8 sec have been analyzed. It has been shown that for test signals with duration between 2 and 3 sec the proposed DNN architecture with raw i-vectors as inputs outperforms GMM-UBM and i-vector/Linear Discriminant Analysis (LDA) baseline systems by 37% and 28%, respectively.

6.2 Future Research Lines

The main goal in this thesis has been taking advantage of Deep Learning (DL) technology in speaker and language recognition while no speaker or phonetic label is used. Phonetic and speaker labels are expensive and usually are not easily accessible. However, an unlimited research line will opene given labeled background data. Hence, the future work could follow two main lines: unsupervised and supervised (given labeled background data) DL techniques for speaker and language recognition.

The main advantage of unsupervised techniques is that no labeled background data is required. Therefore, a large amount of unlabeled data could be used for the development of the recognition systems. Unlabeled data are usually easily accessible with relatively low cost. All three works presented in Chapters 3, 4, and 5 could be considered in this category and improved in future works. For instance, in Chapter 3, different DNN architectures, like that one proposed for LID in Chapter 5, different adaptation techniques, more effective activation functions, like VReLU proposed in Chapter 4, or more effective impostor selection algorithms could be used. In Chapter 4, other alternative unsupervised networks like auto-encoders or DBN could be tried out. Feature vectors can be used directly as inputs to the network, or the speaker adaptation can be carried out through the network rather than through an external GMM Maximum a Posteriori (MAP) adaptation. In Chapter 5, other DL architectures, activation functions, or different inputs like supervectors or feature vectors could be employed. Moreover, the GMM-RBM vectors proposed in Chapter 4 could also be used for LID.

As discussed in the previous section, it has been supposed that phonetic or speaker labels are not available in this thesis. However, given labeled background data, a wide range of DL techniques could be used. Speaker labels are mainly used for speaker and session variability compensation and phonetic labels for training a more accurate acoustic model. One of the reasons that the proposed system in Chapter 3 still does not outperform the baseline PLDA system (with estimated labels) could be that it does not explicitly compensate the session variability as it is carried out in PLDA. Thus, it is expected that adding some explicit session modeling to the proposed hybrid model could improve the performance. Some possibilities could be the use of speaker classes as the output of the networks. The training process would be more computationally expensive but high probably more effective. Other option could be to compensate the undesired i-vector variability using signal processing techniques like Within-Class Covariance Normalization (WCCN), LDA, or PLDA before giving them to the network. Regarding to the vector representation of speech framework, given the speaker labels, a global DNN or hybrid DBN-DNN could be trained on the background data, rather than only an unsupervised Restricted Boltzmann Machine (RBM). Then the output of each hidden layer should be analyzed and the hidden layer showing more speaker discrimination power could be used for vector representation of speech with a similar framework of Chapter 4. Similar techniques can be used for LID as well.

Publications

Book Chapters

.

Ghahabi, O., Safari, P., and Hernando, J. (2018). Deep learning in speaker recognition. *(in review) Handbook of Deep Learning Applications - Springer*

.

Journal Articles

.

Ghahabi, O. and Hernando, J. (2018). Restricted boltzmann machines for vector representation of speech in speaker recognition. *Computer Speech & Language*, 47:16–29. [Online] Available: http://www.sciencedirect.com/science/article/pii/S0885230816302923.

Ghahabi, O. and Hernando, J. (2017). Deep Learning Backend for Single and Multisession i-Vector Speaker Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(4):807–817. [Online] Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7847321 (Awarded the best RTTH doctorate student article in 2017)

.

Conference Papers

.

Ghahabi, O., Bonafonte, A., Hernando, J., and Moreno, A. (2016). Deep neural networks for i-vector language identification of short utterances in cars. In *Proc. INTERSPEECH*, pages 367–371. [Online] Available: http://www.isca-speech.org/archive/Interspeech_2016/pdfs/1045.PDF.

Ghahabi, O. and Hernando, J. (2015). Restricted boltzmann machine supervectors

for speaker recognition. In *Proc. ICASSP*, pages 4804–4808. [Online] Available: http://ieeexplore.ieee.org/abstract/document/7178883/.

Ghahabi, O. and Hernando, J. (2014). Deep belief networks for i-vector based speaker recognition. In *Proc. ICASSP*, pages 1700–1704. [Online] Available: http://ieeexplore.ieee.org/abstract/document/6853888/ (Awarded the Qualcomm travel grant).

Ghahabi, O. and Hernando, J. (2014). i-vector modeling with deep belief networks for multi-session speaker recognition. In *Proc. Odyssey*, pages 305–310. [Online] Available: http://cs.uef.fi/odyssey2014/program/pdfs/44.pdf.

Ghahabi, O. and Hernando, J. (2014). Global impostor selection for DBNs in multi-session i-vector speaker recognition. In *Advances in Speech and Language Technologies for Iberian Languages*, Lecture Notes in Artificial Intelligence, pages 89–98. [Online] Available: https://link.springer.com/chapter/10.1007/978-3-319-13623-3_10.

Safari, P., **Ghahabi, O.**, and Hernando, J. (2016). From features to speaker vectors by means of restricted boltzmann machine adaptation. In *Proc. Odyssey*, pages 366–371. [Online] Available: http://www.odyssey2016.org/papers/pdfs_stamped/15.pdf.

Safari, P., **Ghahabi, O.**, and Hernando, J. (2016). Speaker recognition by means of restricted boltzmann machine adaptation. In *Proc. URSI*, pages 1–4. [Online] Available: https://upcommons.upc.edu/handle/2117/104488.

Safari, P., **Ghahabi, O.**, and Hernando, J. (2015). Feature classification by means of deep belief networks for speaker recognition. In *Proc. EUSIPCO*, pages 2162–2166. [Online] Available: http://ieeexplore.ieee.org/abstract/document/7362758/.

Raboshchuk, G., Nadeu, C., **Ghahabi, O.**, Solvez, S., Mahamud, B. M., Veciana, A., and Hervas, S. (2014). On the acoustic environment of a neonatal intensive care unit: Initial description, and detection of equipment alarms. In *Proc. INTERSPEECH*, pages 2543–2547. [Online] Available: http://www.isca-speech.org/archive/archive_papers/interspeech_2014/i14_2543.pdf (Awarded the ISCA travel grant).

Bibliography

Ambikairajah, E., Li, H., Wang, L., Yin, B., & Sethu, V. (2011). Language identification: A tutorial. *Circuits and Systems Magazine, IEEE*, *11*(2), 82–108.

Bahari, M. H., McLaren, M., Van Hamme, H., & Van Leeuwen, D. (2012). Age Estimation from Telephone Speech using i-vectors. *Proc. Interspeech*, 506–509.

Barua, S., Islam, M., Yao, X., & Murase, K. (2014). MWMOTE-majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, *26*(2), 405–425.

Bhattacharya, G., Alam, J., & Kenny, P. (2017). Deep speaker embeddings for short-duration speaker verification. *Proc. Interspeech*, 1517–1521.

Brummer, N., & Villiers, E. (2011). BOSARIS toolkit user guide: Theory, algorithms and code for binary classifier score processing.. ([Online]. Available: https://sites.google.com/site/bosaristoolkit/)

Campbell, W., Campbell, J., Reynolds, D., Singer, E., & Torres-Carrasquillo, P. (2006). Support vector machines for speaker and language recognition. *Computer Speech & Language*, 210–229.

Campbell, W., Sturim, D., & Reynolds, D. (2006). Support vector machines using GMM supervectors for speaker verification. *IEEE Signal Processing Letters*, *13*(5), 308–311.

Campbell, W., Sturim, D., Reynolds, D., & Solomonoff, A. (2006). SVM based speaker verification using a GMM supervector kernel and NAP variability compensation. In *Proc. ICASSP* (pp. 97–100).

Campbell, W. M. (2014). Using deep belief networks for vector-based speaker recognition. In *Proc. interspeech* (pp. 676–680).

Dahl, G., Yu, D., Deng, L., & Acero, A. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions* *on Audio, Speech, and Language Processing*, *20*(1), 30–42. doi: 10.1109/TASL.2011.2134090

Dehak, N., & Chollet, G. (2006). Support vector gmms for speaker verification. In *Proc. odyssey* (pp. 1–4).

Dehak, N., Dehak, R., glass, J., Reynolds, D., & Kenny, P. (2010). Cosine similarity scoring without score normalization techniques. In *Proc. odyssey* (pp. 71–75).

Dehak, N., Kenny, P., Dehak, R., Dumouchel, P., & Ouellet, P. (2011, May). Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, *19*(4), 788–798.

Dehak, N., Torres-Carrasquillo, P., Reynolds, D., & Dehak, R. (2011). Language recognition via i-vectors and dimensionality reduction. In *Proc. interspeech* (pp. 857–860).

Deng, L., & Yu, D. (2014). Deep Learning: Methods and Applications. *Foundations and Trends® in Signal Processing*, *7*(3–4), 197–387.

Dumitru, E., Manzagol, P., Bengio, Y., Bengio, S., & Vincent, P. (2009). The Difficulty of Training Deep Architectures and the Effect of Unsupervised Pre-Training. In *Proc. artificial intelligence and statistics* (pp. 153–160).

Erhan, D., Bengio, Y., Courville, A., Manzagol, P., Vincent, P., & Bengio, S. (2010, March). Why Does Unsupervised Pre-training Help Deep Learning? *J. Mach. Learn. Res.*, *11*, 625–660. Retrieved 2015-07-14, from http://dl.acm.org/citation.cfm?id=1756006.1756025

Farrell, K., Mammone, R., & Assaleh, K. (1994). Speaker recognition using neural networks and conventional classifiers. *IEEE Transactions on Speech and Audio Processing*, *2*(1), 194–205.

Fauve, B., Matrouf, D., Scheffer, N., Bonastre, J.-F., & Mason, J. S. D. (2007). State-of-the-art performance in text-independent speaker verification through open-source software. *IEEE Transactions on Audio, Speech, and Language Processing*, *15*(7), 1960–1968. doi: 10.1109/TASL.2007.902877

Fer, R., Matejka, P., Grezl, F., Plchot, O., & Cemocky, J. (2015). Multilingual bottleneck features for language recognition. In *Proc. INTERSPEECH* (pp. 389–393).

Ferrer, L., Lei, Y., McLaren, M., & Scheffer, N. (2014). Spoken language recognition based on senone posteriors. In *Interspeech* (pp. 2150–2154).

Ferrer, L., Lei, Y., McLaren, M., & Scheffer, N. (2016a). Study of senone-based deep neural network approaches for spoken language recognition. *IEEE/ACM*

Transactions on Audio, Speech and Language Processing (TASLP), *24*(1), 105–116.

Ferrer, L., Lei, Y., McLaren, M., & Scheffer, N. (2016b, January). Study of Senone-Based Deep Neural Network Approaches for Spoken Language Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *24*(1), 105–116.

Furui, S. (2004). Fifty years of progress in speech and speaker recognition. *The Journal of the Acoustical Society of America*, *116*(4), 2497–2498.

Garcia-Romero, D., & Espy-Wilson, C. Y. (2011). Analysis of i-vector length normalization in speaker recognition systems. In *Interspeech* (pp. 249–252).

Garcia-Romero, D., Zhang, X., McCree, A., & Povey, D. (2014). Improving speaker recognition performance in the domain adaptation challenge using deep neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)* (pp. 378–383).

Gonzalez-Dominguez, J., Lopez-Moreno, I., Sak, H., Gonzalez-Rodriguez, J., & Moreno, P. J. (2014). Automatic language identification using long short-term memory recurrent neural networks. In *Proc. Interspeech* (pp. 2155–2159).

González Martínez, D., Plchot, O., Burget, L., Glembek, O., & Matějka, P. (2011). Language recognition in ivectors space. In *Interspeech* (pp. 861–864).

Greenberg, C., Banse, D., Doddington, G., Garcia-Romero, D., Godfrey, J., Kinnunen, T., … Reynolds, D. (2014). The NIST 2014 speaker recognition i-vector machine learning challenge. In *Proc. Odyssey* (pp. 224–230).

Hansen, J. H. L., & Hasan, T. (2015, November). Speaker Recognition by Machines and Humans: A tutorial review. *IEEE Signal Processing Magazine*, *32*(6).

Hatch, A., Kajarekar, S. S., & Stolcke, A. (2006). Within-Class Covariance Normalization for SVM-based Speaker Recognition. In *Proc. interspeech* (pp. 1471–1474).

Hatch, A., & Stolcke, A. (2006). Generalized linear kernels for one-versus-all classification: Application to speaker recognition. In *Proc. ICASSP* (pp. 585–588).

He, H., & Garcia, E. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263–1284.

Heigold, G., Moreno, I., Bengio, S., & Shazeer, N. (2016). End-to-end text-dependent speaker verification. In *Proc. ICASSP* (pp. 5115–5119).

Heo, H.-s., Jung, J.-w., Yang, I.-h., Yoon, S.-h., & Yu, H.-j. (2017). Joint training of expanded end-to-end dnn for text-dependent speaker verification. In (pp.

1532–1536).

Hernando, J., & Nadeu, C. (1997). CDHMM speaker recognition by means of frequency filtering of filter-bank energies. In *Proc. eurospeech* (pp. 2363–2366).

Hinton, G. (2012). A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade* (pp. 599–619). Springer Berlin Heidelberg.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., ... Sainath, T. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97.

Hinton, G., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554. doi: 10.1162/neco.2006.18.7.1527

Hinton, G., & Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507. doi: 10.1126/science.1127647

Huang, F. J., Boureau, Y.-L., LeCun, Y., et al. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. CVPR* (pp. 1–8).

Huang, G., Lee, H., & Learned-Miller, E. (2012). Learning hierarchical representations for face verification with convolutional deep belief networks. In *Proc. (CVPR)* (pp. 2518–2525).

Isik, Y., Erdogan, H., & Sarikaya, R. (2015). S-vector: A discriminative representation derived from i-vector for speaker verification. In *Proc. eusipco* (pp. 2097–2101).

Kenny, P. (2006). Joint factor analysis of speaker and session variability: Theory and algorithms. *(Technical Report) CRIM-06/08-13*, 1–17.

Kenny, P. (2010). Bayesian speaker verification with heavy tailed priors. In *Proc. Odyssey)*.

Kenny, P., Gupta, V., Stafylakis, T., Ouellet, P., & Alam, J. (2014). Deep neural networks for extracting baum-welch statistics for speaker recognition. In *Proc. odyssey* (pp. 293–298).

Kenny, P., Ouellet, P., Dehak, N., Gupta, V., & Dumouchel, P. (2008). A study of interspeaker variability in speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, *16*(5), 980–988. doi: 10.1109/TASL.2008.925147

Khoshgoftaar, T., Van Hulse, J., & Napolitano, A. (2010). Supervised neural network modeling: An empirical investigation into learning from imbalanced data with labeling errors. *IEEE Transactions on Neural Networks*, *21*(5), 813–830.

Khoury, E., El Shafey, L., Ferras, M., & Marcel, S. (2014). Hierarchical speaker clustering methods for the nist i-vector challenge. In *Proc. odyssey* (pp. 254–259).

Kinnunen, T., & Li, H. (2010). An Overview of Text-independent Speaker Recognition: From Features to Supervectors. *Speech Commun.*, *52*(1), 12–40.

Kudashev, O., Novoselov, S., Pekhovsky, T., Simonchik, K., & Lavrentyeva, G. (2016). Usage of dnn in speaker recognition: Advantages and problems. In *Advances in neural networks – isnn* (pp. 82–91). Springer International Publishing.

Larcher, A., Bonastre, J.-F., Fauve, B., Lee, K., Lévy, C., Li, H., ... Parfait, J.-Y. (2013). ALIZE 3.0 — open source toolkit for state-of-the-art speaker recognition. In *Proc. interspeech* (pp. 2768–2771).

Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring Strategies for Training Deep Neural Networks. *Journal of Machine Learning Research*, *10*, 1–40.

Lee, H., Largman, Y., Pham, P., & Ng, A. Y. (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks. *Advances in neural information processing systems*, *22*, 1096–1104.

Lee, K., You, C., Li, H., Kinnunen, T., & Zhu, D. (2008). Characterizing speech utterances for speaker verification with sequence kernel SVM. *Computer Speech & Language*, 1397–1400.

Lei, Y., Scheffer, N., Ferre, L., & Mclaren, M. (2014). A novel scheme for speaker recognition using a phonetically-aware deep neural network. In *Proc. ICASSP* (pp. 1714–1718).

Li, H., Ma, B., & Lee, K. (2013). Spoken language recognition: from fundamentals to practice. *Proceedings of the IEEE*, *101*(5), 1136–1159.

Ling, Z., Kang, S., Zen, H., Senior, A., Schuster, M., Qian, X., ... Deng, L. (2015). Deep Learning for Acoustic Modeling in Parametric Speech Generation: A systematic review of existing techniques and future trends. *IEEE Signal Processing Magazine*, *32*(3), 35–52.

Ling, Z.-H., Deng, L., & Yu, D. (2013). Modeling spectral envelopes using restricted boltzmann machines and deep belief networks for statistical parametric speech synthesis. *IEEE Transactions on Audio, Speech, and Language Processing*, *21*(10), 2129–2139.

Liu, Y., Qian, Y., Chen, N., Fu, T., Zhang, Y., & Yu, K. (2015, October). Deep feature for text-dependent speaker verification. *Speech Communication*, *73*, 1–13.

Lopez-Moreno, I., Gonzalez-Dominguez, J., Plchot, O., Martinez, D., Gonzalez-Rodriguez, J., & Moreno, P. (2014). Automatic language identification using deep neural networks. In *Proc. icassp* (pp. 5337–5341).

Lozano-Diez, A., Silnova, A., Matejka, P., Glembek, O., Plchot, O., Pesan, J., ... Gonzalez-Rodriguez, J. (2016). Analysis and Optimization of Bottleneck Features for Speaker Recognition. In *Proc. Odyssey* (pp. 352–357).

Luque Serrano, J. (2012). *Speaker diarization and tracking in multiple-sensor environments* (Unpublished doctoral dissertation). Universitat Politecnica de Catalunya, Barcelona, Spain.

López, V., Fernández, A., García, S., Palade, V., & Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, *250*, 113–141.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml* (Vol. 30).

Matějka, P., Glembek, O., Castaldo, F., Alam, M. J., Plchot, O., Kenny, P., ... Černocky, J. (2011). Full-covariance ubm and heavy-tailed plda in i-vector speaker verification. In *Proc. ICASSP* (pp. 4828–4831).

Matejka, P., Zhang, L., Ng, T., Mallidi, H. S., Glembek, O., Ma, J., & Zhang, B. (2014). Neural network bottleneck features for language identification. *Proc. Odyssey*, 299–304.

Matějka, P., Plchot, O., Soufar, M., Glembek, O., D'haro Enríquez, L. F., Veselý, K., ... Dehak, N. (2012). Patrol team language identification system for DARPA RATS P1 evaluation. In *Interspeech* (pp. 1–4). Portland, Oregon.

McCree, A., & Garcia-Romero, D. (2015). DNN senone MAP multinomial i-vectors for phonotactic language recognition. In *Interspeech* (pp. 394–397).

Mclaren, M., Lei, Y., & Ferre, L. (2015). Advances In Deep Neural Network Approaches To Speaker Recognition. In *Proc. ICASSP* (pp. 4814–4818).

McLaren, M., Vogt, R., Baker, B., & Sridharan, S. (2010). Data-driven background dataset selection for SVM-based speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, *18*(6), 1496–1506.

Miguel, A., Llombart, J., Ortega, A., & Lleida, E. (2017). Tied hidden factors in neural networks for end-to-end speaker recognition. *Proc. Interspeech*, 2819–2823.

Mohamed, A., Dahl, G., & Hinton, G. (2009). Deep belief networks for phone recognition. In *Proc. NIPS* (pp. 1–9).

Mohamed, A., Dahl, G., & Hinton, G. (2012). Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, *20*(1), 14–22. doi: 10.1109/TASL.2011.2109382

Mohamed, A., Yu, D., & Deng, L. (2010). Investigation of full-sequence training of deep belief networks for speech recognition. In *Proc. interspeech* (pp. 2846–2849).

Moreno, A., Lindberg, B., Draxler, C., Richard, G., Choukri, K., Euler, S., & Allen, J. (2000). Speechdat-car. a large speech database for automotive environments. In *Proc. LREC.*

Nadeu, C., Hernando, J., & Gorricho, M. (1995). On the decorrelation of filter-bank energies in speech recognition. In *Proc. eurospeech* (pp. 1381–1384).

Nadeu, C., Macho, D., & Hernando, J. (2001). Time and frequency filtering of filter-bank energies for robust HMM speech recognition. *Speech Communication*, *34*(1–2), 93–114.

Nair, V., & Hinton, G. (2009). 3d object recognition with deep belief nets. In *Proc. NIPS* (pp. 1339–1347).

Nair, V., & Hinton, G. (2010). Rectified linear units improve restricted boltzmann machines. In *Proc. ICML.*

NIST. (2006). The NIST year 2006 speaker recognition evaluation plan.. ([Online]. Available: http://www.nist.gov/speech/tests/spk/2006/index.htm)

NIST. (2010). The NIST year 2010 speaker recognition evaluation plan.. ([Online]. Available: https://www.nist.gov/itl/iad/mig/speaker_recognition_evaluation_2010)

NIST. (2014). *The NIST speaker recognition i-vector machine learning challenge.* ([Online]. Available: http://nist.gov/itl/iad/mig/upload/sre-ivectorchallenge_2013-11-18_r0.pdf)

NIST. (2015). The NIST language recognition i-vector machine learning challenge.. ([Online]. Available: http://www.nist.gov/itl/iad/mig/upload/lre_ivectorchallenge_rel_v2.pdf)

Novoselov, S., Pekhovsky, T., Kudashev, O., Mendelev, V. S., & Prudnikov, A. (2015). Non-linear plda for i-vector speaker verification. In *Proc. interspeech* (pp. 214–218).

Novoselov, S., Pekhovsky, T., & Simonchik, K. (2014). STC speaker recognition system for the NIST i-vector challenge. In *Proc. odyssey* (pp. 231–240).

Novoselov, S., Pekhovsky, T., Simonchik, K., & Shulipa, A. (2014). RBM-PLDA subsystem for the NIST i-vector challenge. In *Proc. Interspeech* (pp. 378–382).

Nugraha, A. A., Liutkus, A., & Vincent, E. (2016). Multichannel Audio Source Separation With Deep Neural Networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *24*(9), 1652–1664.

Oglesby, J., & Mason, J. (1988). Speaker identification using neural nets. *IOA Speech*, *88*, 1357–1363.

Oglesby, J., & Mason, J. (1990). Optimisation of neural models for speaker identification. In *Proc. ICASSP* (pp. 261–264).

Pekhovsky, T., Novoselov, S., Sholokhov, A., & Kudashev, O. (2016). On autoencoders in the i-vector space for speaker recognition. , 217–224.

Pelecanos, J., & Sridharan, S. (2001). Feature Warping for Robust Speaker Verification. In *Proc. odyssey* (pp. 213–218). Crete, Greece.

Prince, S., & Elder, J. (2007). Probabilistic linear discriminant analysis for inferences about identity. In *Proc. ICCV* (pp. 1–8).

Reynolds, D., & Rose, R. (1995). Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing*, *3*(1), 72–83. doi: 10.1109/89.365379

Reynolds, D. A., Quatieri, T. F., & Dunn, R. B. (2000). Speaker verification using adapted gaussian mixture models. *Digital signal processing*, *10*(1-3), 19–41.

Richardson, F., Reynolds, D., & Dehak, N. (2015a). Deep Neural Network Approaches to Speaker and Language Recognition. *IEEE Signal Processing Letters*, *22*(10), 1671–1675.

Richardson, F., Reynolds, D., & Dehak, N. (2015b). A unified deep neural network for speaker and language recognition. In *Interspeech.*

Rudasi, L., & Zahorian, S. A. (1991). Text-independent talker identification with neural networks. In *Proc. ICASSP* (pp. 389–392).

Sainath, T. N., Kingsbury, B., Saon, G., Soltau, H., Mohamed, A.-r., Dahl, G., & Ramabhadran, B. (2015). Deep Convolutional Neural Networks for Large-scale Speech Tasks. *Neural Networks*, *64*, 39–48.

Sainath, T. N., Weiss, R. J., Wilson, K. W., Li, B., Narayanan, A., Variani, E., ... Kim, C. (2017). Multichannel Signal Processing With Deep Neural Networks for Automatic Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *25*(5), 965–979.

Salakhutdinov, R., & Hinton, G. (2009). Semantic hashing. *International Journal of Approximate Reasoning*, *50*(7), 969–978.

Senior, A., Sak, H., & Shafran, I. (2015). Context Dependent Phone Models For LSTM RNN Acoustic Modelling. In *Proc. ICASSP* (pp. 4585–4589).

Senoussaoui, M., Dehak, N., Kenny, P., Dehak, R., & Dumouchel, P. (2012). First attempt of boltzmann machines for speaker verification. In *Proc. odyssey* (pp. 117–121).

Senoussaoui, M., Kenny, P., Stafylakis, T., & Dumouchel, P. (2014, January). A Study of the Cosine Distance-Based Mean Shift for Telephone Speech Diarization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *22*(1), 217–227.

Singer, E., Torres-Carrasquillo, P., Reynolds, D., McCree, A., Richardson, F., Dehak, N., & Sturim, D. (2011). The MITLL NIST LRE 2011 language recognition system. In *Proc. Odyssey* (pp. 209–215).

Snyder, D., Garcia-Romero, D., Povey, D., & Khudanpur, S. (2017). Deep neural network embeddings for text-independent speaker verification. *Proc. Interspeech*, 999–1003.

Snyder, D., Ghahremani, P., Povey, D., Garcia-Romero, D., Carmiel, Y., & Khudanpur, S. (2016). Deep neural network-based speaker embeddings for end-to-end speaker verification. In *Proc. SLT* (pp. 165–170).

Solomonoff, A., Campbell, W., & Boardman, I. (2005). Advances in channel compensation for SVM speaker recognition. In *Proc. ICASSP* (pp. 629–632).

Song, Y., Cui, R., Hong, X., McLoughlin, I., Shi, J., & Dai, L. (2015). Improved language identification using deep bottleneck network. In *Proc. ICASSP* (pp. 4200–4204).

Song, Y., Hong, X., Jiang, B., Cui, R., McLoughlin, I., & Dai, L. (2015). Deep bottleneck network based i-vector representation for language identification. In *Proc. Interspeech* (pp. 398–402).

Stafylakis, T., Kenny, P., Senoussaoui, M., & Dumouchel, P. (2012a). PLDA using gaussian restricted boltzmann machines with application to speaker verification. In *Proc. Interspeech* (pp. 1692–1695).

Stafylakis, T., Kenny, P., Senoussaoui, M., & Dumouchel, P. (2012b). Preliminary investigation of boltzmann machine classifiers for speaker recognition. In *Proc.*

Odyssey (pp. 109–116).

Susskind, J. M., Hinton, G. E., Movellan, J. R., & Anderson, A. K. (2008). Generating facial expressions with deep belief nets. In *Affective computing* (pp. 421–440). InTech.

Sutskever, I., & Hinton, G. (2007). Learning multilevel distributed representations for high-dimensional sequences. In *Artificial intelligence and statistics* (pp. 548–555).

Thai-Nghe, N., Gantner, Z., & Schmidt-Thieme, L. (2010). Cost-sensitive learning methods for imbalanced data. In *The 2010 international joint conference on neural networks (IJCNN)* (pp. 1–8).

Torres-carrasquillo, P. A., Singer, E., Kohler, M. A., & Deller, J. R. (2002). Approaches to language identification using gaussian mixture models and shifted delta cepstral features. In *Proc. ICSLP* (pp. 89–92).

Variani, E., Lei, X., McDermott, E., Lopez Moreno, I., & Gonzalez-Dominguez, J. (2014). Deep neural networks for small footprint text-dependent speaker verification. In *Proc. ICASSP* (pp. 4052–4056).

Vasilakakis, V., Cumani, S., & Laface, P. (2013). Speaker recognition by means of deep belief networks. In *Proc. biometric technologies in forensic science.*

Villalba, J., Brümmer, N., & Dehak, N. (2017). Tied variational autoencoder backends for i-vector speaker recognition. In *Proc. Interspeech* (pp. 1004–1008).

Wang, S., Qian, Y., & Yu, K. (2017). What does the speaker embedding encode? *Proc. Interspeech*, 1497–1501.

Xia, R., & Liu, Y. (2012). Using i-vector space model for emotion recognition. In *Proc. interspeech* (pp. 2227–2230).

Xiang, B., Chaudhari, U. V., Navrátil, J., Ramaswamy, G. N., & Gopinath, R. A. (2002). Short-time Gaussianization for robust speaker verification. In *Proc. ICASSP* (pp. 681–684).

Yu, D., Deng, L., Seide, F. T. B., & Li, G. (2016). *Discriminative pretraining of deep neural networks.* Google Patents. (US Patent 9,235,799)

Zeiler, M. D., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q. V., ... others (2013). On rectified linear units for speech processing. In *Proc. ICASSP* (pp. 3517–3521).

Zhang, S.-X., Chen, Z., Zhao, Y., Li, J., & Gong, Y. (2016). End-to-end attention based text-dependent speaker verification. In *Proc. SLT* (pp. 171–178).

Zhang, X.-L., & Wu, J. (2013). Deep belief networks based voice activity detection.

IEEE Transactions on Audio, Speech, and Language Processing, *21*(4), 697–710.