Universitat Autònoma de Barcelona

Towards Practical Neural Image Compression

Fei Yang

Universitat Autònoma
de Barcelona

Towards Practical Neural Image Compression

A dissertation submitted by **Fei Yang** to the Universitat Autònoma de Barcelona in fulfilment of the degree of **Doctor of Philosophy** in the Departament de Ciències de la Computació.

Bellaterra, October 29, 2021

Director	**Dr. Luis Herranz** Centre de Visió per Computador Universitat Autònoma de Barcelona **Dr. Mikhail G. Mozerov** Centre de Visió per Computador Universitat Autònoma de Barcelona **Dr. Yongmei Cheng** School of Automation Northwestern Polytechnical University
Thesis committee	**Dr. Joan Serra Sagristà** Group on Interactive Coding of Images Universitat Autònoma de Barcelona **Dr. Narciso García** Grupo de Tratamiento de Imágenes Universidad Politécnica de Madrid **Dr. Josep Ramón Morros** Image Processing Group (GPI) Universitat Politècnica de Catalunya

This document was typeset by the author using LaTeX 2ε.

ISBN: 978-84-122714-7-8

Printed by Ediciones Gráficas Rey, S.L.

Acknowledgements

It is a really long journey along with troubles, exploration, uncertainty and happiness. At this moment, there is an important node coming to me, but not the terminal point of learning and research. I am so happy that I can stop here for a while to revisit it, and have a chance to thank the people who gave me helps before.

Four years ago, it was my first time to enjoy the clear sky and fresh air of Barcelona when I first stood in a foreign country, it just like happened yesterday. Thanks for the help from Xialei and Lu at that moment and also a lot of their help after that. Also thanks to Kai, Xiaodong, Qiuyue, Chenshen who are my good partners from the beginning. Especially thanks to Kai, because he invited me to practice in SAF which was the thing I do not like before. But this experience refreshed my understanding of fitness, which is so important for keeping a good state of body for better work and life.

Many thanks to my supervisor, Mikhail. He is so good at mathematics which push me to keep learning. It is his patient guidance that helped me through the initial hard time because this was a new environment to me, and I needed to adapt myself for language, life and research. It is he who taught me a lot about writing, useful tools for research, how to improve communication and so on. It is he who help me publish my first research publication.

Thanks a lot to our group leader and my tutor Joost. He is a very wise and friendly person and an excellent researcher. I still remember that he drove by himself to welcome us at the airport. He is so kind to provide his help to us for our life, work and research. I always can learn new knowledge from each discussion with him.

Special thanks to my supervisor, Luis. I can not finish my thesis without his advising and so much help from him. He always is passionate and rigorous on research, and his conscientious and meticulous to hard work had a great influence on me. He taught me how to analyze problems systematically and comprehensively, then find the most suitable way to solve it. He can speak Chinese very well, and actually he is the best one I knew as a non-native Chinese speaker. ‘Wo zhen de fei chang gan xie ni !’

I would like to thank all the Chinese people in CVC, I was not alone because of

them, especially on some Chinese festivals. They are Xialei, Lu, Yaxing, Zhijie, lei, Kai, Chenshen, Yi, Shiqi, Shun, Danna and Yixiong. I am very happy with all of you.

I am also grateful to all LAMP members: Bogdan, Marc, Oğuz, Javad, Abel, Carola, Laura, Mikel, Aitor, Héctor, Albin, Alex. We had many meaningful group meetings and discussions together which gave me many different insights into different research fields. Especially thanks to Marc again for his very useful pieces of advice for my research and carried me to visit the beautiful scenes in Barcelona.

Thanks to other researchers in CVC, Jose, Dena, Ali, Andres, Sanket, Sudeep, Sounak, Diego, Pau, Gemma, Mohamed. They make our research life full of vitality. I would also like to thank the administration staff of CVC for all the help provided over these years. Thanks to Montse, Eva, Gigi, Mireia, Alexandra, Kevin, Joan.

I want to thank my parents, my brother for their enduring support and encouraging over these years when I was far away from home. This thesis is dedicated to them. In addition, I also should thank my country which gave me a chance to study here. Finally, much love and thanks to Yulu Gao, I am so lucky to meet you!

Abstract

Images and videos are pervasive in our life and communication. With advances in smart and portable devices, high capacity communication networks and high definition cinema, image and video compression are more relevant than ever. Traditional block-based linear transform codecs such as JPEG, H.264/AVC or the recent H.266/VVC are carefully designed to meet not only the rate-distortion criteria, but also the practical requirements of applications.

Recently, a new paradigm based on deep neural networks (i.e. neural image/video compression) has become increasingly popular due to its ability to learn powerful nonlinear transforms and other coding tools directly from data instead of being crafted by humans, as was usual in previous coding formats. While achieving excellent rate-distortion performance, these approaches are still limited mostly to research environments due to heavy models and other practical limitations, such as being limited to function on a particular rate and due to high memory and computational cost. In this thesis we study these practical limitations, and designing more practical neural image compression approaches.

After analyzing the differences between traditional and neural image compression, our first contribution is the modulated autoencoder (MAE), a framework that includes a mechanism to provide multiple rate-distortion options within a single model with comparable performance to independent models. In a second contribution, we propose the slimmable compressive autoencoder (SlimCAE), which in addition to variable rate, can optimize the complexity of the model and thus reduce significanlty the memory and computational burden.

Modern generative models can learn custom image transformation directly from suitable datasets following encoder-decoder architectures, task known as image-to-image (I2I) translation. Building on our previous work, we study the problem of distributed I2I translation, where the latent representation is transmitted through a binary channel and decoded in a remote receiving side. We also propose a variant that can perform both translation and the usual autoencoding functionality.

Finally, we also consider neural video compression, where the autoencoder is typically augmented with temporal prediction via motion compensation. One of the

main bottlenecks of that framework is the optical flow module that estimates the displacement to predict the next frame. Focusing on this module, we propose a method that improves the accuracy of the optical flow estimation and a simplified variant that reduces the computational cost.

Key words: *neural image compression, neural video compression, optical flow, practical neural image compression, compressive autoencoders, image-to-image translation, deep learning*

Resumen

Imágenes y vídeos son omipresentes en nuestra vida y comunicación. Con los avances en dispositivos portátiles e inteligentes, las redes de comunicación de alta capacidad y el cine de alta definición, la codificación de imágenes y vídeos resultan más relevantes que nunca. Los codecs tradicionales basados en transformadas por bloques, como por ejemplo JPEG, H.264/AVC o el reciente H.266/VVC han sido diseñados cuidadosamente no sólo para satisfacer criterios de tasa-distorsión sino también otros requisitos prácticos de las aplicaciones.

Recientemente, un nuevo paradigma de compresión de imágenes y vídeos basado en redes neuronales profundas está suscitando interés debido a su capacidad para aprender potentes transformaciones no lineales y otras herramientas de codificación directamente a partir de los datos, en lugar de ser creadas por humanos, como era habitual en anteriores formatos de codificación. Si bien logran un excelente rendimiento en tasa-distorsión, estos enfoques todavía están limitados principalmente a entornos de investigación, debido a que son modelos pesados y otras limitaciones prácticas, entre ellos estar limitados para funcionar a una tasa particular y debido al alto coste computacional y memoria. En esta tesis estudiamos estas limitaciones prácticas y proponemos métodos para abordarlas.

Después de analizar las diferencias entre la compresión de imágenes tradicional y con redes neuronales, nuestra primera contribución es el autoencoder modulado (MAE), una arquitectura que incluye un mecanismo para proporcionar múltiples opciones de tasa (y su correspondiente distorsión) dentro de un solo modelo, con un rendimiento comparable a los modelos independientes. En una segunda aportación, proponemos el "slimmable compressive autoencoder (SlimCAE)", que además de proporcionar tasa variable, puede también adaptar la complejidad del modelo y así reducir significativamente la memoria y la carga computacional.

Los modelos generativos modernos pueden aprender transformaciones personalizadas de imágenes aprendidas directamente a partir de conjuntos de datos, utilizando arquitecturas de codificador-decodificador. Esta tarea se conoce como traducción de

imagen a imagen (I2I). Sobre la base de nuestro trabajo anterior, estudiamos el problema de la traducción I2I distribuida, donde la representación latente se transmite a través de un canal binario y se decodifica en un lado receptor remoto. También proponemos una variante que puede realizar tanto la traducción como la transmisión de la imagen original en un mismo modelo.

Finalmente, también consideramos la compresión de video con redes neuronales profundas, donde el modelo se aumenta con predicción temporal a través de compensación de movimiento. Uno de los principales cuellos de botella de ese marco es el módulo de estimación de flujo óptico que estima el desplazamiento para predecir el siguiente cuadro. Centrándonos en este módulo, proponemos un método que mejora la precisión de la estimación del flujo óptico y una variante simplificada que reduce el coste computacional.

Palabras clave: *reconocimiento visual, aprendizaje auto-supervisado, aprendizaje mediante ránquings, evaluación de calidad de imagen, contador de personas, aprendizaje continuo*

Resum

Les imatges i els vídeos són pervasius en les nostres vides i comunicacions. Amb el avenços en dispositius portables i intel·ligents, xarxes de comunicació d'alta capacitat i cine d'alta definició, la compressió d'imatges i vídeos es més rellevant que mai. Els còdecs de transformació lineals tradicionals basats en block com JPEG, H.264/AVC o el recent H.266/VVC són acuradament designats per satisfer no tan sols criteris de distorsió, sinó a mes a mes el requisits pràctics de les aplicacions.

Recentment, un nou paradigma basat en xarxes neuronals (p.e. compressió neuronal de vídeo i imatge) ha anat incrementant la seva popularitat degut a la habilitat per aprendre potents transformacions no lineals i altres eines de codi directament de les dades en comptes de ser dissenyades per humans, com era habitual en format de còdecs anteriors. Mentre que obtenen un rendiment excel·lent de distorsió, aquest sistemes estan limitats al àmbit de la recerca degut a l'alta densitat del models, cost computacionals i de memòria. En aquesta tesis estudiarem aquestes limitacions pràctiques i proposarem dissenys de xarxes per compressió de imatges més eficients.

Després d'analitzar les diferencies entre models de compressió d'imatge tradicionals i neuronals, la nostre primera contribució es un "autoencoder" modulat (MAE), un framework que inclou un mecanisme que proporciona múltiples mesures de distorsió dintre un sól model amb un rendiment comparable a models independents. En la segona contribució, proposem el que anomenen "slimmable compressive autoencoder (SlimCAE)", que afegint un mesurament variable podem optimitzar la complexitat del model i per tant reduir de manera significativa la memòria utilitzada i la carrega computacional.

Un model generatiu modern pot aprendre transformacions d'imatge personalitzades directament de datasets adients seguint arquitectures encoder-decoder, aquestes tasques son conegudes com translació imatge-a-imatge. Desenvolupant el nostre treball anterior, estudiem el problema de distribució de translació d'imatge-a-imatge, on la representació latent es transmesa a través d'un canal binari i descodificada en una banda receptora remota. També proposem una variant que pot dur a terme la translació i l'usual funcionalitat d'autoencoding.

Finalment, també considerem compressió de vídeo neuronal, on l'autoencoder

es típicament augmentat amb prediccions temporals per mitjà de compensació de moviment. Un del principals cul de sac del framework es el mòdul de flux òptic que estima el desplaçament per predir la següent imatge en el vídeo. Prestant atenció en aquest mòdul, proposem un mètode que millora la precisió del flux òptic i una variant que redueix el cost computacional.

Paraules clau: *compressió neuronal de imatge, compressió neuronal de vídeo, flux òptic, pràctica compressió neuronal de imatge, autoencoders compressius, traducció d'imatge a imatge, aprenentatge profund*

Contents

List of Figures

List of Tables

1 Introduction

With the great advancement of global information technology and industry, communication networks and digital information have become nowadays an inseparable part of people's daily life. In the information age, high speed, efficiency, and convenience are requirements for people to access digital content effectively. In particular, visual information (e.g. image, video, and 3D vision) constitutes the largest portion of multimedia data in modern communication networks.

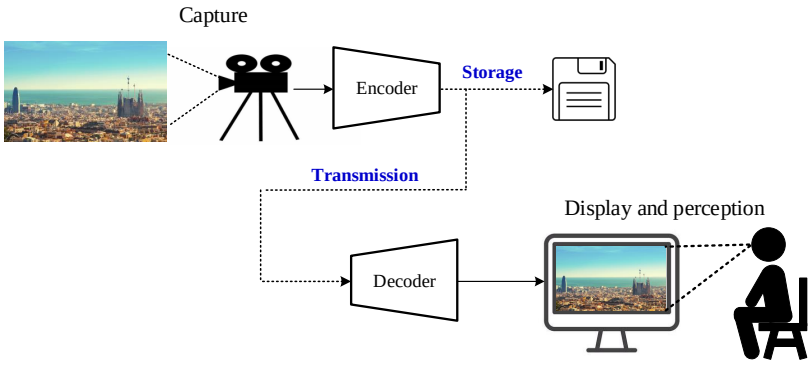


Figure 1.1 – Benefits from data compression: save storage space, speed up communication and reduce bandwidth

For example, today's mobile phones are equipped with high resolution image and video cameras, narrowing the gap with professional cameras. Given the availability of these devices, users produce an increasingly large amount of high resolution visual information every day. This visual content is typically stored, shared with family and friends or uploaded to social media.

1.1 The image communication problem

In general, digital communication systems address the problem of transmitting a source signal via binary digits (i.e. *bits*) through a digital channel in order to reconstruct the

source signal in the receiver side. If the recovered signal is exactly the transmitted one, this particular case is referred to as *lossless compression*. The objective is to remove as much statistically redundant information as possible. If, in contrast, the reconstructed signal is just an approximation, then this case is known as *lossy compression*, since there is an irreversible loss of information. Lossy compression can achieve significantly higher compression rates than lossless compression, and the system is usually designed so the loss is imperceptible or minimally perceptible to the receiving observer. This is the common case in audiovisual signals, where the properties of human visual and auditive systems can be exploited to remove perceptual redundancy.

In general, the lossy communication problem (see Fig. 1.2) is typically posed as *conveying source data with the highest fidelity possible without exceeding an available bit rate* (or conversely, as conveying the source data using the lowest bit rate possible while maintaining a specified reproduction fidelity) [92, 115]. Thus there is a fundamental trade-off between the amount of information transmitted (*rate*) and the quality of the approximation of the recovered signal (measured as *distortion*). Most image and video coding standards are examples of lossy compression. For example, there are JPEG [107], JPEG 2000 [78, 95], BPG [96] for image compression, and H.262 [91], H.264 [116], HEVC [96] for video compression. As we can see, there are two fundamental objectives to minimize in lossy compression (assuming a perfect channel, i.e. we do not consider channel coding):

- **(Bit) rate** (R) measures the amount of information transmitted per second, and is typically limited by the capacity of the transmission channel (which in practice should also account for bits used for channel coding, such as error-correction overhead). In the case of storage instead of transmission, rate determines the amount of storage required to store a given piece of content. In images, rate is often represented in terms of bits per pixel (*bpp*).

- **Distortion** (D) measures how different the recovered signal is compared to the source signal (e.g. reconstruction error). Distortion is the result of lossy components in the encoder-channel-decoder pipeline (e.g. quantization, transmission errors). In images, the distortion between the original image and the reconstructed image is usually measure with metrics such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM)[111].

The theoretical basis for lossy compression is the *rate-distortion theory*, which is a major branch of information theory.

In this thesis, lossy image compression is our main research focus. Fig. 1.2 shows a lossy image coding pipeline consisting of an encoder and a decoder. Given an image $\mathbf{x}$, the encoder can generate its binary representation (i.e. bitstream $\mathbf{b}$). The decoder converts the bitstream $\mathbf{b}$ into the reconstructed image $\hat{\mathbf{x}}$.

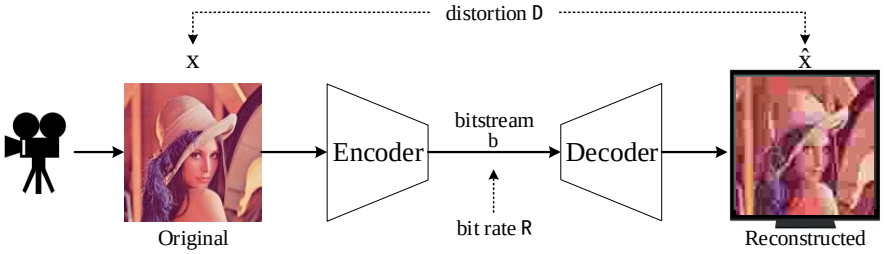


Figure 1.2 – A typical lossy image compression system.

Optimizing rate, distortion and their tradeoff is thus the main objective of a lossy image compression system. The performance (known as *coding efficiency* or *rate-distortion performance*) is generally characterized with rate-distortion operational curves [115]. However, in practical image transmission systems, there are additional issues that need to be carefully considered:

- **Complexity**. For a codec to be practical in a certain scenario, its implementation needs to consider the computational resources available in the device (e.g. computer, smartphone, camera, TV set). We can distinguish between *computational complexity*, the *memory capacity* and *memory access* requirements. Indirectly, the computational complexity can also have significant impact on the energy consumption, battery duration and even device life.

- **Latency** is the total time since the original image (or video frame) is captured and the reconstructed image is presented to the receiver. There are many factors which can influence the latency, such as the delays of image and channel codecs, data processing and buffering latency, as well as the transmission time through the transmission channel (e.g. copper wires, optical fibers, wireless communication channels). In this thesis we will consider only the coding latency, related with encoder and decoder architectures and coding algorithms.

- **Variable rate**. The ability to dynamically adjust the rate-distortion trade-off according to a target rate or to varying rate requirements is particuarly important in video coding. In scenarios such as video streaming and videoconference rate needs to be adapted to the particular network conditions.

- **Domain specific**. While image codecs can be general purpose, knowing the characteristics of the application domain, specific tools can be designed exploiting domain-specific properties (e.g. screencast, videoconference, medical images, sports).

- **Interoperability**. A main motivation of image and video compression and transmission is to share content. A key requirement is that encoders and decoders are compatible. Due to the impossibility of updating decoders every time a new coding format appears or is updated, keeping a decoder compatible with legacy formats (i.e. *backward compatibility*), is an important design concern. Similarly, a codec can be designed so it can process inputs of a future version of the coding format (i.e. *forward compatibility*).

In this thesis we pay special attention to these practical constraints. Wiegand and Schwarz state the *practical image coding problem* as *given a maximum allowed delay and a maximum allowed complexity, achieve an optimal tradeoff between bit rate and distortion for the range of network environments envisioned in the scope of the applications.* [115]

1.2 Image compression paradigms

We briefly compare the dominant image compression paradigm of the last four decades and the emerging paradigm that relies on deep neural networks and machine learning.

1.2.1 Traditional image compression

An image encoder is essentially a mapping function $\mathbf{b} = E(\mathbf{x})$ that maps the continuous-valued image $\mathbf{x}$ to a discrete binary sequence $\mathbf{b}$. The decoder implements the inverse mapping $\hat{\mathbf{x}} = D(\mathbf{b})$. Internally, the encoder first quantizes the image into a sequence of symbols $\mathbf{i}$ (lossy) which then encodes with the entropy encoder $\mathbf{b} = \gamma(\mathbf{i})$ and inverted by the entropy decoder (lossless).

Quantizing an image directly in the pixel domain is extremely difficult. So in practice, image codecs follow the transform coding paradigm [32], where an analysis transform is applied prior to quantization (see Fig. 1.3) and a synthesis transform inverts the mapping in the decoder. The objective of this transform is to decorrelate and concentrate the energy in few coefficients. Then scalar quantization is sufficient to efficiently quantize the resulting coefficients. Another advantage of transform coding is that, for properly designed transforms, the human visual system is less sensitive perceptually to quantization in the transformed space than to quantization in the original pixel space. In general, the analysis and synthesis transforms are orthogonal linear transforms, such as the discrete cosine transform (DCT) [3, 4] and the discrete wavelet transform (DWT) [93]. The former is used in the vast majority of image a video coding standards. These transforms (as well as other coding tools) are carefully handcrafted by researchers and engineers based on signal processing theory.

Practical considerations are essential in traditional image and video coding stan-

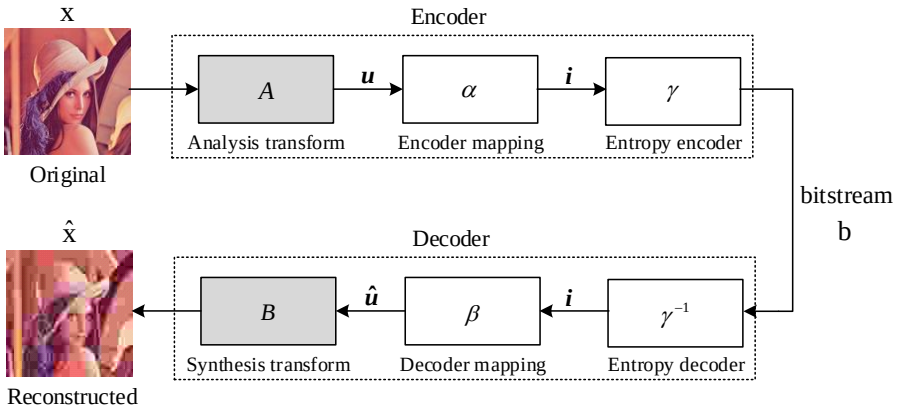


Figure 1.3 – Block diagram for an image transform coding system. The analysis transform converts an image $\mathbf{x}$ into corresponding transform coefficients $\mathbf{u}$, which will be mapped onto quantization indexs with an encoder mapping α. The quantization indexes $\mathbf{i}$ are coded by using a lossless encoder γ, resulting in a bitstream $\mathbf{b}$. In the transform decoder, the bitstream $\mathbf{b}$ is decoded into the quantization indexes $\mathbf{i}$ with the inverse lossless decoder γ^{-1}. Then, the reconstructed transform coefficients $\hat{\mathbf{u}}$ can be obtained with the decoder mapping β. The reconstructed image $\hat{\mathbf{x}}$ is obtained by applying the synthesis transform on the reconstructed transform coefficients $\mathbf{u}$.

dards. In particular, the image is typically partitioned into blocks, which are transformed locally using a small DCT (typically 8×8 blocks). This significantly reduces the complexity, both computational and memory, leading to compact and efficient implementations that can deployed at scale. Blocks are typically encoded and decoded in an autoregressive fashion (one by one), and the current block is predicted based on previously encoded blocks (and frames in video coding). Other practical requirements are , such as variable rate, progressive decoding, spatial/temporal/quality scalability, and backward compatibility. To cover different application use cases and implementation complexities, standards define profiles (sets of coding tools), and levels (appropriate ranges of values of different properties used in the profile).

1.2.2 Neural image compression

Recently, advances in machine learning and the availability of large datasets and suitable hardware platforms (e.g. GPUs) have driven a paradigm shift in areas such as computer vision, audio processing and natural language processing. In this new paradigm, known as *deep learning*, multi-layer neural networks with millions of

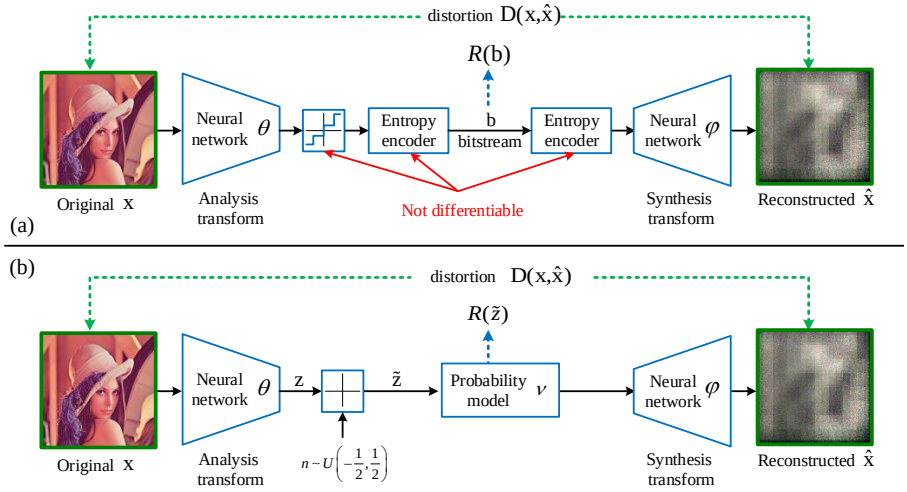


Figure 1.4 – Neural image compression. (a) Basic framework of NIC method consists of deep autoencoder with the learnable parameters θ and ϕ, quantization and entropy coding. It can not be trained directly due to non differentiability of quantizatin and entropy coding; (b) End-to-end trainable framework with differentiable proxies, in which the quantization step is simulated by adding uniform noise, and rate is estimated by a probability model implemented as neural network with the learnable parameters ν.

parameters define flexible models, whose parameters are adjusted from data. Now, instead of carefully handcrafted by engineers, models are learned from data. The role of the engineer focuses now in designing the architecture and learning algorithms, and collecting suitable data.

Image and video coding have been also influenced by deep learning, with the emergence of the *neural image compression* (NIC) paradigm [12, 103, 104], also known as learned image compression or deep image compression. While still following the transform coding pipeline (i.e. transform-quantization-entropy coding), block-based linear transforms are now replaced by more powerful non-linear transforms implemented with deep neural networks, fixed entropy models by parametric probability models, and to whole pipeline is trained to explicitly minimize jointly rate and distortion $D(\mathbf{x}, \hat{\mathbf{x}}) + \lambda R(\mathbf{b})$, where λ controls the specific rate-distortion tradeoff.

From a machine learning perspective, the architecture resembles an autoencoder, but extended with quantization and entropy coding (see Fig. 1.4a) to enable compres-

sion of the latent representation [12, 13, 70, 71]. Thus, this architecture has been termed *compressive autoencoder* (CAE) [103]. One important caveat is that deep learning models are trained using gradient descent, which requires to compute the gradient of the trainable parameters. Gradients at different layers are computed using the backpropagation algorithm, which requires all parametric functions being differentiable. Unfortunately, that is not the case for quantization and entropy coding. In practice, during training quantization is replaced by a differentiable proxy, such as adding uniform noise [12], and entropy coding is bypassed, approximating rate by the entropy of the latent representation (see Fig. 1.4b).

Neural image compression has incorporated insights and tools from neuroscience and machine learning. Notable examples are the *generalized divisive normalization* (GDN) [10], which allows achieving high compression performance with relatively few neural layers, and generative adversarial networks, which allow transmitting realistic images at very low rates [2] by optimizing perceptual losses.

Researchers have explored many different architectures as the nonlinear image transform, e.g., convolutional neural network (CNN) [12, 13, 70, 71], recurrent neural network (RNN) [48, 104, 105], generative adversarial network (GAN) [2, 67, 123], variational autoencoder (VAE) [12, 13, 70, 71]. In addition, some methods improved RD performance of image compression with more accurate and complex entropy models, such as hyperpriors [13] and contextual models [56, 58, 66, 70, 71].

1.3 Motivation

1.3.1 Practical neural image compression

While achieving competitive and sometimes even state-of-the-art rate-distortion performance, NIC has its own drawbacks in practice compared to traditional image compression (TIC), which prevents NIC codecs from being attractive in practice. In this thesis we look beyond rate-distortion performance, and pay attention to the practical limitations of this new paradigm.

Table. 1.1 summarizes and compares the differences between the two image coding paradigms in terms of methodological and practical characteristics. One important difference is that TIC operates in blocks in an iterative fashion (one block after another in sequence) using small linear transforms, while NIC operates in feed-forward fashion (a single pass, not iterative) using complex nonlinear deep transforms. This difference in the size of the transforms leads to high demands in terms of memory and computation, whose nature also requires specific hardware (e.g. GPUs), also leading to high encoding latency. The complexity in TIC arises specially in the encoder due to its block-based iterative nature and the cost of intra-prediction tools. Encoding

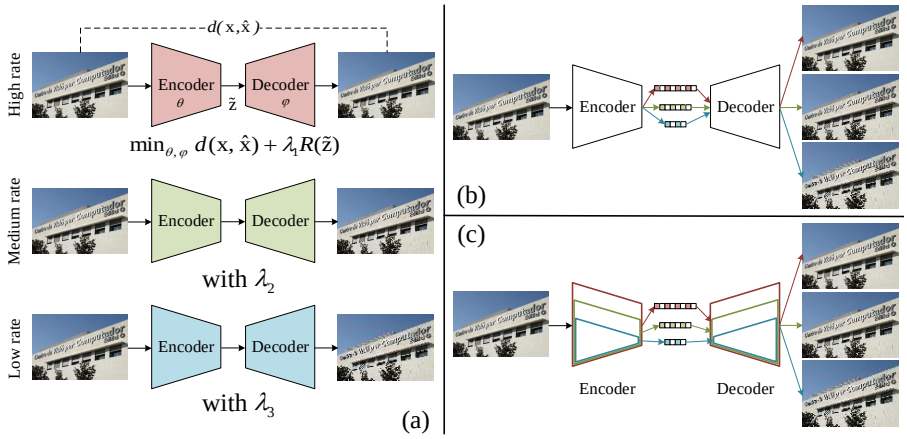


Figure 1.5 – Towards practical neural image compression: (a) independent models for variable rate; (b) variable rate within one single autoencoder; (c) complexity control with variable rate.

is essentially a combinatorial problem where the encoder searchs optimal coding mode and parameters for every block. However, this encoding complexity can be alleviated (at the cost of worse rate-distortion performance) by limiting the number of candidate coding modes and parameters. NIC has additional limitations such as requiring a training stage, and being trained for a particular rate-distortion tradeoff, lacking mechanisms to vary the rate-distortion tradeoff.

In this thesis, we focus on the problems of variable rate and complexity control and adaptation in NIC (see Fig. 1.5).

Table 1.1 – Difference between traditional and neural image compression in practice.

Characteristics		**Traditional image compression**	**Neural image compression**
Transform	**Linear/nonlinear**	Linear	Nonlinear and deep
	Local/global	Local 8x8 to 32x32 (DCT), global (wavelets)	Typically global
Memory requirements		Very low (both transform params and features)	High (network params and features)
Computation	**Computational cost**	Very low (although repeated in many blocks)	Very high
	Dynamically adaptive	Yes (restricting coding tools, search modes, etc.)	No
Training time		N/A (typically handcrafted coding tools)	High
Variable rate		Yes	Typically not (fixed R-D tradeoff)

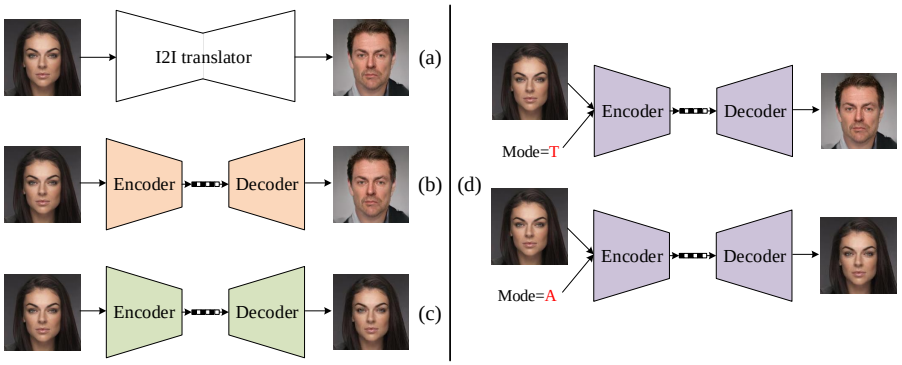


Figure 1.6 – Joint I2I translation and image compression: (a) normal I2I translation; (b) distributed I2I translation with compressed latent representation; (c) neural image compression; (d) a unified framework combining I2I translation and image compression, in which the operating mode (T indicates translation and A indicates autoencoding) is one of inputs as the condition information.

1.3.2 Joint transmission and translation

Neural image compression can be seen as a distributed autoencoder, where we encode an image into a binary latent representation, which then is reconstructed by a remote decoder. Encoder-decoder architectures are pervasive in computer vision, and image autoencoding is just one particular task that can be realized with that approach. Thus, in this thesis we also study how other architectures and tasks can be distributed by integrating quantization and entropy coding.

We focus on the particular task of *image-to-image (I2I) translation*, which addresses the problem of learning to transform images from a source domain to a target domain. This has numerous applications in image restoration and enhancement (e.g. colorisation, super resolution, deblurring), but also more complex data-driven transformations (e.g. style transfer, face attribute modification, scene synthesis, zebra-to-horse). Thus, we study the problem of distributed I2I translation, where the encoding is performed at the sender side, and the decoding at the receiver side, and the coded representation is either transmitted through a communications channel or stored (see Fig. 1.6).

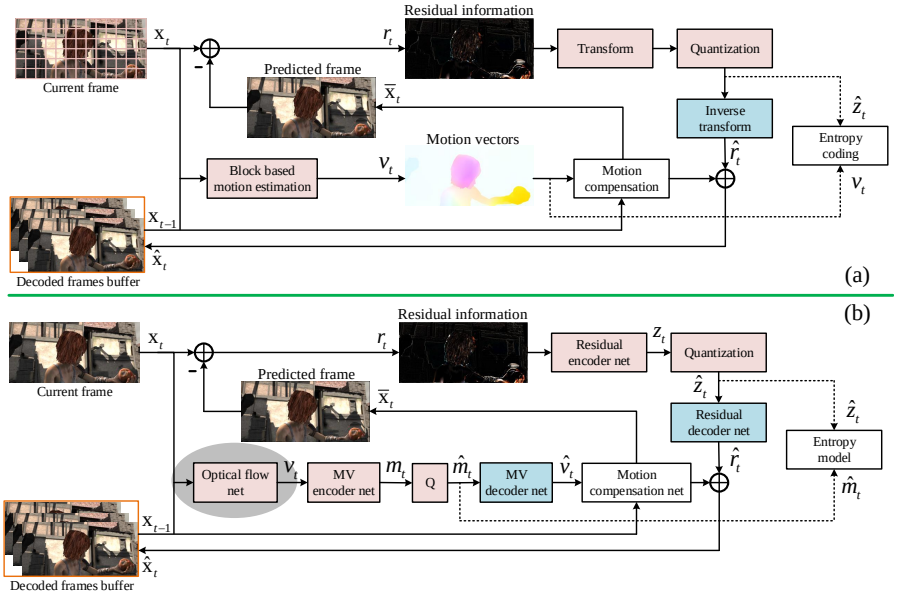


Figure 1.7 – (a): Predictive coding architecture used by the traditional video codec (e.g. H.264 [116] and H.265 [96]); (b): neural video compression framework [60]. The modules with yellow color are not included in the decoder side.

1.3.3 Towards practical neural video compression

Beyond the practical neural image compression, the more general goal is the problem of practical neural video compression (NVC). Traditional video compression (TVC) follows the hybrid video coding combining predictive coding and transform coding (see Fig. 1.7 (a)). More specifically, most video codecs are based on motion-compensated DCT. Some recent works have extended neural image compression to video [60, 85], by combining temporal prediction via optical flow estimation and residual encoding with a compressive autoencoder (see Fig. 1.7 (b)). Thus, these methods still implementing hybrid coding frameworks where transforms and motion estimation are performed via deep neural networks, with the potential advantage of being end-to-end trainable.

Thus, effective temporal prediction of the next frame from a previous one is a crucial factor to achieve high rate-distortion performance in video. Once the predicted frame is obtained, the difference image (i.e. residual information) is encoded with the transform. In general, the difference image has much less information than the original frame and can be encoded with much fewer bits (especially in scenes with

little motion). However, accurate estimation of the motion of pixels is crucial to obtain a good prediction, which will have a significant impact on the rate-distortion performance. Fig. 1.7 illustrates this process for both TVC and NVC. In TVC, encoding is block-based, so motion prediction is also performed on a block basis via block matching, followed by a block-based transform of the residual. In contrast, NVC process the whole frame and the motion estimation is pixel-based via optical flow, followed by a deep transform of the difference image. Therefore an accurate and efficient optical flow estimation algorithm is crucial for NVC to achieve good rate-distortion performance as well as to not being too complex in practice.

1.4 Objectives and approach

As the above discrussion, our objective is to contribute to the development of practical neural image and video compression frameworks. In this thesis we focus on images, with the 5th chapter addressing accurate motion estimation problem as the preliminary work to videos. Based on the compressive autoencoder framework, we propose modifications to achieve:

1.4.1 Variable rate neural image compression

Most of NIC methods [12, 13, 70, 71, 103] jointly optimize rate and distortion at a particular rate-distortion (R-D) tradeoff. It is necessary to train multiple independent models with the different R-D tradeoffs to realize variable rate, although NIC has shown better R-D performance than the traditional image codecs. This is an obvious limitation in practice, since it requires training and storing each model separately, resulting in large memory requirement.

To address variable rate in NIC, we propose *variable rate deep image compression with modulated autoencoders* (MAEs) in Chapter 2. This method can adapt the representations of a shared autoencoder at different layers to a specific R-D tradeoff via a modulating network. The proposed method can achieve almost the same RD performance of independent models with much fewer overall parameters.

1.4.2 Efficient and complexity-adaptive compression

Complexity is one of most important characteristics for practical neural image compression. However, NIC usually is very heavy in model size compared with traditional image codecs, which also cause high memory requirement and much coding time, even with parallel computation on GPUs. Almost all the NIC methods need to train multiple models independently with different capacities for different complexity requirements.

In addition, this cost will be much larger considering the variable rate meanwhile.

To address the problem of complexity control in NIC, we propose *slimmable compressive autoencoders* (SlimCAEs) in Chapter 3, where we show that applying slimming mechanism with one single model can enable both adaptive complexity and variable rate in NIC. Especially, We crucially observe that each R-D tradeoff has an corresponding minimum capacity that additional filters in neural network can not provide any benifits on R-D performance. Thus, it requests us to find the optimal R-D tradeoffs λs depending on different capacities. Then, we propose λ-scheduling which can obtain optimal λs by alternating between the model optimization and adjusting the values of λs during training. SlimCAEs can not only achieve variable rate but also complexity adaptation.

1.4.3 Integrated framework for distributed image-to-image translation

A short bitstream is the practical requirement for the distributed image-to-image translation, which is also a natural extension of NIC because of their similar architecture based on neural networks. Beyond of this, I2I translation can also be viewed as an additional functionality of normal neural image compression (e.g. autoencoding with compressed latent representation). For example, people may need a option to decide showing their real face or a fake one with face translation in a videoconference.

To achieve these goals, we propose *a novel framework for I2I translation and image Compression* combining these two paradigms. We consider multi-domain image synthesis as I2I translation task. Firstly, we propose the I2Icodec by integrating quantization and entropy coding into an distributed I2I translation framework, which can produce the compressed latent representation for the transmission. Then, we further propose a unified framework that allows both translation and autoencoding capabilities in a single codec. The proposed adaptive residual blocks conditioned on the translation/compression mode provide flexible adaptation to the desired functionality. The experiments show promising results in both I2I translation and image compression using a single model.

1.4.4 Efficient and effective optical flow estimation

To pursue more accurate and efficient motion estimation to improve R-D performance of neural video compression, we explore to introduce multiple frames into optical flow estimation. Approaches that use more than two consecutive video frames in the optical flow estimation have a long research history. However, most of such methods utilize extra information for a pre-processing flow prediction or for a post-processing flow correction and filtering. Different from previously developed techniques, we propose

improved discrete optical flow estimation with triple image matching cost, in which a new algorithm is designed for the likelihood function calculation (alternatively the matching cost volume) that is used in the maximum a posteriori estimation. We exploit the fact that in general, optical flow is locally constant in the sense of time and the likelihood function depends on both the previous and the future frame. Implementation of our idea increases the robustness of optical flow estimation. In addition, we also propose a simplified variant that reduces the computational cost and keep comparable accuracy.

2 Variable Rate Deep Image Compression with A Modulated Autoencoder[1]

2.1 Introduction

In many scenarios, communication networks or storage devices may impose a constraint on the maximum bitrate, which requires the image encoder to adapt to a given bitrate budget. In some applications this constraint may even change dynamically over time (e.g. video transmission). In all these cases, a bitrate control mechanism is required, and it is available in most traditional image and video compression codecs. In general, reducing the bitrate causes an increase in the distortion, i.e. there is a rate-distortion (R-D) tradeoff. This mechanism is typically based on scaling the latent representation prior to quantization to obtain finer or coarser quantizations, and then inverting the scaling at the decoder side (Fig. 2.1a).

Traditional block DCT-based image compression methods enable control of the rate via quantization tables that scale DCT coefficients according to the target rate. However, in the NIC paradigm, the parameters of the encoder and decoder are learned from certain image data by jointly minimizing rate and distortion at a particular R-D tradeoff. Thus, providing compression at different bitrates requires an independent model for every R-D tradeoff. This is an obvious limitation, since it requires storing each model separately, resulting in large memory footprint.

To address this limitation, Theis et al. [103] use a single autoencoder whose bottleneck representation is scaled before quantization depending on the target bitrate (Fig. 2.1b). However, this approach only considers the importance of different channels from the bottleneck representation of learned autoencoders under R-D tradeoff constraint. In addition, the autoencoder is optimized for a single specific R-D tradeoff (typically high bitrate). These aspects lead to a drop in performance for low bitrates and a narrow effective range of bitrates. Recurrent neural networks can also realize variable rate coding [104], yet are demanding computationally. Cai *et al.* [22] proposed a multi-scale decomposition network, each scale targeting a different rate.

In this chapter, in order to tackle the limitations of multiple independent models and bottleneck scaling, we formulate the problem of variable R-D optimization for NIC, and propose the *modulated autoencoder* (MAE) framework, where the representations at different layers of a shared autoencoder are adapted to a specific R-D tradeoff via

[1] This chapter is based on a publication in IEEE Signal Processing Letters (SPL 2020) [122].

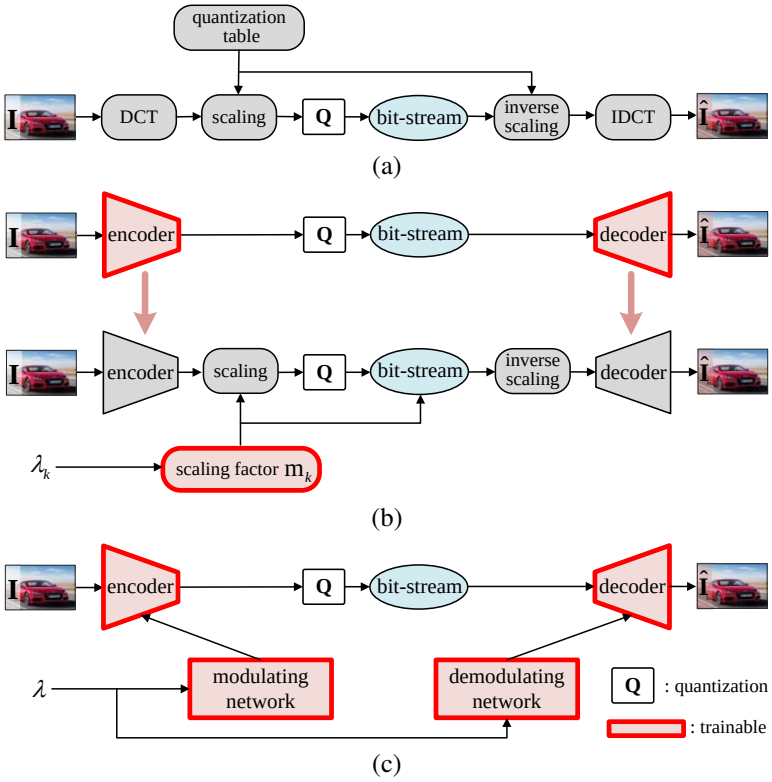


Figure 2.1 – Image compression and R-D control: (a) JPEG transform, (b) pre-trained autoencoder with bottleneck scaling [103], and (c) our proposed modulated autoencoder with joint training. Entropy coding/decoding is omitted for simplicity.

a modulating network. The modulating network is conditioned on the target R-D tradeoff, and synchronized with the actual tradeoff optimized to learn the parameters of the autoencoder and the modulating network. MAEs can achieve almost the same operational R-D points of independent models with much fewer overall parameters (i.e. just the shared autoencoder plus the small overhead of the modulating network). Multi-layer modulation does not suffer from the main limitations of bottleneck scaling, namely, drop in performance for low rates, and shrinkage of the effective range of bitrates. Concurrently, Choi et al. [25] proposed a conditional autoencoder (cAE) with a similar mechanism to achieve variable rate.

2.2 Background

Almost all lossy image and video compression approaches follow the transform coding paradigm [32]. The basic structure is a transform $\mathbf{z} = f(\mathbf{x})$ that takes an input image $\mathbf{x} \in \mathbb{R}^N$ and obtains a transformed representation $\mathbf{z}$, followed by a quantizer $\mathbf{q} = Q(\mathbf{z})$ where $\mathbf{q} \in \mathbb{Z}^D$ is a discrete-valued vector. The decoder reverses the quantization (i.e. dequantizer $\hat{\mathbf{z}} = Q^{-1}(\mathbf{q})$) and the transform (i.e. inverse transform) as $\hat{\mathbf{x}} = g(\hat{\mathbf{z}})$ reconstructing the output image $\hat{\mathbf{x}} \in \mathbb{R}^N$. Before the transmission (or storage), the discrete-valued vector $\mathbf{q}$ is binarized and serialized into a *bitstream* $\mathbf{b}$. Entropy coding [117] is used to exploit the statistical redundancy in that bitstream and reduce its length.

In NIC, the handcrafted analysis and synthesis transforms are replaced by the encoder $\mathbf{z} = f(\mathbf{x};\theta)$ and decoder $\hat{\mathbf{x}} = g(\hat{\mathbf{z}};\phi)$ of a convolutional autoencoder, parametrized by θ and ϕ. The fundamental difference is that the transforms are not designed but *learned* from training data. The model is typically trained by minimizing the optimization problem

$$\arg\min_{\theta,\phi} R(\mathbf{b}) + \lambda D(\mathbf{x},\hat{\mathbf{x}}), \tag{2.1}$$

where $R(\mathbf{b})$ measures the rate of the bitstream $\mathbf{b}$ and $D(\hat{\mathbf{x}},\mathbf{x})$ represents a distortion metric between $\mathbf{x}$ and $\hat{\mathbf{x}}$, and the Lagrange multiplier λ controls the R-D tradeoff. Note that λ is fixed in this case. The problem is solved using gradient descent and backpropagation [87].

To make the model differentiable, which is required to apply backpropagation, during training the quantizer is replaced by a differentiable proxy function [12, 103, 104]. Similarly, entropy coding is invertible, but it is necessary to compute the length of the bitstream $\mathbf{b}$. This is usually approximated by the entropy of the distribution of the quantized vector, $R(\mathbf{b}) \approx H[P_{\mathbf{q}}]$, which is a lower bound of the actual bitstream length.

In this paper, we will use scalar quantization by (element-wise) rounding to the nearest neighboor, i.e. $\mathbf{q} = \lfloor \mathbf{z} \rceil$, which will be replaced by additive uniform noise as proxy during training, i.e. $\tilde{\mathbf{z}} = \mathbf{z} + \Delta\mathbf{z}$, with $\Delta\mathbf{z} \sim \mathcal{U}(-\frac{1}{2},\frac{1}{2})$. There is no de-quantization in the decoder, and the reconstructed representation is simply $\hat{\mathbf{z}} = \mathbf{q}$. To estimate the entropy we will use the entropy model described in [12] to approximate $P_{\mathbf{q}}$ by $p_{\tilde{\mathbf{z}}}(\tilde{\mathbf{z}})$. Finally, we will use mean squared error (MSE) as a distortion metric. With these particular choices, (2.1) becomes

$$\arg\min_{\theta,\phi} R(\tilde{\mathbf{z}};\theta) + \lambda D(\mathbf{x},\hat{\mathbf{x}};\theta,\phi), \tag{2.2}$$

with

$$R(\tilde{\mathbf{z}};\theta) = \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}},\Delta\mathbf{z}\sim\mathcal{U}}\left[-\log_2 p_{\tilde{\mathbf{z}}}(\tilde{\mathbf{z}})\right], \tag{2.3}$$

$$D\left(\mathbf{x},\hat{\mathbf{x}};\theta,\phi\right) = \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}},\Delta\mathbf{z}\sim\mathcal{U}}\left[\|\mathbf{x}-\hat{\mathbf{x}}\|^2\right]. \tag{2.4}$$

2.3 Multi-rate neural image compression with modulated autoencoders

2.3.1 Problem definition

We are interested in NIC models that can operate satisfactorily on different R-D tradeoffs, and adapt to a specific one when required. Note that Eq.(2.2) optimizes rate and distortion for a fixed tradeoff λ. We extend that formulation to multiple R-D tradeoffs (i.e. $\lambda \in \Lambda = \{\lambda_1,\ldots,\lambda_M\}$) as the *multi-rate-distortion problem*

$$\underset{\theta,\phi}{\operatorname{argmin}} \sum_{\lambda\in\Lambda} \left[R(\tilde{\mathbf{z}};\theta,\lambda) + \lambda D\left(\mathbf{x},\hat{\mathbf{x}};\theta,\phi,\lambda\right)\right], \tag{2.5}$$

with

$$R(\tilde{\mathbf{z}};\theta,\lambda) = \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}},\Delta\mathbf{z}\sim\mathcal{U}}\left[-\log_2 p_{\tilde{\mathbf{z}}}(\tilde{\mathbf{z}})\right], \tag{2.6}$$

$$D\left(\mathbf{x},\hat{\mathbf{x}};\theta,\phi,\lambda\right) = \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}},\Delta\mathbf{z}\sim\mathcal{U}}\left[\|\mathbf{x}-\hat{\mathbf{x}}\|^2\right], \tag{2.7}$$

where we are simplifying the notation by omitting features dependency on λ, i.e. $\tilde{\mathbf{z}} = \tilde{\mathbf{z}}(\lambda) = f(\mathbf{x};\theta,\lambda)$ and $\hat{\mathbf{x}} = \hat{\mathbf{x}}(\lambda) = g\left(\tilde{\mathbf{z}}(\lambda);\phi,\lambda\right)$. This formulation can be easily extended to a continuous range of tradeoffs. Note also that these optimization problems assume that all R-D operational points are equally important. It could be possible to integrate an importance function $I(\lambda)$ to further give more importance to certain R-D operational points if required. We assume uniform importance (continuous or discrete) for simplicity.

2.3.2 Bottleneck scaling

A possible way to make the encoder and decoder aware of λ is simply scaling the latent representation in the bottleneck before quantization (implicitly scaling the quantization bin), and then inverting the scaling in the decoder. In this case, $\mathbf{q} = Q(\mathbf{z}\odot\mathbf{s}(\lambda))$ and $\hat{\mathbf{x}}(\lambda) = g\left(\hat{\mathbf{z}}(\lambda)\odot(\mathbf{1}/\mathbf{s}(\lambda));\phi\right)$, where $\mathbf{s}(\lambda)$ is the specific scaling factor for the tradeoff λ. Conventional codecs use predefined tables for $\mathbf{s}(\lambda)$ (the descaling is often implicitly subsumed in the dequantization, e.g. JPEG), while other approaches [103] keep

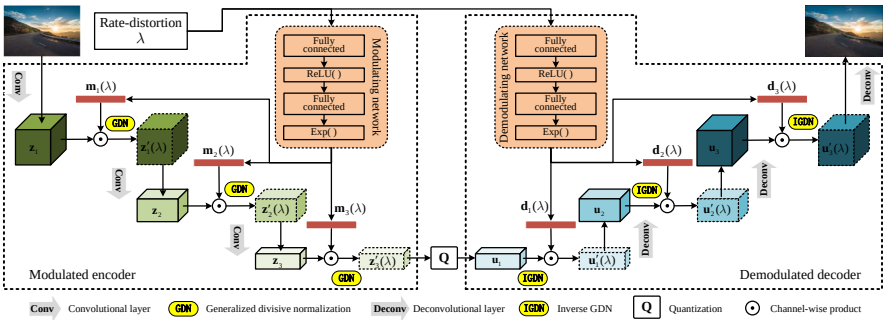


Figure 2.2 – Modulated autoencoder (MAE) architecture, combining modulating networks and shared autoencoder. The channel-wise product is performed before GDN in the encoder and after IGDN in the decoder.

encoder and decoder fixed, optimized for a particular R-D tradeoff (Fig. 2.1(b)).

We observe several limitations in this approach: (1) scaling only the bottleneck features is not flexible enough to adapt to a large range of R-D tradeoffs, (2) using the inverse of the scaling factor in the decoder may also limit the flexibility of the adaptation mechanism, (3) optimizing the parameters of the autoencoder only for a single R-D tradeoff leads to suboptimal parameters for other distant tradeoffs, (4) training the autoencoder and the scaling factors separately may also be limiting. In order to overcome these limitations we propose the *modulated autoencoder (MAE)* framework.

2.3.3 Modulated autoencoders

Variable rate is achieved in MAEs by modulating the internal representations in the encoder and the decoder (Fig. 2.2). Given a set of internal representations in the encoder $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_K\}$ and in the decoder $\mathbf{U} = \{\mathbf{u}_1, \ldots, \mathbf{u}_L\}$, they are replaced by the corresponding modulated and demodulated versions $\mathbf{Z}' = \{\mathbf{z}_1 \odot \mathbf{m}_1(\lambda), \ldots, \mathbf{z}_K \odot \mathbf{m}_K(\lambda)\}$ and $\mathbf{U}' = \{\mathbf{u}_1 \odot \mathbf{d}_1(\lambda), \ldots, \mathbf{u}_L \odot \mathbf{d}_L(\lambda)\}$, where $\mathbf{m}(\lambda) = (\mathbf{m}_1(\lambda), \ldots, \mathbf{m}_K(\lambda))$ and $\mathbf{d}(\lambda) = (\mathbf{d}_1(\lambda), \ldots, \mathbf{d}_L(\lambda))$ are the modulating and demodulating functions.

Our MAE architecture extends the NIC architecture proposed in [12] which combines convolutional layers and GDN/IGDN layers [11]. In our experiments, we choose to modulate the outputs of the convolutional layers in the encoder and decoder, i.e. $\mathbf{Z}$ and $\mathbf{U}$, respectively.

The modulating function $\mathbf{m}(\lambda)$ for the encoder is learned by a *modulating network* as $\mathbf{m}(\lambda) = \mathbf{m}(\lambda; \vartheta)$ and the demodulating function $\mathbf{d}(\lambda)$ by the *demodulating network*

as $\mathbf{d}(\lambda) = d(\lambda;\varphi)$. As a result, the encoder has learnable parameters $\{\theta,\vartheta\}$ and the decoder $\{\phi,\varphi\}$.

Finally, the optimization problem for the MAE is

$$\underset{\theta,\phi,\vartheta,\varphi}{\operatorname{argmin}} \sum_{\lambda\in\Lambda} \left[R(\tilde{\mathbf{z}};\theta,\vartheta,\lambda) + \lambda D\left(\mathbf{x},\hat{\mathbf{x}};\theta,\phi,\vartheta,\varphi,\lambda\right)\right], \tag{2.8}$$

which extends Eq.(2.5) with the modulating/demodulating networks and their corresponding parameters. All parameters are learned jointly using gradient descent and backpropagation.

This mechanism is more flexible than bottleneck scaling since it allows multi-level modulation, decouples encoder and decoder scaling and allows effective joint training of both autoencoder and modulating network, therefore optimizing jointly to all R-D tradeoffs of interest.

2.3.4 Modulating and demodulating networks

The modulating network is a perceptron with two FC layers and ReLU [75] and exponential nonlinearities (Fig. 2.2). The exponential nonlinearity guarantees positive outputs which we found beneficial in training. The input is a scalar value λ and the output is $\mathbf{m}(\lambda) = (\mathbf{m}_1(\lambda),\ldots,\mathbf{m}_K(\lambda))$. A small first hidden layer allows learning a meaningful nonlinear function between tradeoffs and modulation vectors, which is more flexible than simple scaling factors and allows more expressive interpolation between tradeoffs. In practice, we use normalized tradeoffs as $\hat{\lambda}_k = \lambda_k / \max_{\lambda\in\Lambda}(\lambda)$. The demodulating network follows a similar architecture.

2.4 Experiments

2.4.1 Experimental setup

We evaluated MAE on the CLIC 2019 Professional dataset with 585 training images and 226 test images. In addition, we also test our models on Kodak dataset. Our implementation [2] is based on the autoencoder architecture of [12], which is augmented with modulation mechanisms and modulating networks (two FC layers, with 150 and 3×192 units respectively) for all the convolutional layers. We use MSE as distortion metric. The model is trained with crops of size 240×240 using Adam with a minibatch size of 8 and initial learning rates of 0.0004 and 0.002 for MAE and the entropy model, respectively. After 400k iterations, the learning rates are halved for another 150k iterations. We also tested MAEs with scale hyperpriors, as described in [13]. In

[2] https://github.com/FireFYF/modulatedautoencoder

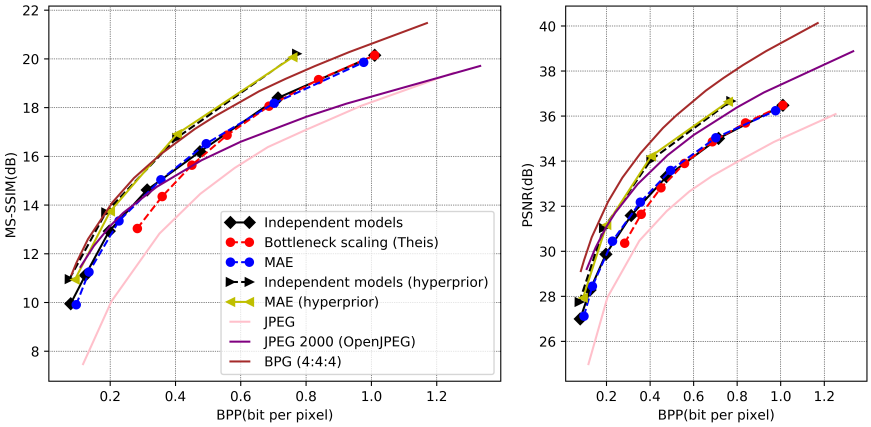


Figure 2.3 – R-D curves for different methods on CLIC 2019 Professional test dataset.

our experiments, we consider seven ($\lambda \in [64, 128, 256, 512, 1024, 2048, 4096]$) and four ($\lambda \in [64, 256, 1024, 4096]$) R-D tradeoffs for the models without and with hyperprior, respectively. We consider two baselines:

Independent models. Each R-D operational point is obtained by training a new model with a different R-D tradeoff λ in (2.2), requiring each model to be stored separately. This provides the optimal R-D performance, but also requires more memory to store all the models for different R-D tradeoffs.

Bottleneck scaling. The autoencoder is optimized for the highest R-D tradeoff in the range. Then it is frozen and the scaling parameters are learned for the other tradeoffs.

2.4.2 Results

We use MS-SSIM (dB)[3] and PSNR (dB) to evaluate image distortion. Fig. 2.3 and Fig. 2.4 show the R-D operational curves for the proposed MAE and the two baselines on the CLIC 2019 Professional dataset and the Kodak dataset, respectively. In addition, it also includes JPEG [107], JPEG2000 [1] and BPG [15] coding methods to demonstrate their performance. For both of them, we can see that the best R-D performance is obtained by using independent models. Hyperprior models also have superior R-D performance. Bottleneck scaling is satisfactory for high bitrates, closer to the optimal R-D operational point of the autoencoder, but degrades for lower bitrates.

[3] MS-SSIM (dB) is computed as $-10\log_{10}(1 - \text{MS-SSIM})$ [48]

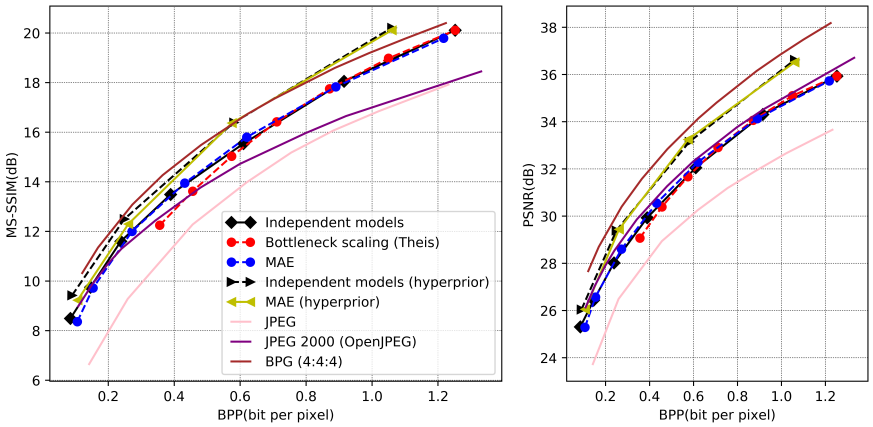


Figure 2.4 – R-D curves for different methods on Kodak dataset.

Table 2.1 – Model size (millions of parameters)

Architecture	w/o hyperprior [12] (seven R-D tradeoffs)	w/ hyperprior [13] (four R-D tradeoffs)
Independent models	28.02 **M**	40.53 **M**
Bottleneck scaling [103]	4.00 **M**	-
Modulated AE (ours)	4.06 **M**	10.27 **M**

Interestingly, bottleneck scaling cannot achieve as low bitrates as independent models since the autoencoder is optimized for a high bitrate. This can be observed in the R-D curve as a narrower range of bitrates. Note that our independent models results did not achieve the same performance as in [12] and [13] due to the different training datasets. The proposed MAEs can achieve an R-D performance very close to the corresponding independent models, demonstrating that multi-layer modulation with joint training is a more powerful mechanism to achieve effective variable rate compression.

The main advantage of bottleneck scaling and MAEs is that the autoencoder is shared, which results in much fewer parameters than independent models, which depend on the number of R-D tradeoffs (Table 2.1). Both methods have a small overhead due to the modulating networks or the scaling factors (which is smaller in bottleneck scaling).

In order to illustrate the differences between the bottleneck scaling and MAE bitrate adaptation mechanisms, we consider the image in Fig. 2.5b and the reconstructions for high and low bitrates in Fig. 2.5a. We show two of the 192 channels in the bottleneck

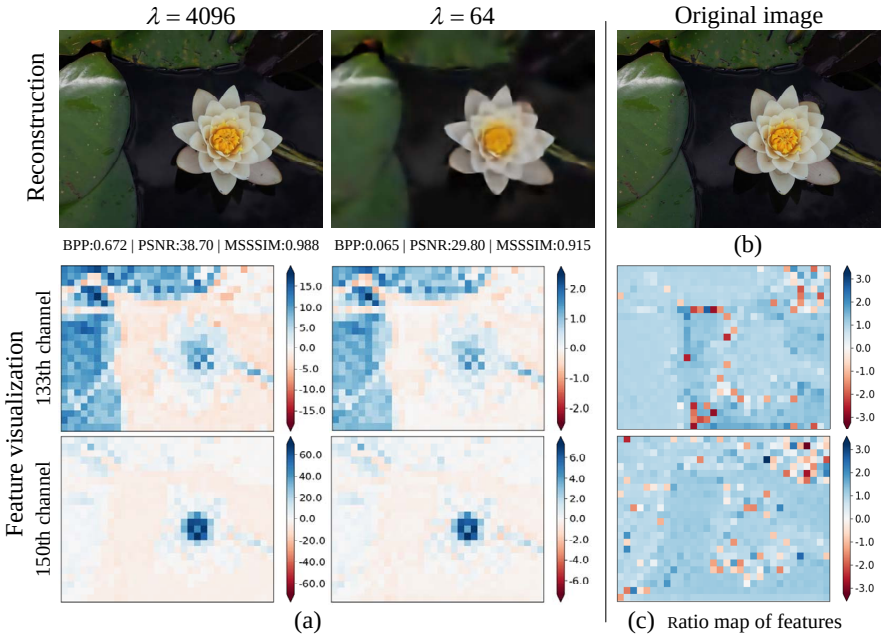


Figure 2.5 – Modulated feature maps: (a) reconstructed images for high ($\lambda = 4096$) and low ($\lambda = 64$) bitrates (first row), and the corresponding feature maps for two channels of the bottleneck before quantization (2nd and 3rd rows), (b) original image for comparison, and (c) element-wise ratio (logarithmic scale) between the feature maps at the two different tradeoffs.

feature before quantization (Fig. 2.5a), and observe that the maps for the two bitrates are similar but the range is higher for $\lambda = 4096$, so the quantization will be finer. This is also what we would expect in bottleneck scaling. However, a closer look highlights the difference between both methods. We also compute the element-wise ratio between the bottleneck features at $\lambda = 4096$ and $\lambda = 64$, and show the ratio image for the same channels of the example (Fig. 2.5c). We can see that the MAE learns to perform a more complex adaptation of the features beyond simple channel-wise bottleneck scaling since different areas of the ratio map show different values (the ratio map would be uniform in bottleneck scaling), which allows MAE to allocate bits more freely when optimizing for different R-D tradeoffs, especially for low bitrates.

2.5 Conclusion

In this chapter, we introduce the modulated autoencoder, a novel variable rate deep image compression framework, based on multi-layer feature modulation and joint learning of autoencoder parameters. MAEs can realize variable bitrate image compression with a single model, while keeping the performance close to the upper bound of independent models that require significantly more memory. We show that MAE outperforms bottleneck scaling [103], especially for low bitrates.

3 Slimmable Compressive Autoencoders for Practical Neural Image Compression[1]

3.1 Introduction

While MAE provides variable rate and prevents having multiple NIC models, the deep encoder and decoder are still heavy modules. As we discussed previously, the challenging problem of practical image compression further includes real-world constraints such as limited memory, computation and latency, often related with their deployment in resource-constrained devices (e.g. mobile phones) and networks. Similarly, video compression addresses the same problem for sequences of images, where low complexity and latency become even more critical [115].

Method	Rate-dist. perform.	Total memory	Variable			Training time
			Rate	Memory	FLOPs	
JPEG,JP2K	Low	Very low	Yes	-	-	-
BPG	High	Low	Yes	-	-	-
Single CAE	Optimal	Medium	No	No	No	Low
Multiple CAEs	Optimal	High	Yes	Yes	Yes	High
BScale[103]	Medium	Medium	Yes	No	No	Low
MAE[122],cAE[25]	High	Medium	Yes	No	No	Low
SlimCAE	Optimal	Medium	Yes	Yes	Yes	Low

Table 3.1 – Comparison of compression methods.

Traditional block-based transform codings methods (e.g. JPEG [107], JPEG2000 [78, 95], BPG [96]) include carefully designed linear transforms and coding tools that consider practical limitations, thus effectively addressing practical image compression. In this paradigm, encoding is block-based and iterative. Rate and distortion are optimized during encoding by exhaustively searching optimal block partitions and coding and prediction modes. Complexity can be controlled by limiting the range of coding and

[1]This chapter is based on a publication at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2021) [121].

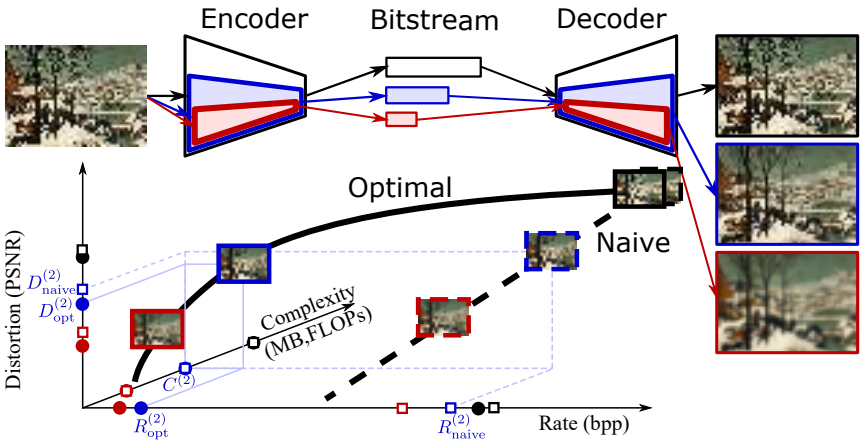


Figure 3.1 – Variable rate and complexity adaptive image compression with a slimmable compressive autoencoder.

prediction modes, or using heuristics. The rate can be estimated from previous blocks, and controlled by adjusting quantization parameters.

In contrast to block-based codecs, NIC approaches are typically image-based and feed-forward, resulting in a constant processing cost. In addition, these heavier deep networks require specialized hardware and still has a high computational cost, making them unattractive in practical resource-limited scenarios.

The challenge of deploying deep neural networks in resource-limited devices (e.g. smartphones, tablets) has motivated research on lightweight architectures [37, 89], integer and binary networks [45, 50, 81] and automatic architecture search [101]. However, only few works have addressed efficiency in NIC [47, 84]. We borrow the idea of slimmable neural networks [126], where the width of the layers (i.e. number of channels) of a classifier can be adjusted to trade off accuracy for computational efficiency.

This chapter describes the slimmable compressive autoencoder (SlimCAE) framework, where we show that the slimming mechanism can enable both variable rate and adaptive complexity (see Fig. 3.1). We propose and study different variants of slimmable generalized divisive normalization [11] (GDN) layers, and slimmable probability models. Naive training of SlimCAEs, with the different subnetworks (i.e. *subCAEs*) optimizing the same loss on all widths, results in suboptimal performance. We crucially observe that each R-D tradeoff has an corresponding minimum capacity. This suggest that, in contrast to other slimmable networks, each width should have

different objectives, i.e. different $D + \lambda R$, determined by the corresponding tradeoff λ. This characteristic makes SlimCAEs more difficult to train, and unlikely to benefit from implicit or explicit distillation [126]. Addressing this problem, we propose λ-scheduling an algorithm that alternates between training the model and adjusting the different λs. Via slimming, SlimCAEs can address the main requirements of *practical neural image compression (PNIC)* in a simple and integrated way.

Our main contributions are: (1) a novel rate and complexity control mechanism via layer widths, motivated by a key insight connecting optimal R-D tradeoffs and capacity; (2) the SlimCAE framework, which enables control of computation, memory and rate, required for PNIC; (3) an efficient training algorithm for SlimCAEs; (4) novel slimmable modules (i.e. GDNs, entropy models). In addition, SlimCAEs can be easily adapted to obtain scalable bitstreams.

3.2 Related work

3.2.1 Variable rate image compression

As we described in the previous chapter, many practical applications require certain control of the target rate. Several NIC approaches can provide variable rate control [22, 25, 104, 122]. However, none of these methods provides explicit control over complexity. Our approach (see Fig. 3.2c), in addition to variable rate, can jointly reduce significantly the memory and computational cost for low rates.

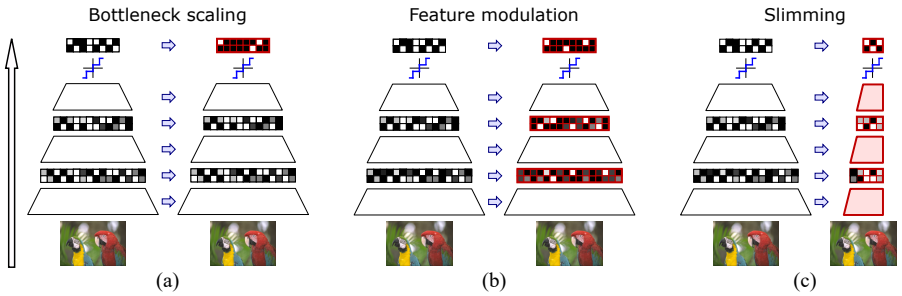


Figure 3.2 – Mechanisms to achieve variable rate in compressive autoencoders: (a) bottleneck scaling [103], (b) feature modulation [25, 122], and (c) proposed method (SlimCAE). Adaptation from high rate (left) to low rate (right). Changes are highlighted in red. Only SlimCAE reduces memory and computation. GDN layers are not included for simplicity.

3.2.2 Efficiency

Lightweight architectures, such as GoogleNet [100] and MobileNet [37, 89], are designed for resource-limited devices by reducing the number of parameters and computation. At the cost of small drop in performance, integer or binary weights can further improve efficiency [45, 50, 81]. Network architecture search (NAS) [8, 101, 134] includes design hyperparameters (e.g. width, number of layers) in the optimization space. Slimmable neural networks [125, 126] enable models that can be run at different accuracy-efficiency tradeoffs. Regarding NIC, Johnston *et al.* [47] use NAS to achieve 2-3× coding speed-up. Cai *et al.* [23] use progressive coding to reduce initial latency, although memory and computational cost remain similar. While tackling run-time or latency, these methods still focus on a single R-D tradeoff, not providing rate, memory nor computation control.

3.3 Slimmable compressive autoencoders

3.3.1 Slimmable autoencoders

The basic structure of an autoencoder (AE) is a learnable encoder $\mathbf{z} = f(\mathbf{x};\theta)$ parametrized by θ that maps an input image $\mathbf{x} \in \mathbb{R}^N$ to a transformed (latent) representation $\mathbf{z} \in \mathbb{R}^D$, followed by a learnable decoder $\hat{\mathbf{x}} = g(\mathbf{z};\phi)$ parametrized by ϕ that maps the latent representation to $\hat{\mathbf{x}} \in \mathbb{R}^N$, with the objective of reconstructing the input image $\mathbf{x}$. Hence the combination of encoder and decoder is autoencoding $\mathbf{x}$, and the objective is to learn the parameters $\psi = (\theta, \phi)$ by minimizing a loss $\mathcal{L}(\theta, \phi; \mathcal{X})$ given a training dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{|\mathcal{X}|}$. The loss measures the reconstruction error, possibly combined with other objectives.

We are interested in AEs whose layers are slimmable [126], i.e. *slimmable autoencoders* (*SlimAEs*), thus enabling dynamic control over the memory and computation costs. An *slimmable layer* allows for discarding part of the layer parameters (in most cases is equivalent to setting them to zero) while still performing a valid operation, trading off expressiveness for lower memory and computational cost. We consider SlimAEs containing K *subautoencoders* (*subAEs*), each of them parametrized by a pair $\psi^{(k)} = (\theta^{(k)}, \phi^{(k)}) \in \Psi = \{(\theta^{(1)}, \phi^{(1)}), \dots, (\theta^{(K)}, \phi^{(K)})\}$, where we assume that $\theta^{(1)} \subset \cdots \subset \theta^{(K)} = \theta$ and $\phi^{(1)} \subset \cdots \subset \phi^{(K)} = \phi$ (we assume these conditions are met for every layer in the SlimAE). Similarly, we can define the loss for the subAE k as $\mathcal{L}^{(k)}(\theta^{(k)}, \phi^{(k)}; \mathcal{X})$, and train the SlimAE with the joint loss or a weighted average $\mathcal{L}(\Psi; \mathcal{X}) = \sum_k \mathcal{L}^{(k)}(\theta^{(k)}, \phi^{(k)}; \mathcal{X})$.

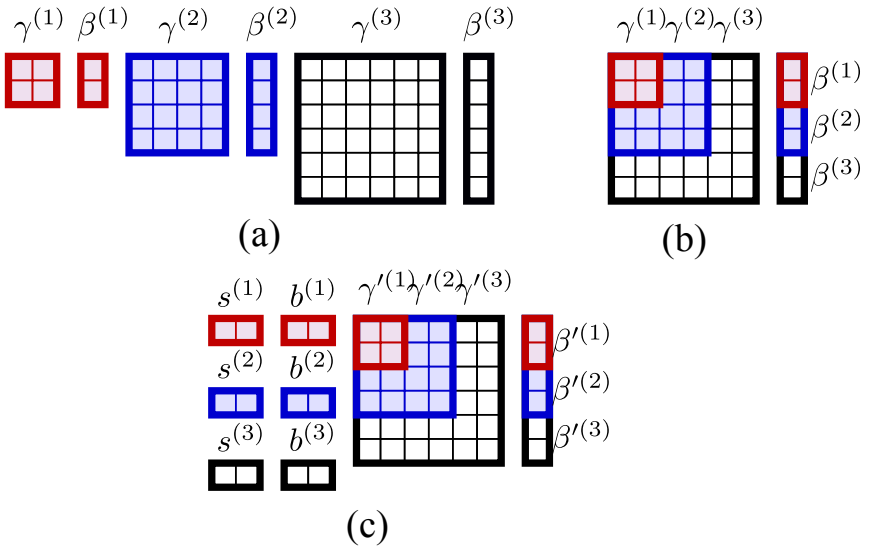


Figure 3.3 – GDN variants: (a) SwitchGDN, (b) SlimGDN, (c) SlimGDN+ (SlimGDN with switch. param. modulation).

3.3.2 Compressive autoencoders

A compressive autoencoder (CAE) is an AE, where the output of the encoder is a binary stream (*bitstream*), typically stored or transmitted through a communications channel. The objective is to maximize the quality of the reconstructed image (i.e. minimize the *distortion*) while minimizing the number of bits transmitted (i.e. minimize the *rate*). CAEs are based on AEs, where the encoder is followed by a quantizer $\mathbf{q} = Q(\mathbf{z})$, where $\mathbf{q} \in \mathbb{Z}^D$ is a discrete-valued symbol vector. A losseless entropy encoder then binarizes and serializes $\mathbf{q}$ into the bitstream $\mathbf{b}$, exploiting its statistical redundancy to achieve code lengths close to its entropy. These operations are reversed in the decoder.

CAEs are typically trained by solving a *rate-distortion optimization (RDO)* problem with loss

$$\mathcal{L}(\theta, \phi; \mathcal{X}, \lambda) = D(\theta, \phi; \mathcal{X}) + \lambda R(\theta; \mathcal{X}), \tag{3.1}$$

where $\mathcal{X}$ is the training dataset, λ is the (fixed) tradeoff between rate and distortion. To allow end-to-end optimization with backpropagation, during training non-differentiable operations, such as quantization and entropy coding, are replaced by differentiable proxies, such as additive noise and entropy estimation.

Without loss of generality, we focus on the CAE framework of Balle *et al.* [12], which combines convolutional layers, generalized divisive normalization (GDN) and inverse GDN (IGDN) layers, scalar quantization to the nearest neighbor (i.e. i.e. $\mathbf{q} = \lfloor \mathbf{z} \rceil$) and arithmetic coding. During training, quantization is replaced by additive uniform noise (i.e. $\tilde{\mathbf{z}} = \mathbf{z} + \Delta\mathbf{z}$, with $\Delta\mathbf{z} \sim \mathcal{U}\left(-\frac{1}{2}, \frac{1}{2}\right)$). Similarly, arithmetic coding is bypassed and rate is approximated by the entropy of the quantized symbol vector $R(\mathbf{b}) \approx H\left[P_{\mathbf{q}}\right] \approx H\left[p_{\tilde{\mathbf{z}}}(\tilde{\mathbf{z}}; \nu)\right]$, where ν are the parameters of the entropy model used in [12]. Distortion is measured as the reconstruction mean square error (MSE), i.e. $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$. The CAE is thus parametrized by $\psi = \left(\theta, \phi, \nu\right)$.

3.3.3 Slimmable CAEs

In order to obtain a *slimmable compressive autoencoder* (*SlimCAE*), all operations in the CAE are required to be non-parametric, slimmable or efficiently switchable. Quantization is non-parametric in our case, and convolutional layers are implemented slimmable [126]. For GDN/IGDN layers, we propose and compare several variants (see next subsection) layers. Finally, we use switchable entropy models, i.e. each subCAE k has its own parameters $\nu^{(k)}$, which can be easily switched since the size is negligible compared to the other parameters $\theta^{(k)}$ or $\phi^{(k)}$.

Our approach can also be extended to more complex frameworks including hyperpriors [13] and autoregressive context models [70].

3.3.4 Switchable and slimmable GDN/IGDN layers

While GDN [10] was proposed to Gaussianize the local joint statistics of natural images, Balle *et al.* [12] proposed an approximate inverse operation (IGDN), and showed that GDN/IGDN layer pairs are highly beneficial in learned image compression, and since then have been adopted by many CAE frameworks. Both GDN and IGDN are parametrized by $\gamma \in \mathbb{R}^{w \times w}$ and $\beta \in \mathbb{R}^{w}$, where w is the number of input (and output) channels.

In the case of a SlimCAE with K subCAEs, the input to the GDN layer has the following possible channel dimensions $w^{(1)}, \ldots, w^{(K)}$. We consider three possible variants:

- **Switchable GDNs**[2]. We use independent sets of parameters $\gamma^{(k)} \in \mathbb{R}^{w^{(k)} \times w^{(k)}}$ and $\beta^{(k)} \in \mathbb{R}^{w^{(k)}}$ for every subGDN k (see Fig. 3.3 a). The normalized represen-

[2] In the following, we omit IGDN for clarity (the same analysis applies).

tation for an input $\mathbf{y}^{(k)} \in \mathbb{R}^{w^{(k)}}$ is then

$$\tilde{y}_i^{(k)} = \frac{y_i^{(k)}}{\left(\beta_i^{(k)} + \sum_j \gamma_{ij}^{(k)} |y_j^{(k)}|^2\right)^{\frac{1}{2}}} \tag{3.2}$$

While flexible, the total number of parameters is relatively high $\sum_{k=1}^{K} \left(w^{(k)}+1\right) w^{(k)}$, and switching may be not very efficient.

- **Slimmable GDN (SlimGDN)**. A more compact option is to reuse parameters from smaller subGDNs by imposing $\gamma^{(1)} \subset \cdots \subset \gamma^{(K)}$ and $\beta^{(1)} \subset \cdots \subset \beta^{(K)}$. Now the total number of parameters in a SlimGDN layer is $\left(M^{(K)}+1\right) w^{(K)}$ (see Fig. 3.3b).

- **SlimGDN with switchable parameter modulation**. SlimGDNs usually performs worse than switchable GDNs, since they are less flexible to adapt to the statistics of the different $\mathbf{y}^{(k)}$. We propose a variant using switchable parameter modulation, where a global scale and bias are learned separately for every subGDN (i.e. switchable), i.e. $\gamma_{ij}^{(k)} = s_\gamma^{(k)} \gamma'^{(k)}_{ij} + b_\gamma^{(k)}$ and $\beta_i^{(k)} = s_\beta^{(k)} \beta'^{(k)}_i + b_\beta^{(k)}$, where $\gamma'^{(k)}$ and $\beta'^{(k)}$ are shared and slimmable and $s_\gamma^{(k)}$, $b_\gamma^{(k)}$, $s_\beta^{(k)}$ and $b_\beta^{(k)}$ are switchable scalars specific for the subGDN k. This variant requires only 4 additional parameters per subGDN (see Fig. 3.3c), for a total number of parameters $\left(w^{(K)}+1\right) w^{(K)} + 4K$.

3.3.5 (Naive) training of SlimCAEs

For the training of SlimCAEs, we can extend Eq. (3.1) and optimize the joint loss of all K subCAEs $\operatorname{argmin}_\psi \sum_{\psi \in \Psi} \mathcal{L}\left(\psi; \mathcal{X}, \lambda\right)$, with parameters $\Psi = \left\{\psi^{(1)}, \ldots, \psi^{(K)}\right\}$. The problem can be solved using stochastic gradient descent (SGD) and backpropagation. We refer to this case as *naive SlimCAE*.

3.4 SlimCAEs with multiple rate-distortion tradeoffs

3.4.1 Rate-distortion and capacity

While a naive SlimCAE can already control the rate of the output bitstream and the complexity of the model, it is limited to a relatively narrow range of rates with suboptimal R-D performance. This can be observed in Fig. 3.4, where we show the R-D curves obtained with independent CAEs with different capacities controlled via the layer width w (i.e. number of filters per convolutional layer) compared with one

SlimCAE with the same layer widths.

Fig. 3.4 shows that there is a limit in the minimum achievable distortion by a CAE (i.e. at high rates), which is in turn related to its capacity (the lower the capacity, the higher the minimum distortion). Additionally, the figure shows that, when the rate is low enough, additional capacity is unnecessary since the curves converge before that point, (i.e. an optimal capacity for every segment of the optimal R-D curve). Note also

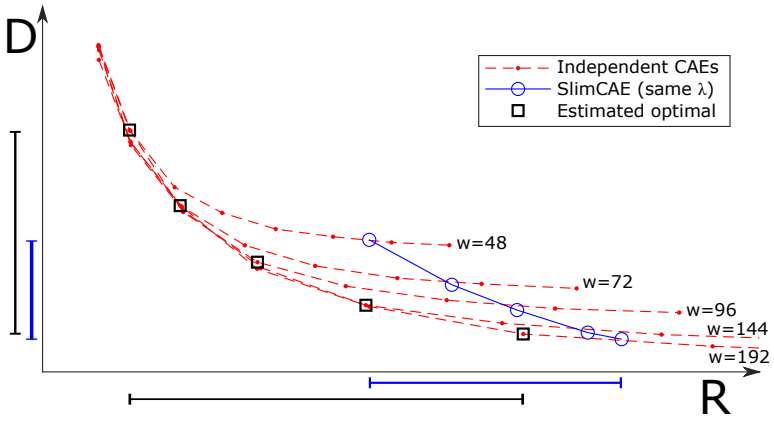


Figure 3.4 – Comparison of R-D curves of independent CAEs and SlimCAE with shared λ. Better choices in the R-D curves are also highlighted.

that the R-D points of the SlimCAE are located over the R-D curves of independent CAEs with the same capacity, suggesting that a slimmable version does not entail R-D penalty. We aim at training the SlimCAE so it can achieve optimal R-D performance at the different capacities. We define the *multi-RD optimization* (MRDO) loss as

$$\mathcal{L}(\Psi;\mathcal{X},\Lambda) = \sum_{k=1}^{K} D\left(\psi^{(k)};\mathcal{X}\right) + \lambda^{(k)} R\left(\psi^{(k)};\mathcal{X}\right), \tag{3.3}$$

where $\Lambda = \{\lambda^{(1)},\dots,\lambda^{(K)}\}$ is a given set of R-D tradeoffs for the different K subCAEs, and the corresponding MRDO problem is $\arg\min_{\Psi} \sum_{\psi^{(k)}\in\Psi} \mathcal{L}(\Psi;\mathcal{X},\Lambda)$.

An important aspect to note is that in this case each subCAE solves a different optimization problem determined by the specific tradeoff $\lambda^{(k)}$. This is an important difference with slimmable networks and SlimAEs in general (including naive SlimCAEs), where every subnetwork or subAE solves exactly the same problem, just with different capacity. This has implications, such as more difficulty to jointly solve all the subproblems and also makes implicit distillation [126] across subCAEs more unlikely.

The problem now is to find appropriate values of Λ and train the SlimCAE.

3.4.2 Estimating optimal λs from R-D curves

One possible way is to leverage the R-D curves of independent CAEs and try to estimate the optimal points to switch to the next subCAE, which is where the curves start diverging because R-D performance saturates for that capacity. We can then estimate $\lambda^{(k)}$ as the slope of the corresponding curve at that optimal point (see Fig. 3.4).

3.4.3 Automatic estimation via λ-scheduling

While knowing in advance the empirical R-D curves leads to better R-D performance and wider rate ranges, it has the important drawback of having very high computation cost, since we need to compute K R-D curves, one for each target capacity, and every curve requires training a number of independent CAE exploring different λs.

MRDO training with λ scheduling In order to address the previous limitation, we propose an effective MRDO training algorithm to automatically estimate Λ without requiring independent CAEs curves (see Alg. 1).

The algorithm is based on progressively varying the values of every $\lambda^{(k)}$ following a predefined schedule. We alternate phases of updating Λ and updating the SlimCAE using SGD. The initial stage is a naive SlimCAE, where λ is set to target high rate and low distortion, which requires full use of the capacity. Once trained, the SlimCAE is already optimized for that full capacity. We fix $\lambda^{(K)}$ and update the remaining $k = 1, \dots, K-1$ as $\lambda_t^{(k)} = \kappa \lambda_{t-1}^{(k)}$ with a factor $\kappa > 1$. Then we update the SlimCAE for another number of iterations, which tends to reduce the rate and moves the R-D of subCAE $K-1$ closer to the optimal R-D curve. Geometrically, this results in the slope of the segment between the R-D points of the two consecutive subCAEs $K-1$ and K decreasing (see Fig. 3.5b and c). When this slope does not decrease anymore, we fix $\lambda^{(K-1)}$ and continue the process recursively. The overall effect of the λ scheduling is to progressively accommodate the target R-D point for each subCAE so training can approximate the optimal R-D points.

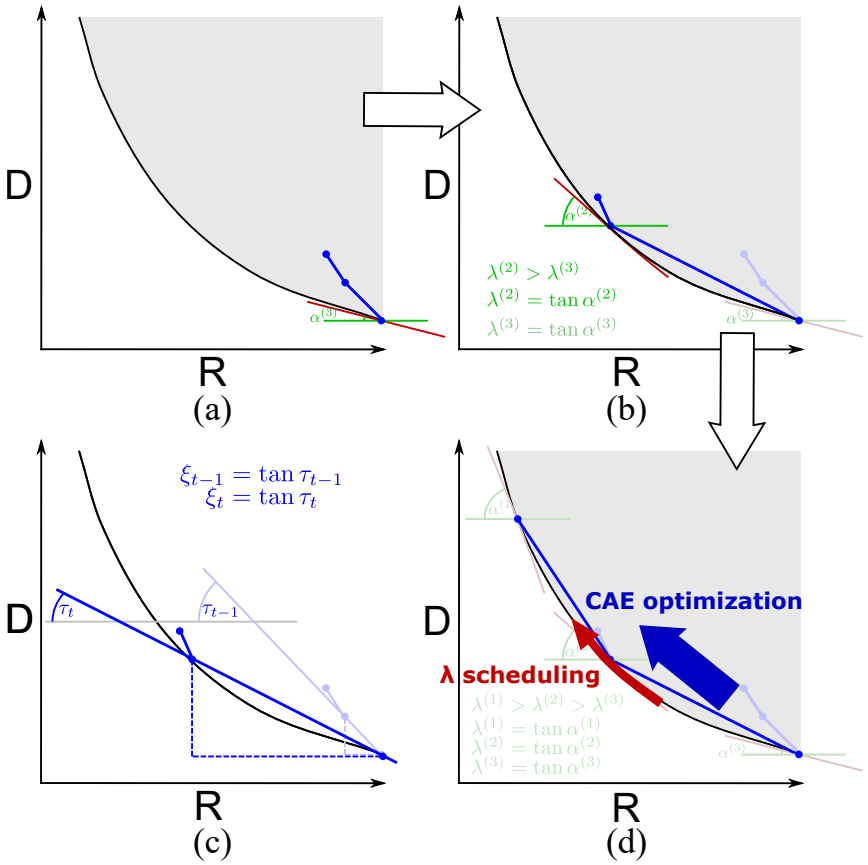


Figure 3.5 – Training SlimCAEs: (a) naive training with a single R-D tradeoff λ leads to small ranges and suboptimal R-D performance, (b) varying the λ progressively for smaller subCAEs changes the target R-D point and stretches the R-D range, (c) the slope ξ of R-D segments is used to monitor convergence of the proposed training algorithm, and (d) illustration of how λ scheduling changes the R-D target and SlimCAE optimization stretches the R-D range.

Algorithm 1 SlimCAE training with λ-scheduling

Input: $\mathcal{X}X_{tr}$, $\mathcal{X}_{val}$ ▷ Training, val. data
Input: SlimCAE with K subCAEs and parameters Ψ
Input: $\lambda^{(K)}$ ▷ R-D tradeoff for largest subCAE
Input: κ, T, M ▷ Other hyperparameters
Output: Ψ, Λ

$\Lambda_0 \leftarrow \left[\lambda^{(K)}, \ldots, \lambda^{(K)}\right]$
$\Psi_0 \leftarrow \arg\min_{\Psi} \mathcal{L}(\Psi; \mathcal{X}_{tr}, \Lambda_0)$ ▷ Naive training
Calculate $R_0^{(K-1)}, R_0^{(K)}, D_0^{(K-1)}, D_0^{(K)}$ over $\mathcal{X}_{val}$
$\xi_0 \leftarrow \frac{D_0^{(K)} - D_0^{(K-1)}}{R_0^{(K)} - R_0^{(K-1)}}$
$t \leftarrow 1$
for $i \leftarrow K-1$ to 1 **do**
  for $m \leftarrow 1$ to M **do**
    $\Lambda_i \leftarrow \left[\kappa\lambda_{i-1}^{(1)}, \ldots, \kappa\lambda_{i-1}^{(i)}, \lambda_{i-1}^{(i+1)}, \ldots, \lambda_{i-1}^{(K)}\right]$
    for $j \leftarrow 1$ to T **do**
      $\Psi_t \leftarrow \arg\min_{\Psi} \mathcal{L}(\Psi_{t-1}; \mathcal{X}_{tr}, \Lambda_i)$
      $t \leftarrow t+1$
    end for
    if $R_t^{(i+1)} < R_t^{(i)}$ **then**
      continue
    else
      Calculate $\xi_t \leftarrow \frac{D_t^{(i+1)} - D_t^{(i)}}{R_t^{(i+1)} - R_t^{(i)}}$ over $\mathcal{X}_{val}$
      if $\xi_t > \xi_{t-T}$ **then**
        break
      end if
    end if
  end for
end for

3.5 Experiments

3.5.1 Experimental settings

We implemented[3] and evaluated the proposed approaches building upon the widely used image compression framework proposed by Balle at al. [12] which typically uses

[3] https://github.com/FireFYF/SlimCAE

layers with a width of 192 filters (encoder: 3 conv, 3 GDN; decoder: 3 deconv, 3 IGDN). To address different complexities and rates, we consider five different widths ($w \in \{48, 72, 96, 144, 192\}$), which also control the total capacity of the model. The models are trained and evaluated on the CLIC dataset[4]. In Section 3.5.6 we evaluate on the Kodak [5] and Tecnick[6] datasets to compare to other methods. We evaluate both R-D performance and efficiency (memory footprint, computational cost and latency).

SlimCAE variants. We consider three GDN/IGDN variants (SwitchGDN, SlimGDN and SlimGDN+ corresponding to Fig. 3.3a, b and c respectively) and three training strategies (naive, estimated λs and λ-scheduling, corresponding to the methods described in Sections 3.3.5, 3.4.2 and 3.4.3). We used a switchable entropy model (i.e. one independent entropy model per width), since the number of parameters is negligible compared to the overall model.

Baselines. We compare SlimCAE to independently trained CAEs for different widths (five models in total following [12]). We also compare to three approaches to provide variable rate in a single model: bottleneck scaling (BScale) [103][7], modulated autoencoders (MAE) [122] and conditonal autoencoders (cAE) [25].

Training details. We use 240×240 pixel crops and a batch size of 8. Some methods are trained in one step, while other require two steps. The former includes independent CAEs, MAE, cAE, SlimCAE with naive training and training with estimated λs. In these cases we use a learning rate of 1e-4 (1e-3 for the entropy model) during 1.2M iterations, and then halve them for an additional 200K iterations. SlimCAE with λ-scheduling uses a SlimCAE after naive training 1.2M iteration, followed by training with λ-scheduling ($\kappa = 0.8$, $T = 2000$ and $M = 7$ in Alg. 1)[8] during 28K iterations and then fine tuned (by halving the learning rates) until a total of 1.5M iterations with the final λs fixed. We measure distortion as MSE during training and as PSNR during λ-scheduling. BScale uses a CAE trained during 1.2M iterations, which is then fixed and scaling parameters are learned during 300K iterations.

3.5.2 Qualitative analysis

SlimCAE can effectively distribute and optimize the capacity in a way that each subCAE can focus on the patterns relevant to its own optimal R-D tradeoff. For example, the first convolutional layer of the smallest subencoder (see Fig. 3.6) contains

[4] https://www.compression.cc/2019/challenge

[5] http://r0k.us/graphics/kodak

[6] https://testimages.org/sampling

[7] Note that [103] actually introduces the term *compressive autoencoder*. s in a general sense, and in our experiments CAE refers to our baseline [12], while BScale denotes the variable rate approach in [103]

[8] We extend the implementation of [12], which optimizes $\lambda D + R$. We adapt λ-scheduling correspondingly.

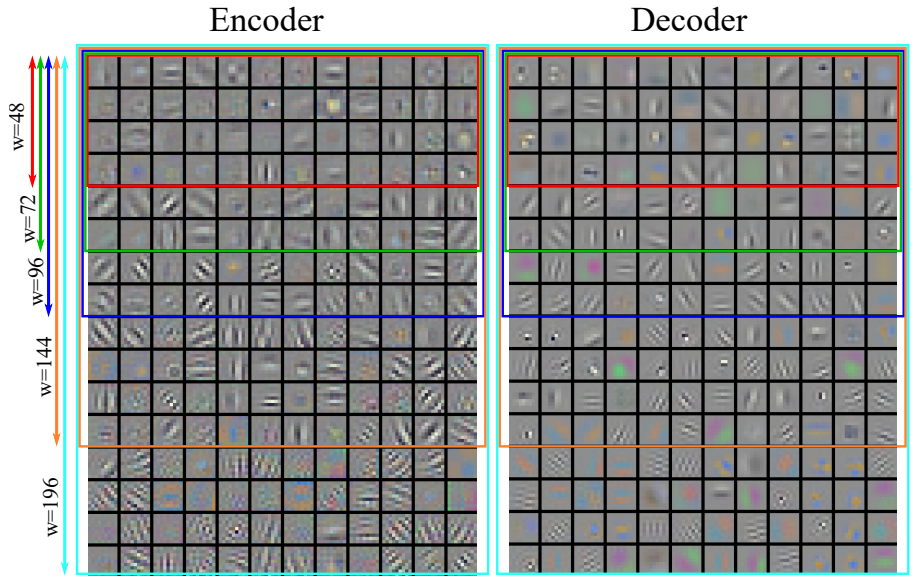


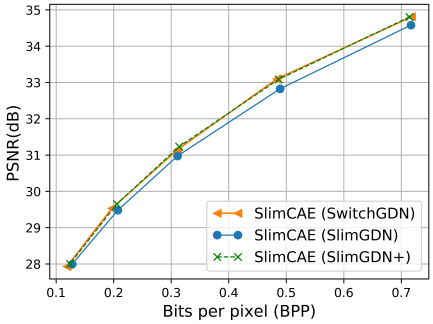
Figure 3.6 – Filters in the first convolutional layer (encoder) and last convolutional layer (decoder) for different widths.

only filters sensitive to low frequency patterns, while larger subencoders progressively include filters related with higher frequency, since they are necessary to achieve lower distortion. The latent representation in the bottleneck is also structured in a similar way from coarse reconstruction to fine details (see Fig. 3.8 a-b in supp. mat.).

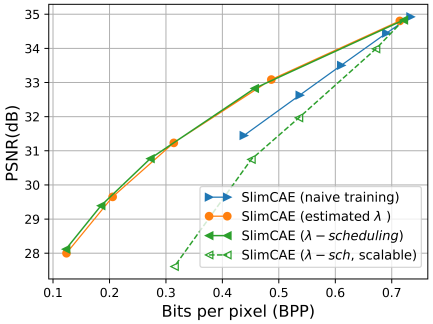
3.5.3 Rate-distortion

Fig. 3.7a shows the rate-distortion performance obtained with different GDN variants. Sharing GDN parameters across different widths (i.e. SlimGDN) results in worse performance than independent ones (i.e. SwitchGDN). However, this loss can be recovered when parameter modulation (i.e. SlimGDN+) at a negligible parameter increase.

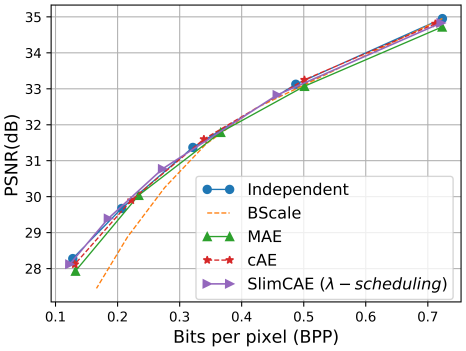
Fig. 3.7b shows that naive training suffers from the limitations of using a single shared tradeoff λ, while training with more adequate width-specific λs (those estimated in Fig. 3.4) results in an R-D curve closer to the obtained with independent CAEs. Fig. 3.7b also illustrates the effect of λ-scheduling in the R-D curve, which gets progressively closer until it essentially achieves the same performance as independent



(a) GDN variants (with estimated λs)



(b) Training strategies and scalability



(c) Variable rate methods

Figure 3.7 – Rate-distortion performance comparison (CLIC dataset).

CAEs (see Fig. 3.9), but without requiring training auxiliary models.

Finally, we compare SlimCAE to other baselines enabling variable rate in a single model. SlimCAE obtains the best R-D performance, overlapping with that of independent CAEs, but with a much lower training and memory cost, as we see next.

3.5.4 Efficiency

We also evaluate the efficiency of SlimCAE in terms of memory footprint (in MB), computational cost (in FLOPs) and latency (in ms)[9]. Values in features and parameters

[9]We only compare to NIC codecs, since traditional codecs run in different hardware (CPU instead of GPU). As reference, our BPG baseline takes on CPU (for 0.1-1.0 bpp) 3.2-4.5 s/img (enc) and 95-131

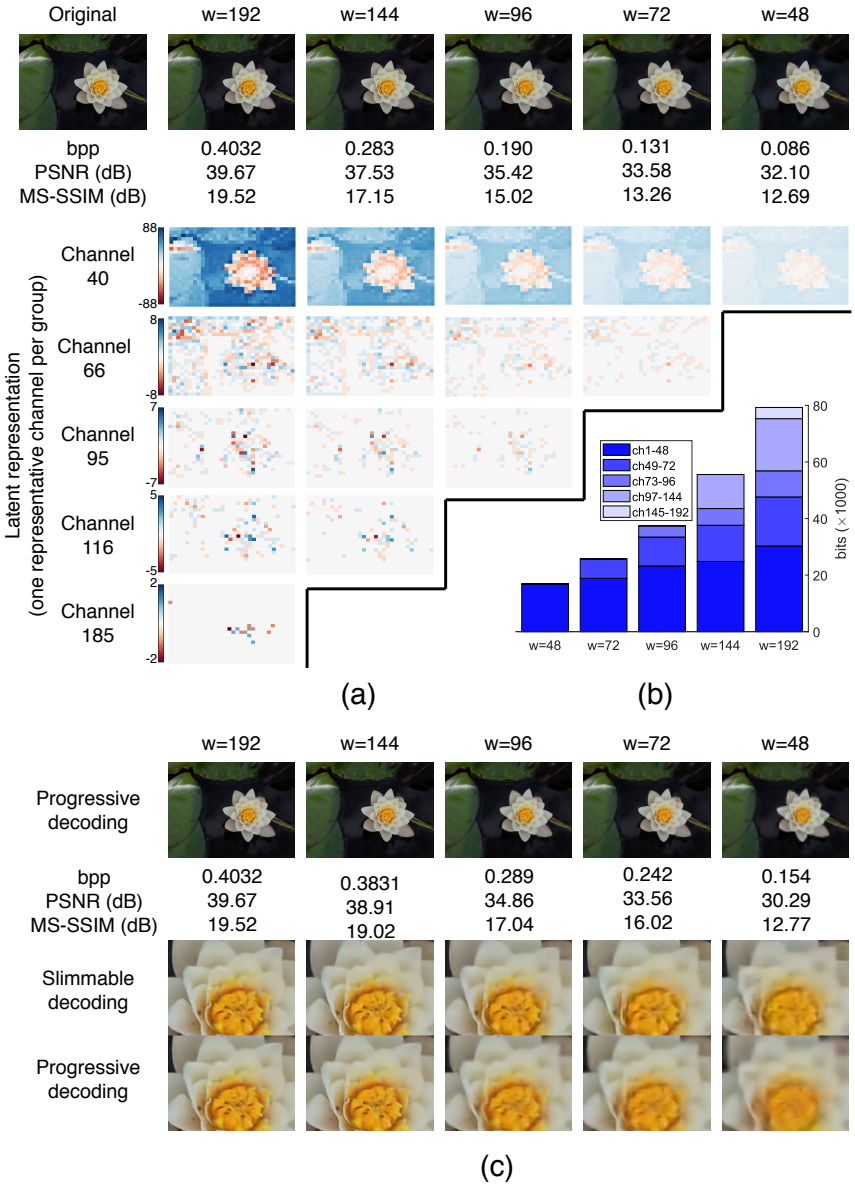


Figure 3.8 – The latent representations.

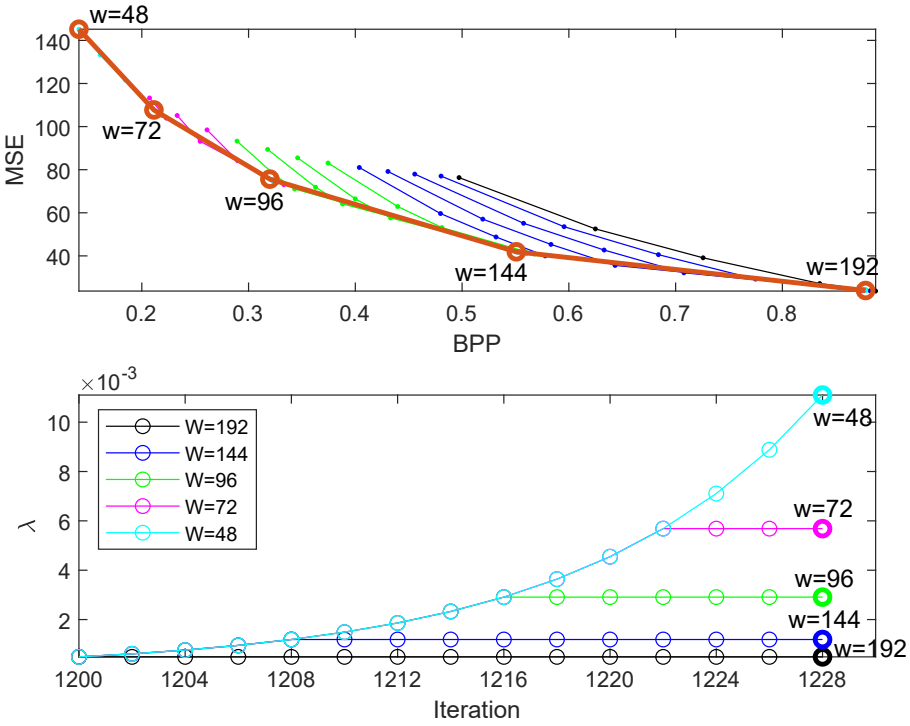


Figure 3.9 – Evolution of the R-D curve (top) and λ during the λ-scheduling phase. Naive training shown in black (top).

are represented with 4 bytes, and the features are calculated for input images of size 768×512 pixels. We consider a baseline with five independent CAEs optimized for different R-D tradeoffs. For fair comparison we use the minimum width that guarantees that the R-D performance at a particular tradeoff λ remains optimal (see Fig. 3.4).

Fig. 3.10 shows the memory footprint in the encoder and decoder at different widths. Features require significantly more memory than model parameters, especially at small widths. While for the largest width all methods require similar memory, an independent CAE and SlimCAE can reduce significantly the memory footprint for small widths. This reduction also results in a significantly lower computational cost (see Table 3.2), and much lower latency during both encoding and decoding (see

ms/img (dec).

Table 3.2 – Computational cost of trained models (millions of FLOPs). Some methods adjust layer widths.

Methods	Low rate	→	Medium rate	→	High rate
Independent	15.34**M** (w=48)	31.69**M** (w=72)	53.81**M** (w=96)	115.53**M** (w=144)	200.28**M** (w=192)
BScale [103]	200.28**M** (w=192)	200.28**M** (w=192)	200.28**M** (w=192)	200.28**M** (w=192)	200.28**M** (w=192)
MAE [122]	200.40**M** (w=192)	200.40**M** (w=192)	200.40**M** (w=192)	200.40**M** (w=192)	200.40**M** (w=192)
cAE [25]	200.31**M** (w=192)	200.31**M** (w=192)	200.31**M** (w=192)	200.31**M** (w=192)	200.31**M** (w=192)
SlimCAE	**15.34M** (w=48)	**31.69M** (w=72)	**53.81M** (w=96)	**115.53M** (w=144)	**200.28M** (w=192)

Table 3.3). Now we consider the total memory required to provide the five different rates. It requires to store the model parameters of the independent CAEs (31.1 MB), in contrast to just a single model for BScale (15.3 MB), MAE (15.7 MB), cAE (15.3 MB) and SlimCAE (15.3 MB with SlimGDN+). If we consider the memory used to store features (note that at only one model is use at a time), the memory footprint of multiple CAEs varies from 42.9 to 78.3 MB, depending on the selected rate, and similarly to SlimCAE (27.1 to 62.5 MB). In contrast, other methods cannot adapt the complexity and remain with a higher and constant footprint (BScale 62.5 MB, MAE 63 MB and cAE 62.6 MB)[10]. The SlimGDN+ layers require 0.85 MB (compared to 1.71 MB and 0.85 MB in SwitchGDN and SlimGDN, respectively).

Finally, Table 3.1 summarizes the main advantages and drawbacks of different methods. SlimCAE is the most complete of them providing variable rate with a single model and controllable memory and computational requirements, while achieving optimal R-D performance. Training and switching between multiple CAEs suffers from a higher memory footprint (that increases with the number of target R-D points), and the much higher cost of training multiple models. The other baselines can adapt rate, but not memory and computational costs, which remain high at low rates.

3.5.5 Scalable bitstreams

Motivated by SlimCAE's structured latent respresentation, we consider a variant where each group of channels are encoded independently, allowing *quality scalability* and

[10]Note that, in practice, an optimized implementation could save some memory by discarding intermediate features once they are processed.

Table 3.3 – Encoding and decoding latency (ms) for a 768×512 input image (i.e. batch size 1) on a NVIDIA GTX 1080Ti GPU (excluding data loading/writing and arithmetic coding.

	Methods	Low rate	→	Medium rate	→	High rate
Encoding	Independent	1.9 ± 0.19	2.2 ± 0.17	2.8 ± 0.22	4.0 ± 0.19	5.1 ± 0.20
	BScale [103]	5.2 ± 0.11	5.2 ± 0.16	5.2 ± 0.22	5.2 ± 0.15	5.2 ± 0.13
	MAE [122]	5.4 ± 0.20	5.4 ± 0.20	5.4 ± 0.13	5.4 ± 0.10	5.4 ± 0.11
	cAE [25]	5.5 ± 0.18	5.5 ± 0.11	5.5 ± 0.14	5.5 ± 0.21	5.5 ± 0.28
	SlimCAE	**1.9 ± 0.15**	**2.2 ± 0.27**	**2.8 ± 0.12**	**4.0 ± 0.17**	**5.1 ± 0.10**
Decoding	Independent	2.9 ± 0.20	3.5 ± 0.10	4.3 ± 0.07	6.1 ± 0.07	8.0 ± 0.13
	BScale [103]	8.0 ± 0.21	8.0 ± 0.13	8.0 ± 0.18	8.0 ± 0.11	8.0 ± 0.13
	MAE [122]	8.4 ± 0.15	8.4 ± 0.13	8.4 ± 0.08	8.4 ± 0.07	8.4 ± 0.14
	cAE [25]	8.5 ± 0.20	8.5 ± 0.07	8.5 ± 0.09	8.5 ± 0.13	8.5 ± 0.21
	SlimCAE	**2.9 ± 0.10**	**3.5 ± 0.07**	**4.3 ± 0.10**	**6.1 ± 0.16**	**8.0 ± 0.13**

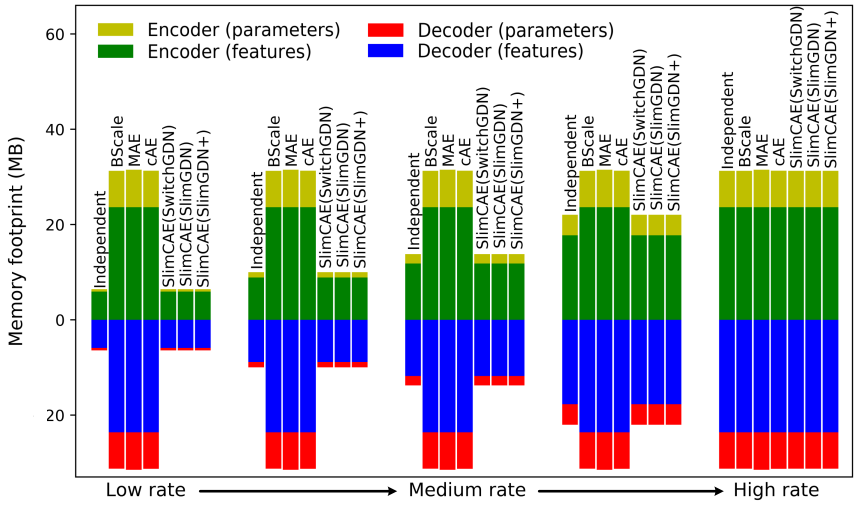


Figure 3.10 – Memory footprint comparison for different rates.

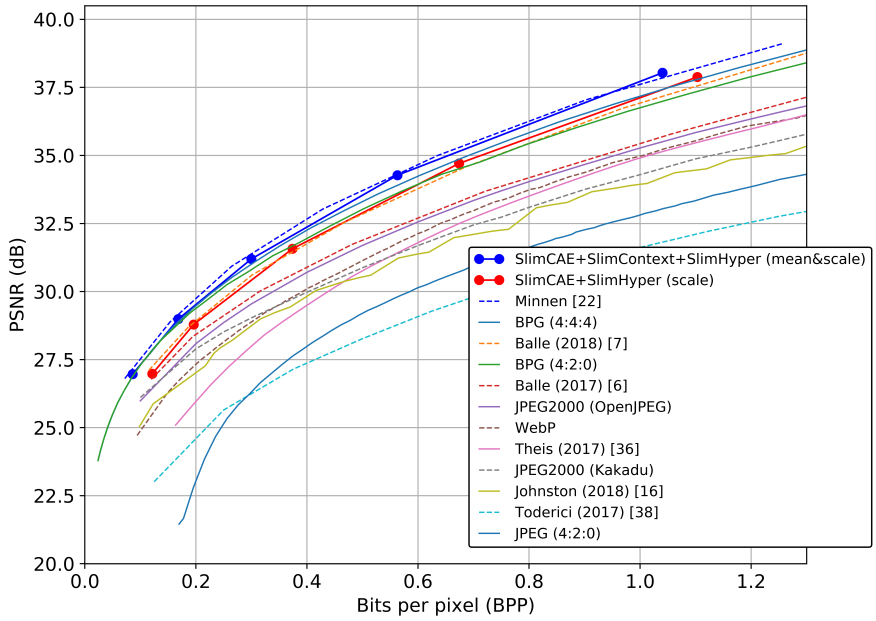


Figure 3.11 – Rate-distortion performance of SlimCAE with slimmable entropy model (Kodak dataset).

progressive decoding. The resulting bitstreams (i.e. base[+ enhancement stream(s)]) are all decodable by the SlimCAE decoder. On the other hand, the SlimCAE is no longer slimmable, so enabling quality scalability disables memory and computation scalability since the SlimCAE is no longer slimmable, and also has certain penalty in R-D performance (see Fig. 3.7b and Fig. 3.8c), a usual compromise in scalable image and video coding [79, 90, 102].

3.5.6 Slimmable entropy models

Our approach is general and can be easily extended including slimmable versions of entropy models to achieve state-of-the-art R-D performance. Following [13, 70], we train[11] a larger capacity autoencoder[12] with a three conv layer slimmable hy-

[11] 3M iter., batch 8, 256×256 crops, λ-sched ($\kappa = 0.8, T = 10^4, M = 7$).
[12] $w \in \{96, 144, 192, 288, 384\}$

Table 3.4 – BD-rate (%) over BPG. Lower means better.

Dataset	PSNR (opt. for MSE)				MS-SSIM (opt. for MS-SSIM)			
	[7]	[23]	Slim[7]	Slim[23]	[7]	[23]	Slim[7]	Slim[23]
Kodak	9.68	-8.94	9.52	-6.17	-41.46	-46.92	-41.41	-47.88
Tecnick	2.95	-11.77	5.23	-10.43	-41.50	-43.24	-40.34	-48.94

perprior[13] [13] and conditional convolutions [25]. We also include a slimmable autoregressive context model[14] [70]. We trained these models[15] and evaluated on Kodak and Tecnik datasets. Fig. 3.11 shows that our approach can be integrated with almost no penalty in R-D performance, while providing the aforementioned advantages in terms of rate and complexity control of SlimCAEs. The same conclusions hold when optimizing MS-SSIM, keeping a significant gain over BPG (see Table 3.4).

3.6 Conclusion

While providing excellent R-D performance, current NIC approaches are also resource demanding, and usually tied to a particular rate, which limits their application in practice. In this chapter we consider the important practical problem of complexity in NIC, both in terms of memory and computational cost. The latter also can significant impact on the latency, which is often an important practical concern.

we propose the SlimCAE framework, a novel approach towards practical and adaptive neural image compression, combining in a single model important functionalities, such as excellent rate-distortion performance, low and dynamically adjustable memory footprint, computational cost and latency, all of them easily controlled via a lightweight slimming mechanism. This makes our approach attractive to resource-limited devices (e.g. smartphones), when rate and computation needs to be controlled dynamically (e.g. video coding, multi-tasking) or to deploy different models to heterogeneous devices, adapted to their computational capabilities. SlimCAE can also generate scalable bitstreams, which can be useful in streaming and broadcasting scenarios with heterogeneous devices.

Finally, this chapter also studies the fundamental connection between rate-distortion performance and model capacity, and propose an efficient and effective approach to train the slimmable model in a single pass.

[13] $w \in \{48, 72, 96, 144, 192\}$, and Leaky ReLUs (same in decoder)

[14] One masked conv layer with $w \in \{96, 144, 192, 288, 384\}$

[15] On CLIC extended with 20k high quality images from flickr.com

4 A Novel Framework for Image-to-image Translation and Image Compression[1]

4.1 Introduction

Modern computer vision and image processing approaches heavily rely on deep neural networks and machine learning. One prominent example is image-to-image translation (I2I) addresses the problem of learning to transform images from a source domain (source) to a target domain. This has numerous applications in image restoration and enhancement (e.g. colorisation, superresolution, deblurring), but also more complex data-driven transformations (e.g. style transfer, face attribute modification, scene synthesis, zebra-to-horse). a transforming one image in one domain into another one, where the transformation is learned from data. A image transformation from a source generating a new image based on an input image, which usually occurs between two different domains (source and target). More recently, image and video coding paradigms are starting to shift towards neural image compression (NIC) approaches, competing and often surpassing the rate-distortion performance of traditional transform coding approaches (e.g. BPG [15]). This technology has also significant implications in visual communications and storage and distribution of video content.

In this chapter we study the problem of *distributed I2I translation*, where the encoding is performed at the sender side, and the decoding at the receiver side, and the coded representation is either transmitted through a communications channel or stored. Thus, in addition to addressing the translation problem, we also aim at obtaining compact binary representations (i.e. bitstreams).

A naive approach to this problem would be translating the image before compressing it at the sender side, or translating the reconstructed image at the receiver side. These approaches have several limitations. First, they require encoding and decoding images twice, once with the translator and once with the image codec, resulting in lower computational efficiency. Similarly it requires storing two separate codecs (i.e. for translation and autoencoding). Finally, each encoding and decoding pass is a lossy transformation, therefore it is likely that through these four transformations more information is lost, resulting in lower quality in the translation with artifacts, and/or larger bitstreams. In order to address this limitations, we propose the I2Icodec

[1]

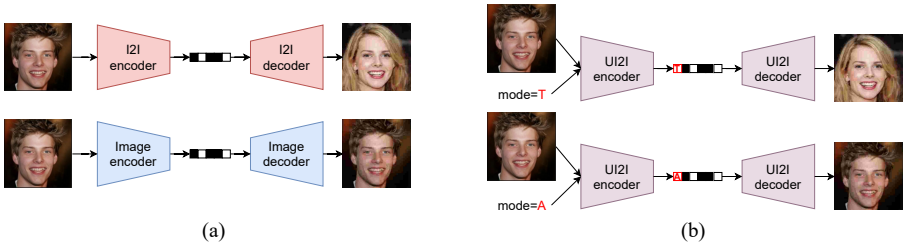


Figure 4.1 – Proposed approaches: (a) distributed I2I translation (*I2Icodec*), alongside a regular image compression codec, and (b) unified translation and autoencoding framework using a single codec (*UI2Icodec*).

framework (see Fig. 4.1a), which addresses distributed I2I translation with a single encoder and decoder, thus avoiding computational overheads and potential loss of information.

While I2Icodec only requires a single encoder and decoder pair, it cannot perform regular autoencoding (i.e. conventional image coding), which is an important functionality. A naive solution is to deploy a regular image codec alongside, but that increases the memory requirements significantly. Thus, to avoid deploying two separate models, we also propose UI2Icodec, a unified framework that can perform compression for distributed translation and autoencoding in a single model (see Fig. 4.1b).

In summary, our contributions are: 1) we study the problem of distributed I2I translation, which involves I2I translation under rate constraints; 2) a novel framework for distributed I2I translation (I2Icodec); and 3) a unified framework for distributed I2I translation and autoencoding (UI2Icodec).

4.2 Related work

Image-to-image translation has been studied widely in recent years. Paired I2I translation [30, 44, 108, 109, 132] assumes the availability of input-output image pairs. Unpaired I2I translation [28, 51, 59, 65, 77, 124, 131] is a more challenging setting where translations are learned from (unpaired) images from the input and output domain.

Often, a given input image can have multiple plausible translations (e.g. colorization). Multimodal I2I translation (or diverse I2I translation) [5, 41, 54, 86, 127] addresses this problem by disentangling content and style. Style is sampled randomly, ensuring the model generate diverse translations. Early approaches assume only two domains. More recently, multi-domain I2I translation approaches [26, 55, 127] can

translate between a range of domains using a single model. In particular, we build upon on StarGAn v2 [26] which provides state-of-the-art translation, including multimodal and multi-domain translation, with content-style disentanglement.

In all I2I translation approaches, encoder and decoder are integrated in a single model, and therefore the latent representation is continuous valued. In this chapter we study the case where this latent representation is transmitted via a digital communication channel, and decoded remotely by the receiver side. This brings the challenges of integrating quantization and entropy coding into the I2I translation framework.

4.3 Distributed image-to-image translation

We first consider the problem of distributed I2I translation to a target domain. In particular, an image $\mathbf{x} \in \mathcal{X}$ is encoded into a compressed bitstream $\mathbf{b}$ at the sender side. The receiver side then decodes $\mathbf{b}$, reconstructing the translated image in a target domain, indicated by the label $y \in \mathcal{Y}$. Following the common practice of disentangling content and style, the bitstream $\mathbf{b}$ captures the content component, while the style of the translated image is determined by a style vector $\mathbf{s}$, either sampled randomly or obtained from a reference image. This disentanglement enables the reconstruction of diverse translations for a given image $\mathbf{x}$.

The objective in distributed I2I translation is to obtain successful translations with compact bitstreams.

4.3.1 I2Icodec framework

Our framework is based on the I2I translation framework of [26], but with the encoder and decoder located separately in the sender and receiver sides, respectively. The framework is augmented with compression capabilities, i.e. quantization and entropy coding. It is composed of *content encoder* E^c, *style encoder* E^s, *mapping network* M, *decoder* G and *discriminator* D.

Content encoder The content encoder E^c extracts a latent representation $\mathbf{z} = E^c(\mathbf{x})$ of the content of the image $\mathbf{x}$. To transmit via a binary channel, the representation $\mathbf{z}$ is quantized (in this paper we use scalar quantization) as $\mathbf{q} = Q(\mathbf{z})$ to obtain a discrete-valued representation $\mathbf{q} \in \mathbb{Z}^D$. Then, $\mathbf{q}$ is binarized using entropy coding (arithmetic coding in our case) to reduce statistical redundancy. Quantization is lossy, but entropy coding is not. During training we replace quantization by uniform noise, and bypass entropy coding, approximating the rate by the entropy of $\mathbf{z}$.

Mapping network and style encoder. The style $\mathbf{s}$ is used for guiding I2I translation towards a specific style in the target domain. This style code can either be sampled randomly (providing diversity), or obtained from a reference style image. In the

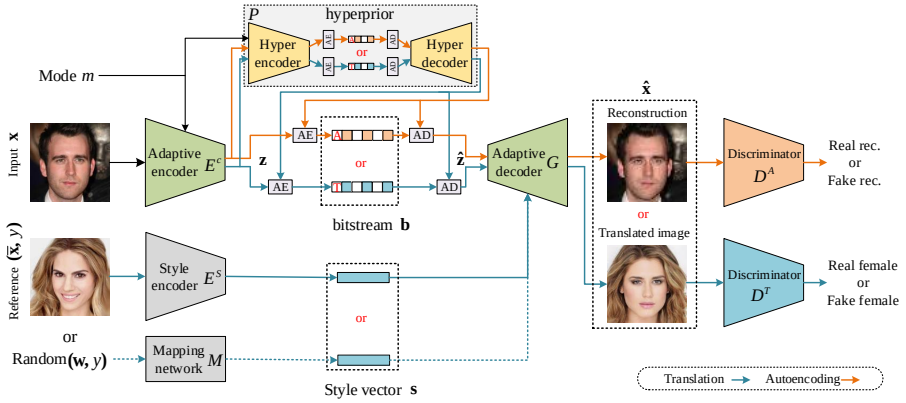


Figure 4.2 – Architecture of the unified framework for regular image compression and I2I translation (*UI2Icodec*). The mode m signals whether the model runs as autoencoder or as I2I translator (store as 1 bit in the bitstream). The simplified framework without m and D^A is referred to as *I2Icodec* .

former, the mapping network M obtains the domain-specific style representation $\mathbf{s}$ from a domain-independent random style $\mathbf{w}$ as $\mathbf{s} = M(\mathbf{w}, y)$. Alternatively, the domain-specific style representation can be obtained from a reference image $\tilde{\mathbf{x}}$ with the style encoder E^s as $\mathbf{s} = E^s(\tilde{\mathbf{x}}, y)$.

Decoder. The decoder G receives the bitstream and performs entropy decoding and maps back to the real-valued representation $\hat{\mathbf{z}}$. It then generates the reconstructed image from $\mathbf{z}$ and the style $\mathbf{s}$ as $\hat{\mathbf{x}} = G(\mathbf{z}, \mathbf{s})$.

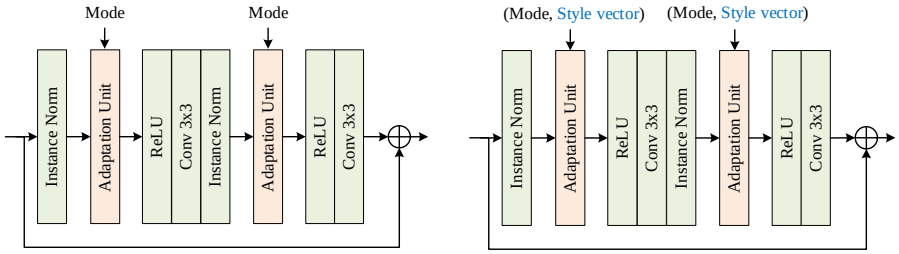
Discriminator. Following [26], we use a multi-task discriminator where $D(\mathbf{x}, y)$ returns the probability that $\mathbf{x}$ is classified in domain y, for every domain.

Entropy model. We use a learnable hyperprior [13] to model the latent distribution $P(\mathbf{z})$.

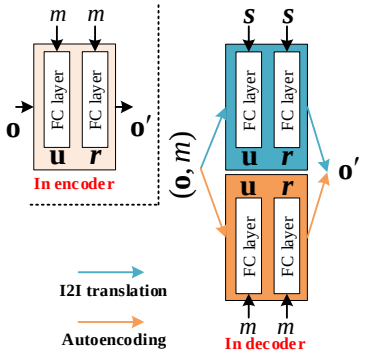
4.3.2 Loss

Our objective during training is to optimize translation while minimizing rate. Regarding translation, we following the losses used in [26]. Given an image $\mathbf{x} \in \mathcal{X}$ and its original domain label $y \in \mathcal{Y}$, we can obtain its latent representation $\mathbf{z} = E^c(\mathbf{x})$ the loss $\mathcal{L}_T$ consists of the following terms.

Adversarial loss. We first generate a random domain-specific target style as $\tilde{s} = M(\mathbf{w}, \tilde{y})$ from a random domain-independent style $\mathbf{w}$ and random target domain $\tilde{y} \in \mathcal{Y}$.



(a) AdaResBlock in adaptive encoder E (b) AdaResBlock in adaptive decoder G



(c) Adaptation units in E and G

Figure 4.3 – (a-b): Adaptive ResBlock for switching between I2I translation and autoencoding; (c) Details of adaptation units in encoder and decoder.

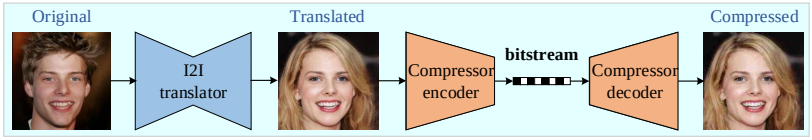
The decoder then synthesizes the translated image as $G(\mathbf{x}, \tilde{s})$. We employ adversarial loss [31] to distinguish the generated images from the real images

$$\mathcal{L}_{\text{adv}} = \mathbb{E}_{\mathbf{x},y}\left[\log D\left(\mathbf{x}, y\right)\right] + \mathbb{E}_{\mathbf{x},\tilde{y},\mathbf{w}}\left[\log\left(1 - D\left(G\left(\mathbf{z}, \tilde{s}\right), \tilde{y}\right)\right)\right], \tag{4.1}$$

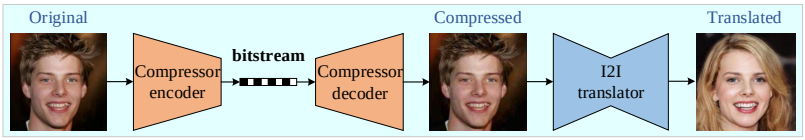
Style reconstruction. We encourage the decoder to optimize the style representation $\tilde{s}$ when generating the image $G(\mathbf{x}, \tilde{s})$ with a style reconstruction loss

$$\mathcal{L}_{\text{sty}} = \mathbb{E}_{\mathbf{x},\tilde{y},\mathbf{w}}\left[\left\|\tilde{s} - E\left(G\left(\mathbf{z}, \tilde{s}\right), \tilde{y}\right)\right\|_1\right]. \tag{4.2}$$

Style diversification. To encourage diversity and prevent mode collapse, we sample and map random pairs of styles $\tilde{s}_1 = M\left(\mathbf{w}_1, \tilde{y}\right)$ and $\tilde{s}_2 = M\left(\mathbf{w}_2, \tilde{y}\right)$, using diversity



(a) Compression after translation (CaT)



(b) Compression before translation (CbT)

Figure 4.4 – Baselines for distributed I2I translation.

sensitive loss

$$\mathcal{L}_{\text{ds}} = \mathbb{E}_{\mathbf{x},\tilde{y},\mathbf{w_1},\mathbf{w_2}} \left[\| G(\mathbf{z}, \tilde{s}_1) - G(\mathbf{z}, \tilde{s}_2) \|_1 \right]. \tag{4.3}$$

Cycle consistency. To ensure that the domain-invariant structure of the input image $\mathbf{x}$ in the translated image $G(\mathbf{z}, \tilde{s})$ is preserved we use the cycle consistency mechanism [131]

$$\mathcal{L}_{\text{cyc}} = \mathbb{E}_{\mathbf{x},y,\tilde{y},\mathbf{w}} \left[\| \mathbf{x} - G(E(G(\mathbf{z}, \tilde{s})), \hat{s}) \|_1 \right], \tag{4.4}$$

where $\hat{s} = E^s(\mathbf{x}, y)$ is the style of the input image.

Rate. We estimate the rate as the entropy of the bitstream via modeling the distribution of $\mathbf{z}$ using the entropy model P. This term encourages the model to retain important in a compact representation

$$\mathcal{L}_{\text{rate}} = \mathbb{E}_{\mathbf{x}} \left[-\log(P(\mathbf{z})) \right] \tag{4.5}$$

using $\mathbf{z} = E^c(\mathbf{x})$. The final loss is $\mathcal{L}_T = \mathcal{L}_{\text{adv}} + \gamma_{\text{sty}} \mathcal{L}_{\text{sty}} - \gamma_{\text{ds}} \mathcal{L}_{\text{ds}} + \gamma_{\text{cyc}} \mathcal{L}_{\text{cyc}} + \lambda_T \mathcal{L}_{\text{rate}}$.

4.4 Unified translation and autoencoding

While the I2Icodec framework can realize distributed I2I translation, being able to recover the original input image (i.e. regular image compression) is equally important in practice. In order to avoid having to deploy two independent codecs (i.e. I2Icodec and autoencoding codec), here we propose a unified framework to transmit an input

image and recover either a reconstruction of the original image (autoencoding mode) or a translated image (translation mode), in a single model.

4.4.1 UI2Icodec framework

As shown in Fig. 4.2, we endow the I2Icodec with a switching mechanism controlled via the mode input $m \in \{\mathrm{A}, \mathrm{T}\}$, which signals the operating mode. In the following we describe the additional modifications to the I2Icodec framework to implement the joint functionality.

Conditional encoder and decoder. The content encoder $E^c(\mathbf{x}; m)$ and the decoder $G(\mathbf{z}, \mathbf{s}; m)$ are conditioned on the mode m. When $m = T$, the encoder and decoder operate exactly as the I2Icodec described earlier.

Adaptive residual blocks (AdaResBlocks). The switching functionality is implemented by conditioning the residual blocks of the content encoder and decoder on the mode m, via an adaptation unit that modulates intermediate features within the residual block (see Fig. 4.3 (a) and (b)). As shown in Fig. 4.3 (c), adaptation units of the content encoder modulate a given input feature $\mathbf{o}$ as $\mathbf{o}' = \mathbf{u}(m) \odot \mathbf{o} + \mathbf{r}(m)$. The adaptation units in the decoder implement $\mathbf{o}' = \mathbf{u}(\mathbf{s}; m) \odot \mathbf{o} + \mathbf{r}(\mathbf{s}; m)$. In both cases, scale and bias parameters themselves are obtained via linear functions of m (and $\mathbf{s}$ in the decoder).

Conditional entropy model. We condition the hyperprior on the mode, i.e. $P(\mathbf{z}; m)$, and one underlying factorized model for translation and another for autoencoding, selected depending on m.

Task-specific discriminators. We use a separate discriminator when optimizing the autoencoding task. We found this more effective than using a shared discriminator for both tasks.

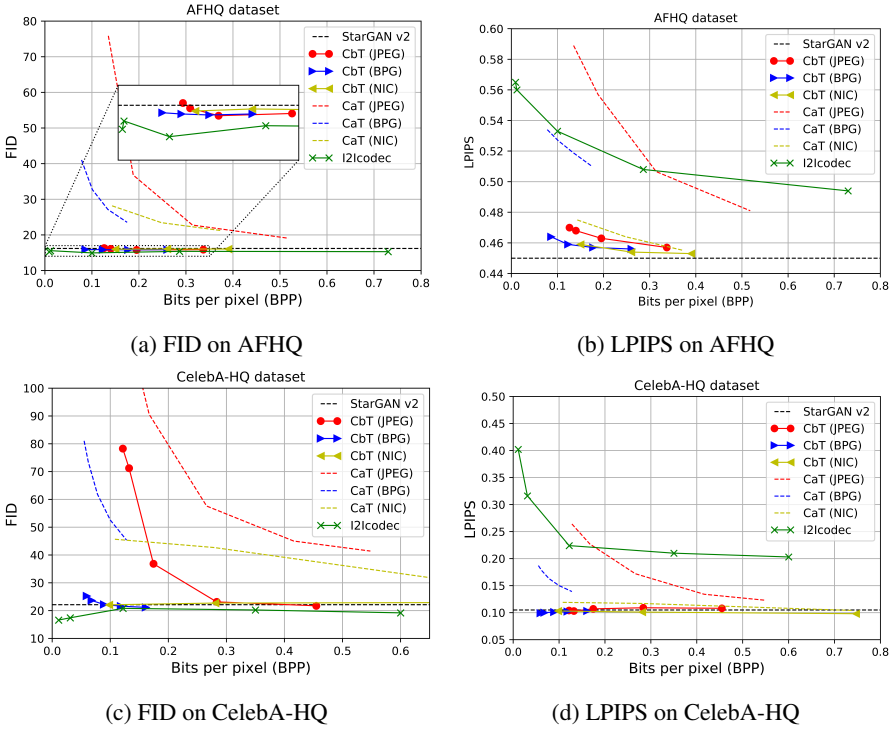
4.4.2 Losses

During training we optimize a loss with two terms corresponding to each of the operating modes

$$\mathcal{L} = [m = \mathrm{A}]\,\mathcal{L}_\mathrm{A} + [m = \mathrm{T}]\,\mathcal{L}_\mathrm{T}, \tag{4.6}$$

where $[P]$ is the Iverson bracket (1 when P is true, 0 otherwise), $\mathcal{L}_\mathrm{T}$ is the loss described in the previous section minimized when $m = \mathrm{T}$, and $\mathcal{L}_\mathrm{A}$ is the autoencoding loss, minimized for $m = \mathrm{A}$. The latter combines three terms $\mathcal{L}_T = \mathcal{L}_\mathrm{RD} + \beta\mathcal{L}_\mathrm{adv2}$

Rate-distortion . In autoencoding we optimize a combination of rate and distortion,



(a) FID on AFHQ

(b) LPIPS on AFHQ

(c) FID on CelebA-HQ

(d) LPIPS on CelebA-HQ

Figure 4.5 – Results of different schemes on AFHQ and CelebA-HQ dataset.

where the tradeoff is controlled by the parameter λ_A. The loss is

$$\mathcal{L}_{\text{RD}} = \mathbb{E}_{\mathbf{x}} \left[d\left(\mathbf{x}, \hat{\mathbf{x}}\right) - \lambda_A \log\left(P\left(\mathbf{z}\right)\right) \right] \tag{4.7}$$

where $\mathbf{z} = E(\mathbf{x}; m = \text{A})$ is the latent representation, $\hat{\mathbf{x}} = G(\mathbf{z}; m = \text{A})$ is the reconstructed image, and $d\left(\mathbf{x}, \hat{\mathbf{x}}\right)$ is the distortion metric (mean-squared error in our case).

Adversarial loss. Similarly to $\mathcal{L}_{\text{adv}}$, we encourage realism in the reconstructed images using

$$\mathcal{L}_{\text{adv2}} = \mathbb{E}_{(\mathbf{x},y)} \left[\log D^A\left(\mathbf{x}, y\right) \right] + \mathbb{E}_{(\mathbf{x},y)} \left[\log\left(1 - D^A\left(\hat{\mathbf{x}}, y\right)\right) \right], \tag{4.8}$$

Method	Latent-guided synthesis				Reference-guided synthesis			
	CelebA-HQ		AFHQ		CelebA-HQ		AFHQ	
	FID↓	LPIPS ↑	FID↓	LPIPS↑	FID↓	LPIPS↑	FID ↓	LPIPS ↑
MUNIT [41]	31.4	0.363	41.5	0.511	107.1	0.176	223.9	0.199
DRIT [54]	52.1	0.178	95.6	0.326	53.3	0.311	114.8	0.156
MSGAN [62]	33.1	0.389	61.4	0.517	39.6	0.312	69.8	0.375
StarGANv2 [26]	22.1*	0.115*	16.2	0.450	23.3*	0.209*	19.8	0.432
I2Icodec ($\lambda_T = 0.5$)	16.6	0.402	15.2	0.565	21.0	0.354	20.6	0.526
I2Icodec ($\lambda_T = 0.1$)	20.8	0.224	14.9	0.533	20.7	0.247	20.0	0.494
I2Icodec ($\lambda_T = 0.05$)	20.2	0.210	15.4	0.508	20.0	0.220	19.9	0.471
T + A (w GAN)	20.0	0.088	25.8	0.288	22.2	0.082	28.1	0.265
T + A (w/o GAN)	20.6	0.098	28.7	0.268	21.1	0.089	32.5	0.238
UI2Icodec (T mode)	18.14	0.403	13.5	0.531	17.45	0.360	17.9	0.496
Real images	14.8	-	12.9	-	14.8	-	12.9	-

Table 4.1 – Quantitative comparison. The FIDs of real images are computed between the training and test sets. Note that they may not be optimal values since the number of test images is insufficient, but we report them for reference. * means the results of StarGAN v2 on CelebA-HQ are from the same model architecture on AFHQ, which doesn't include skip connections with the adaptive wing based heatmap [110]

4.5 Experiments

In this section, we describe our experimental setup and results. We analyze the effects of the rate constraint of I2Icodec in the translations (see Section 4.5.1). We also show the results of our unified framework UI2Icodec on both autoencoding and I2I translation (Section 4.5.2). Ablation study and additional results are shown in Section 4.5.3.

Datasets. Our experiments are mainly conducted on CelebA-HQ and the animal faces (AFHQ) datasets [26]. As in [26], CelebA-HQ is separated into male and female domains, and AFHQ into cat, dog and wildlife domains. We resized all images to 256×256 for training and comparisons.

Training. For I2Icodec, we train the model minimizing $\mathcal{L}_T$ during 100k iterations. We set $\gamma_{sty} = \gamma_{ds} = \gamma_{cyc} = 1$ and $\lambda_T \in \{0.01, 0.05, 0.1, 0.3, 0.5\}$. For UI2Icodec, we first train E^c and G just with distortion loss (MSE) on autoencoding mode for 50k iterations, then the whole model is jointly trained with the total loss of Eq. 4.6 for another 100k iterations with $\lambda_T = 1$, $\lambda_A = 20$ and $\beta = 1$.

Evaluation metrics. We compute FID and LPIPS metrics to evaluate the quality and diversity of the translated images, respectively. The performance of autoencoding is measured with the distortion metrics PSNR, MS-SSIM and LPIPS. Note that, for translation, LPIPS is computed between translated images, while for autoencoding, is computed between original and reconstructed images.

Method	CelebA-HQ				AFHQ			
	BPP	PSNR↑	MSSSIM (dB)↑	LPIPS↓	BPP	PSNR↑	MSSSIM (dB)↑	LPIPS↓
JPEG	0.132	21.98	6.27	0.495	0.308	25.02	10.32	0.296
BPG	0.138	28.49	12.48	0.208	0.316	27.87	12.74	0.250
NIC	0.132	29.55	11.85	0.185	0.302	29.05	12.42	0.216
UI2Icodec (A mode)	0.134	28.65	13.36	0.085	0.290	28.01	12.82	0.107

Table 4.2 – Compression performance on CelebA-HQ and AFHQ dataset.

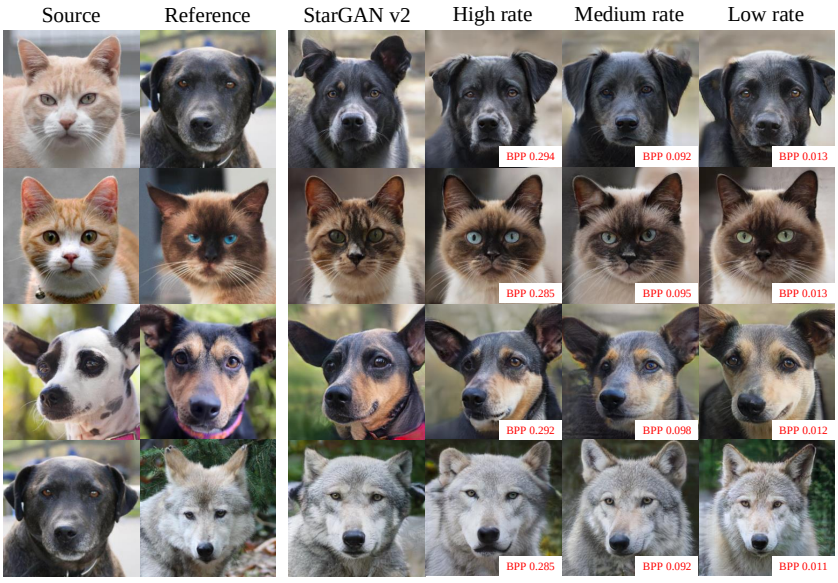
Method	NIC	StarGAN v2	I2Icodec	UI2Icodec
Number of parameters (millions)	35.30	53.73	54.22	54.23
Training time (hours)	10.3	65.7	71.34	97.6

Table 4.3 – Number of parameters and training time of different methods when run on CelebA-HQ dataset.

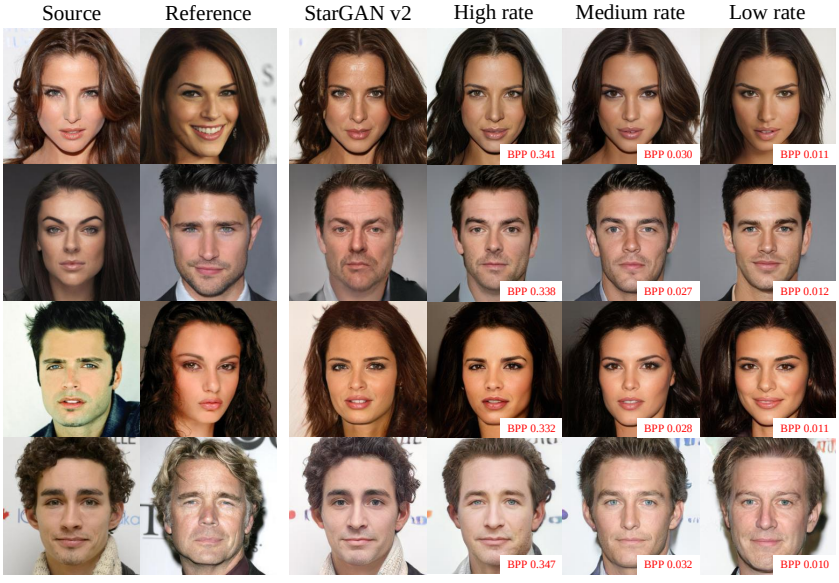
4.5.1 Distributed I2I translation

In this section, we analyze different methods to address distributed I2I translation: (1) compression before translation (CbT): the input image is compressed and then translated after reconstruction; (2) compression after translation (CaT): the input image is translated and then compressed with the image codec; (3) the proposed I2Icodec. We use the pretrained model same with [26] as translator, and two classic compression methods: JPEG and BPG as two compressor options for CbT and CaT. In addition, we also train domain-specific neural image compression models on CelebA-HQ and AFHQ separately (NIC in our experiments). We use the same encoder and decoder architecture of StarGAN v2 for fair comparison, and optimized with mean square error.

Influence of compression. As shown in Fig. 4.5, we observe changes of both FID and LPIPS values with varying rates (BPP). Notably, CbT always obtains lower FID scores than CaT, which is not surprising since CaT compresses translated images the final images have compression artifacts, resulting in worse FID. I2Icodec obtains the lowest FID among of all methods at the same rate. The diversity measured by LPIPS is shown in Fig. 4.5b and Fig. 4.5d, where CaT can achieve larger scores but with higher distortion (e.g. the second row in Fig. 4.7). CbT obtains a similar LPIPS as StarGAN v2, which are lower than our I2Icodec. In summary, I2Icodec provides an effective way to guide I2I translation by controlling the amount of information in the bottleneck via the rate constraint. In Table. 4.1, we also report the quantitative comparison with other I2I translation methods [26, 41, 54, 62] without considering compression. It shows that I2Icodec can achieve a range of scores (FID and LPIPS) on different rates for both latent-guided and reference-guided synthesis on two datasets. We want to emphasis that I2Icodec provides a lever for I2I translation.

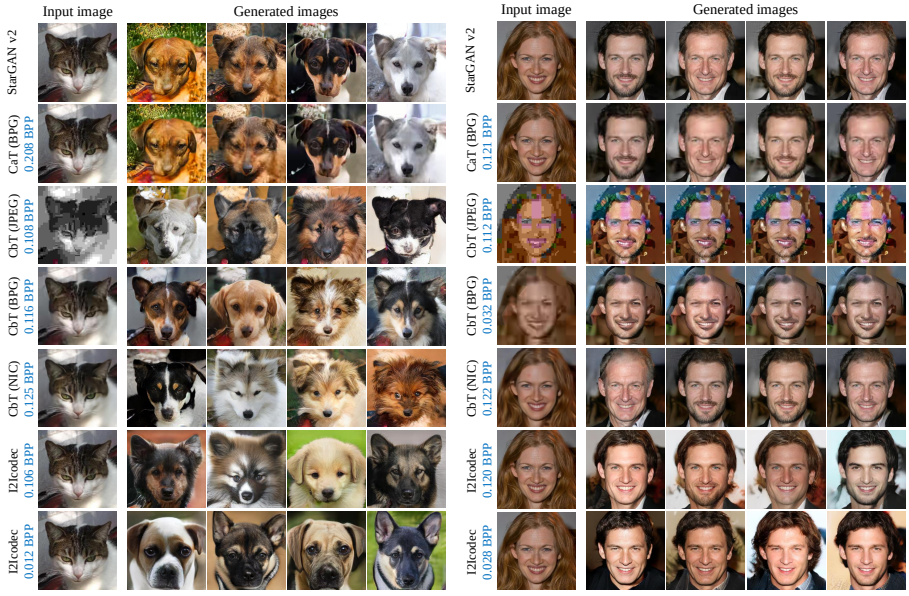


(a) AFHQ dataset



(b) CelebA-HQ dataset

Figure 4.6 – Translated images with reference on different rate.



(a) AFHQ dataset (b) CelebA-HQ dataset

Figure 4.7 – Visualization of translations with different methods (more in appendix).

Visualization of translated images. In Fig. 4.7 we show translated images using different methods. It is obvious that CaT suffers from artifacts (see 2nd row) even with the better codec BPG and higher rate than other methods. CbT with JPEG can keep some structure information, but resulting in unnatural translation (on AFHQ) or serious artifacts (on CelebA-HQ) due to the JPEG compression artifacts themselves. With BPG and NIC, the influence of compression is largely reduced (see fourth and fifth row in Fig. 4.7a), but note that it still appears again in low rates (see third and fourth row in Fig. 4.7b). I2Icodec can generate natural and diverse images even with extreme low rate. In addition, we also show the synthesized images guided by a reference image on three different rates from high to low in Fig. 4.6. It shows that the translated images have more similar style to reference image when the rate is lower, and also illustrate that this method can control well how much the translated image obtains the same style of reference image along with the rate.

4.5.2 Unified I2I translation and autoencoding

In this section, we evaluate the performance of UI2Icodec in I2I translation and autoencoding.

Autoencoding. We compare to JPEG, BPG and NIC as image compression baselines. As shown in Table. 4.2, UI2Icodec in the autoencoding mode outperforms JPEG largely and BPG marginally on PSNR and MS-SSIM (dB) on similar rates. NIC has the best PSNR results, since it was optimized for mean square error. In contrast, UI2Icodec obtains much better LPIPS scores than others methods due to the adversarial loss. Comparing the examples in Fig. 4.11, we can see that our method can keep more high frequency information and more natural reconstruction at the same rate. **I2I translation.** The quantitative evaluation of UI2Icodec in the translation

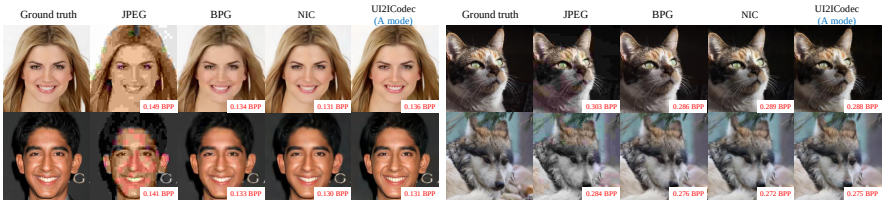


Figure 4.8 – Reconstructions with different compression methods

mode is shown in Table. 4.1. It shows that it is possible to switch image compression and I2I translation by using our method. In addition, some images generated with UI2Icodec have been already shown in Fig. 4.11. More visualization samples can be viewed in Fig. 4.9 and Fig.4.10.

4.5.3 Additional results

Ablation study. We evaluate the effects of the two main modifications of StarGAN v2 architecture: quantization+entropy coding (for compression), and adaptation units (for integrated translation+autoencoding). Comparing StarGAN v2 and I2Icodec in Table. 4.1, we observe that compression tends to improve FID and LPIPS. In contrast, comparing StarGAN and T+A (StarGAN with adaptation units and autoencoding loss), we observe that the combination of both functionalities has a small penalty in those metrics. However, an important caveat is that FID and LPIPS could be somewhat limited as evaluation metrics in this setting, and further research is required.

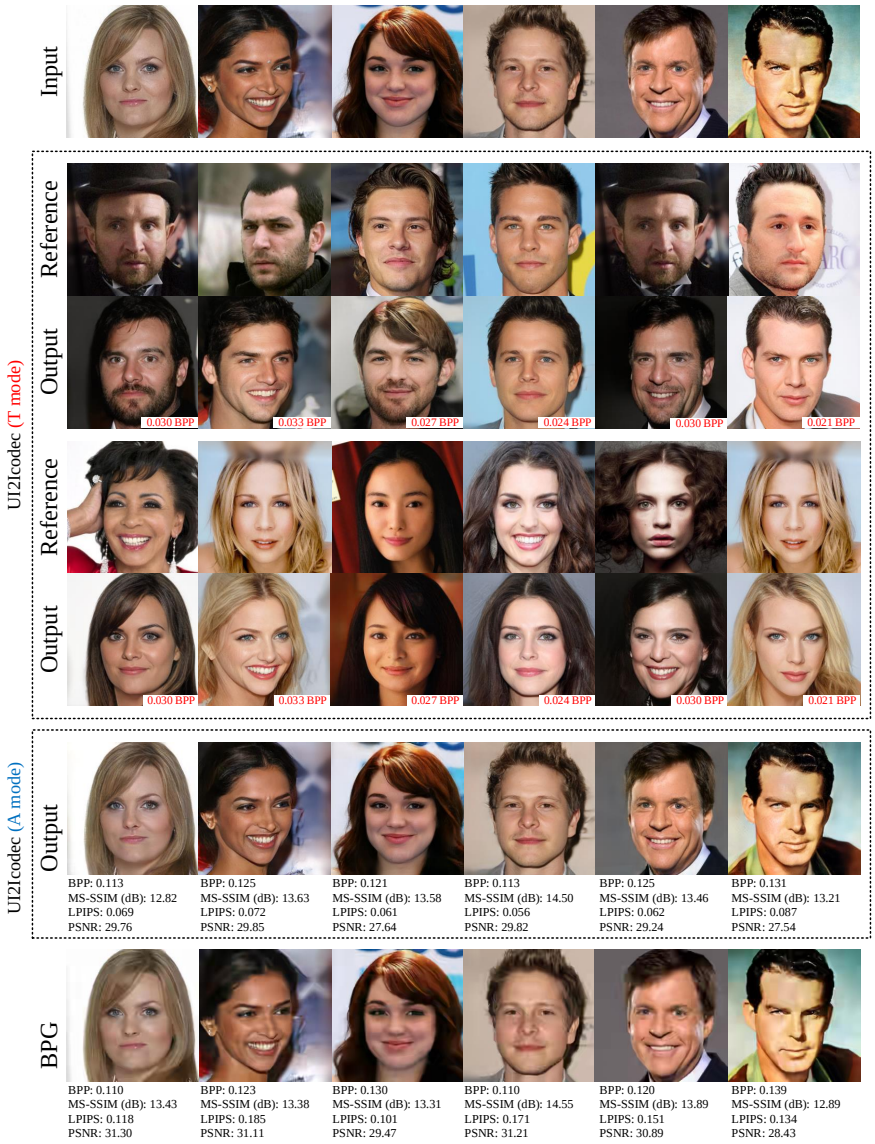


Figure 4.9 – Diverse image synthesis results of UI2Icodec in the translation and autoencoding modes on CelebA-HQ dataset.

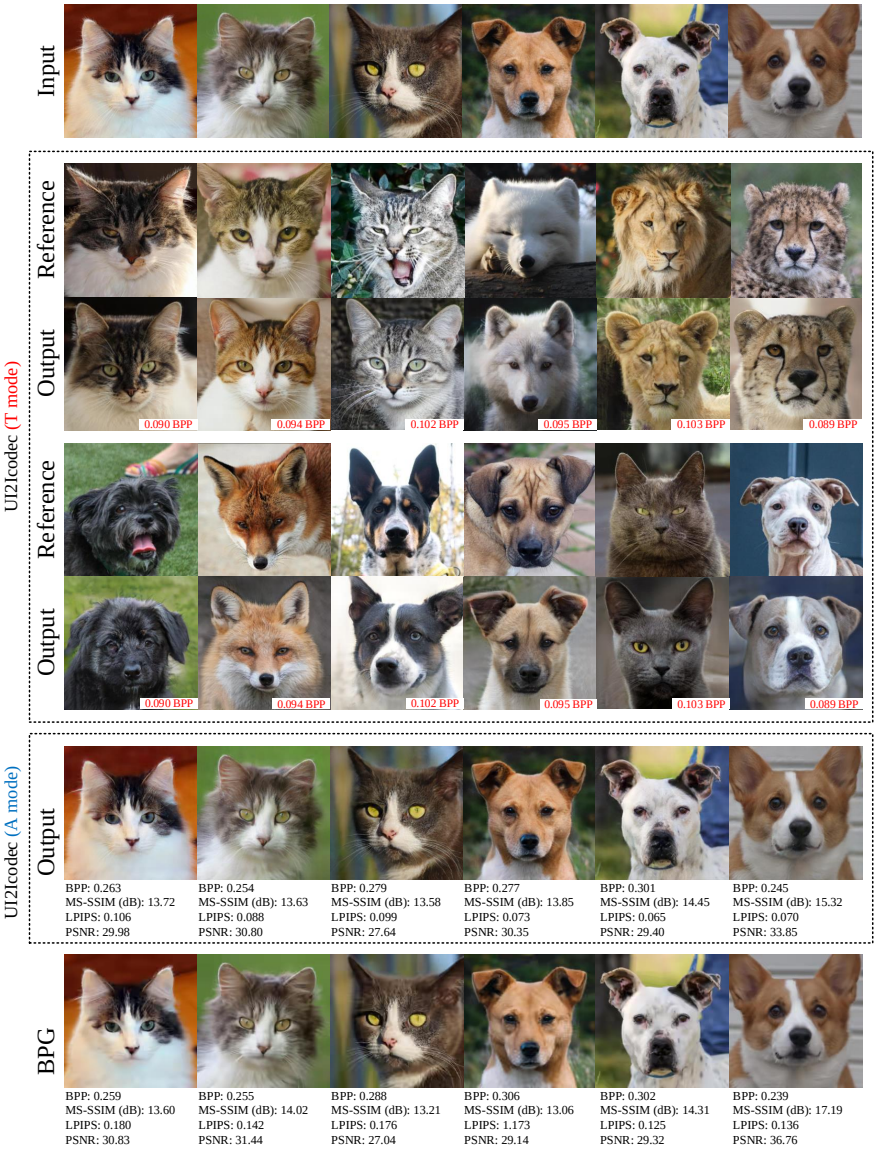


Figure 4.10 – Diverse image synthesis results of UI2Icodec in the translation and autoencoding modes on AFHQ dataset.

Model size and training time. Table. 4.3 shows that the proposed models only have 1% more parameters than StarGANv2. Note that CbT and CaT with NICs require around double amount of parameters (since there are two encoders and two decoders). Training requires 8.6%/48% more time (for I2Icodec/UI2Icodec, respectively). Similarly, note that training CbT and CaT requires training a NIC model and StarGAN, so I2Icodec requires less training time.

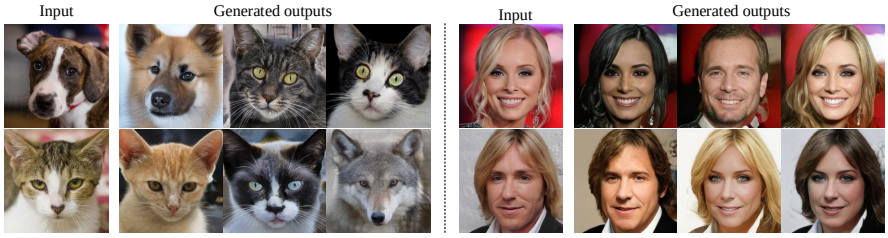


Figure 4.11 – Diverse image synthesis results of UI2Icodec in the translation mode on the CelebA-HQ and AFHQ datasets.

4.6 Conclusion

In this paper, we study a novel problem combining I2I translation and image compression, and propose a framework (I2Icodec) to address it, integrating quantization and entropy coding in an I2I translation framework, and jointly both translation and autoencoding (UI2Icodec). Interestingly, constraining the rate can control the amount of source information in I2I translation. The experiments show that our joint model can keep competitive autoencoding and translation performance.

5 Improved Discrete Optical Flow Estimation With Triple Image Matching Cost[1]

5.1 Introduction

Recently, NVC methods [40, 60, 85] have shown comparable or even superior performance than TVC methods, but there are few works considering the problem of complexity, memory requirement and latency in practice. In addition, there is still space to improve the performance of NVC with a more accurate motion estimation method. In this chapter, we study the optical flow estimation problem as our preliminary work to practical neural video compression.

Optical flow estimation is important for a large variety of computer vision applications, such as video compression, 3D scene reconstruction, autonomous driving systems and robotics. Thus, optical flow can be considered as one of the fundamental problems of computer vision. Originally, the optical flow methods are based on the assumptions of brightness constancy and spatial smoothness [36, 61]. Although there is a long research history, accurate and robust optical flow is still an open problem due to illumination changes, large displacement, blur, texture-less regions and occlusions.

Recently, several new approaches [27, 43, 80, 99] leverage end-to-end convolutional neural networks [53] to take an important step forward in optical flow estimation, and results are close to state-of-the-art. However, these networks without a special architecture for optical flow estimation can realize their full potential only with adequate training data and appropriate training arrangements [98]. For example, PWC-Net [99] gets state-of-the-art results with a new neural network architecture by embedding several classical principles: pyramid, warping and cost volume. In this chapter, we focus our efforts to improve the part of neural network architectures that is based on classic methods. Several typical methods use an initialization by approximate nearest neighbor fields (ANNF) [6, 7, 39, 113] or sparse descriptor matching [20], they leverage edge-preserving interpolation techniques [38, 83] to get final dense optical flow. High quality correspondence is the key for dense optical flow estimation.

Early optical flow methods make two assumptions: the optical flow motion vector field belongs to the set of continuous values; and the magnitude of the motion vector values is relatively small. Thus, the problem of large motion vectors arises. Note that the same problem exists in stereo matching and the most successful algorithms

[1] This chapter is based on a publication in IEEE Access (2020) [120].

rely on discrete inference, where all possible discrete disparity vectors form the matching search domain. Stereo matching methods achieve an impressive accuracy, and this is why many optical flow estimation algorithms try to exploit the same technique to solve the optical flow problem in the framework of the discrete matching paradigm [29, 68, 72, 119]. Generally, a stereo matching method pipeline consists of the following steps: matching cost computation, cost aggregation or optimization and post-processing refinement. Unfortunately, the straightforward application of this stereo matching scheme to optical flow is very difficult due to the huge size of the discrete 2D motion vector domain in comparison to the 1D stereo problem [68]. Recent progress in parallel calculation architectures shows that processing on the non-restricted cost volume is feasible and that the regular structure of this volume allows the use of global optimization techniques [24].

The main part of the above pipeline is the cost volume formation. A matching cost or a dissimilarity measure is an essential part of the correspondence problem that, in turn, is a fundamental problem in computer vision. Thus, calculation of the cost volume in stereo matching and discrete optical flow is a very important sub-problem [34]. Despite progress in the robust matching cost formation there is still one fundamental problem in the matching dissimilarity estimation: the cost uncertainty in the occluded region, due to the lack of a real correspondence between matched pixels in this case. For the standard stereo matching that uses only two images the mentioned problem cannot be solved in a straightforward manner. Fortunately, optical flow methods usually deal with more than two images and in the presented work we show how to handle occlusion problem using three consecutive images in the considered video sequences. Note that the occlusion handling in the cost volume domain improves the solution robustness also in non-occluded regions, because the energy minimization approach is very sensitive to the cost outliers.

In this chapter, we propose a new matching cost formation based on two assumptions: most occlusion regions that are invisible in the forward frame image (relative to the current frame) are visible in the backward frame; the forward flow is approximately equal to the negative value of the backward flow. The assumptions allow us to form the composite matching cost as a combination of two independent forward and backward matching costs. We consider the proposed composite cost as the main contribution of the paper. Implementation of our method increases the robustness of the optical flow estimation.

To demonstrate the advantage of the proposed cost formation we incorporate our cost in the pipeline of the state-of-the-art method DCFlow [118] and perform several experiments during each step of the prototype estimation scheme. Consequently we show that the results of the prototype method is improved for results of intermediate steps and for the final estimation of the full pipeline. Our approach considerably increases the optical flow estimation on the MPI-Sintel dataset [21] after the matching

cost processing that is the most important part of the proposed pipeline. As a result, accuracy of our estimation without the back flow consistency check increases up to 50% in occluded regions and up to 9% in non-occluded regions relative to the DCFlow algorithm original results. After post-processing steps our estimation results achieve state-of-the-art results especially on the MPI-Sintel dataset.

The rest of this chapter is organised as follows. The related works are described in section 5.2. In section 5.3 we introduce our problem definition. In section 5.4 we describe the proposed cost volume formation with triple frame. Our improved optical flow estimation pipeline is described in section 5.5. In section 5.6 we present our experiments and conclude our work in section 5.7.

5.2 Related work

There are two kinds of cost calculation approaches: a per-pixel and an area based dissimilarity measure estimation. The per-pixel dissimilarity measure usually is the Euclidean distance in the RGB color space between two image matched values or the same distance between the gradients. The robust per-pixel measure is reported [72, 119] when distances between gradients and values are combined in one measure. Early methods that exploited area based cost models calculated the cost by using a non-parametric transform with a support region such as rank and census [128] or normalized cross correlation [57]. Using a combination of these two costs can significantly improve the result of stereo matching [64]. A patch match approach is proposed in [52], where they used the sum of squared distances to compute an initial matching cost. Consequently, Kong and Tao [52] propose a new cost learning technique, which is theoretically extended by Brown et al. [18]. The latest progress in the field of CNN provides a more robust matching cost for stereo [33, 129, 130] and optical flow [7, 118]. Consequently, the traditional cost computation has been replaced by the CNN based cost in most recent works, and we also include the CNN based framework as a part of our cost calculation process.

Formally, all methods that use more than two images can be considered as related work, however the proposed triple patch match model is fundamentally different from these approaches. Usually, the related work introduces a temporal regularization [19, 49, 74, 112], a trajectory regularization [88, 106, 114] or predicts optical flow between previous frames to guide the estimation of the current flow field [16, 63]. Another related work is [14], which proposed a variational model for joint optical flow and occlusion estimation with three frames. Recently, there are several papers that embed a multi-frame optical flow estimation into the convolutional neural network architecture. Maurer *et al.* [63] proposed an unsupervised online learning approach that estimates a current motion model with multi-frame and provides predicted motion information for

forward flow estimation. Janai *et al.* [46] proposed an unsupervised learning method for multi-frame optical flow. They construct past cost volume and future cost volume with three frames and leverage convolutional neural network to reason occlusion. Neoral *et al.* [76] also estimate occlusion masks by introducing the previous frame flow and named it ContinualFlow. Ren *et al.* [82] use a neural network to fuse optical flows of different moments depending on longer-term temporal cues.

5.3 Problem definition

Discrete optical flow estimation belongs to the general matching problem, and in the framework of the global approach the matching problem is formulated in terms of energy minimization with the energy function in the following form:

$$E(\mathbf{v}) = \sum_{p \in \mathcal{V}} C_p(\mathbf{v}_p) + \sum_{(p,q) \in \mathcal{E}_p} B_{p,q}(\mathbf{v}_p, \mathbf{v}_q) \tag{5.1}$$

where set $p \in \mathcal{V}$ corresponds to pixels and set $(p,q) \in \mathcal{E}_p$ to edges of a pixel p neighborhood of an image graph $\mathcal{G} = (\mathcal{E}, \mathcal{V})$; $\mathbf{v}_p$ denotes the label of pixel p which belongs to some discrete set of 2D motion vectors $\mathbf{v} \in \mathbf{V}$ that represents the so called correspondence search region; $C_p(\cdot)$ defines a unary potential which corresponds to the conventional penalty or dissimilarity cost; $B_{p,q}(\cdot,\cdot)$ is a binary potential which defines edge interaction between pixels (p,q). Here we assume that the search region is discrete and rectangular $\mathbf{V} = [-\mathbf{v}_{max}, -\mathbf{v}_{max+1}, ..., \mathbf{v}_{max-1}, \mathbf{v}_{max}]$.

Consequently, the integer solution of the optical flow estimation problem $\mathbf{v}$ should minimize the energy functional in Eq. (5.1):

$$\mathbf{v}_p = \arg\min_{\mathbf{v}_p} E(\mathbf{v}_p) \tag{5.2}$$

The binary potential $B_{p,q}$ in Eq. (5.1) defines the local smoothness of the estimated optical flow $\mathbf{v}$ and in our algorithm has the following form:

$$B_{p,q} = \mu \min(|\mathbf{v}_p - \mathbf{v}_q|, \Delta) \tag{5.3}$$

where μ and Δ the algorithm intrinsic parameters.

As we note in the introduction the choice of the unary potential (cost) in matching tasks is very important. The main contribution of our paper is a new dissimilarity cost formation based on triple image matching. We explain the main concept and

motivation of the cost calculation in the next section. However, firstly it is necessary to describe the prototype cost calculation based on a simple convolutional neural network (CNN).

To calculate the cost volume in our pipeline we use the feature extraction CNN network trained by [118]. This small network contains 4 convolutional layers and each layer uses 64 filters. The first three layers are followed by a ReLU layer and output are normalized to produce a unit-length feature. The receptive field of this network or the size of a matched image patch is 9×9, which has proven to be effective for stereo and optical flow estimation. Or formally, the considered CNN transforms the 81D vectors $\mathbf{i}_{\mathcal{N}_p}$ that consist of image values in the patch neighborhood relevant to a pixel p into the 64D $\mathbf{u}_p$ vectors of the CNN feature space: $\mathbf{u}_p = \mathcal{T}_{CNN}\left(\mathbf{i}_{N_p}\right)$.

In turn, the cost is the vector dot-product of two matched pixel features:

$$C\left(p, \mathbf{v}\right) = 1 - \mathbf{u}_p^t \mathbf{u}_{p+\mathbf{v}}^{t+1} \tag{5.4}$$

For more details about the used CNN readers can refer the paper [118].

Energy minimization methods have lately attracted much attention in computer vision, especially in the context of image segmentation and optical flow estimation. The first implementations of the energy minimization methods such as belief propagation [42] and graph cuts [17] in stereo matching have provided a significant progress in disparity map estimation. However in our case, where a huge cost volume has to be handled in Eq. (5.1) the above approaches are computationally demanding. As the trade-off between computational complexity and accuracy of energy minimization we use semi-global matching technique (SGM) [35] to process the cost volume the same as in the DCFlow method [118].

5.4 Cost volume formation based on triple image matching

Our idea to use three constitutive frames for cost calculation is based on two principles: supplementing visibility of occluded regions in a triple frame set of a video sequence and local time optical flow constancy.

The first principle is illustrated in Fig. 5.1(a). One can see that the occlusion region $t+1$ in the frame f^{t+1} has no corresponded pixels relative to the current frame f^t, but this occluded region is visible in the frame f^{t-1}. The same supplementing visibility exists for the occluded region $t-1$.The illustrated assumption is not a physical law or a strict general observation, however in real world scenarios, those pixels which are visible in a current frame and turn invisible in the next frame are usually visible in

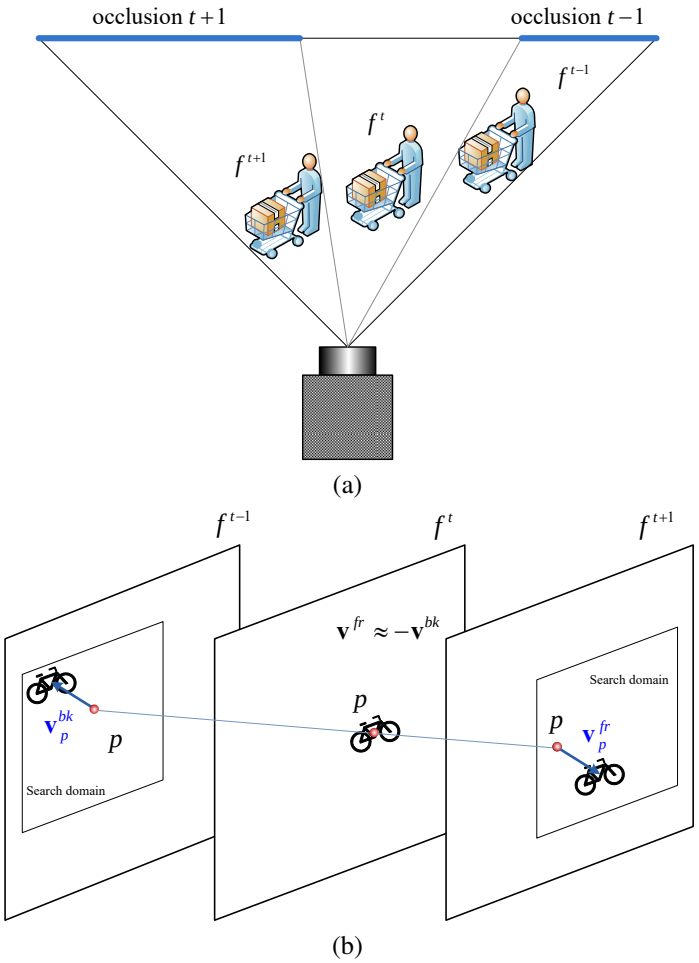


Figure 5.1 – Illustration of two main principles for the triple image matching: (a) - supplementing visibility of occluded regions in a triple frame set of a video sequence; (b) - local time optical flow constancy.

the previous frame. Thus, in the set that consists of three consecutive frames there are less pixels in the current frame that have no correspondent pixels in the next or in the previous frames.

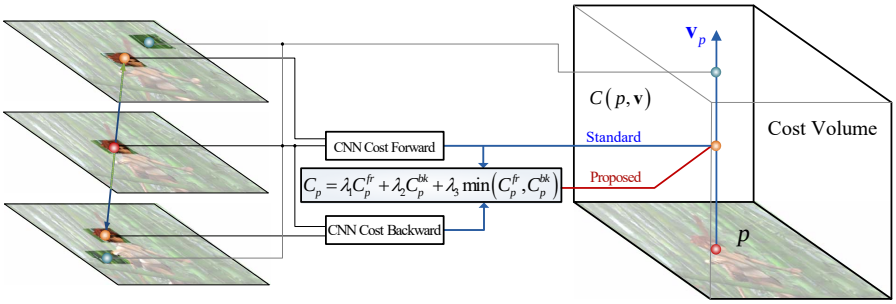


Figure 5.2 – Cost volume calculation scheme for the standard CNN based approach and for the proposed triple image matching technique. Both individual costs are unified in one triple image matching cost. Here, the middle image is used twice: for the forward and backward costs calculation.

The second principle is illustrated in Fig. 5.1(b). We suppose that the motion vector $\mathbf{v}_p^{fr}$, which corresponds to the forward optical flow direction (to the future) is equal to the negative motion vector $-\mathbf{v}_p^{bk}$, which corresponds to the backward optical flow direction (to the past). This principal is a direct consequence of the optical flow framework, and we reformulate it in the cost form rewriting Eq. 5.4:

$$C^{fr}(p,\mathbf{v}) = 1 - \mathbf{u}_p^t \mathbf{u}_{p+\mathbf{v}}^{t+1} = 1 - \mathbf{u}_p^t \mathbf{u}_{p-\mathbf{v}}^{t-1} = C^{bk}(p,-\mathbf{v}) \tag{5.5}$$

It is obvious that the above equality Eq. 5.5 holds only for non-occluded pixels in the next and the previous frames simultaneously. And for these pixels it is reasonable to make the final cost as a linear combination of the forward and backward costs to make the composite cost more robust:

$$C = \lambda_1 C^{fr} + \lambda_2 C^{bk} \tag{5.6}$$

However, if Eq. 5.5 does not hold as illustrated in Fig. 5.1(a), we assume that one of the costs C^{fr} or C^{bk} is the true cost. Consequently, to avoid ambiguities caused by occlusion, we have to choose the true one. Recall that the energy minimization approach is derived from the maximum a posteriori probability rule with the assumption that the cost of the estimated motion vector is inversely proportional to its probability: $C \propto -\log P$. It means that a lower cost value corresponds to a higher probability. Because we think that this is a good reason to choose the most probable cost as a true cost, consequently, we formalize our paradigm in the presence of occlusion as a

minimum choice between correspondent cost values:

$$C = \min\left(C^{fr}, C^{bk}\right) \tag{5.7}$$

To unify both sets of pixels: occluded and non-occluded, the final cost can be written in the following form:

$$C = \lambda_1 C^{fr} + \lambda_2 C^{bk} + \lambda_3 \min\left(C^{fr}, C^{bk}\right) \tag{5.8}$$

where linear weights λ_1,λ_2 and λ_3 are our algorithm intrinsic parameters to be optimized and we explain their choice in the experimental section. In Fig. 5.2 the computational scheme of the composite cost volume is summarized. Also one can see the difference between the proposed cost formation and the standard one.

5.5 Improved optical flow estimation pipeline

To demonstrate advantages of our proposed cost formation we design our optical flow estimation pipeline based on triple image matching cost formation (**TIMCflow**), which mainly follows the DCFlow algorithm [118]. In Fig. 5.3 we depict our main algorithm (red arrow) in parallel with the two-frame DCFlow prototype (black arrow). We demonstrate the result difference between the compared algorithms in all control points (steps) by including the relevant table in the same figure. The compared intermediate results are based on the Sintel training dataset under the EPE of **all** | **noc** | **occ** metrics. Note that numbers in Fig. 5.3 corresponding to the outlier handling step are not meaningful, because the sparseness density of the algorithms is different.

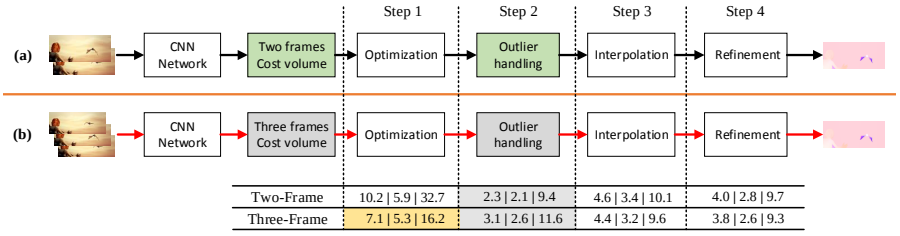


Figure 5.3 – The pipeline of two-frame baseline method (a) and our TIMCflow algorithm (b) with the results comparison on MPI Sintel data set after each step.

The cost volume formation of the scheme Fig. 5.3 is described in Section III, and in

the experimental section we compare the simple flow results obtained by two different cost formation.

The optimization block of the scheme in Fig. 5.3 is described in Section II. From Fig. 5.3 one can see that after the optimization step our three-frame approach considerably outperforms the two-frame prototypes (results highlighted by yellow).

The next step is the occlusion detection and outlier removal. To perform this task most work uses the forward-backward consistency strategy [39, 68, 118]. We also follow this idea, but the problem is that the consistency check procedure usually removes estimated flow values in occluded regions, thus we cannot capitalize advantages of the flow estimation that are achieved on the previous algorithmic step. Consequently we try several different strategies for outlier removal and choose the best that is described in the experimental section.

The next step of our pipeline is the sparse data interpolation, because the output result of the previous steps is the sparse set of estimated values and this set should be interpolated to the dense optical flow. For this purpose we choose the state-of-the-art interpolation method InterpoNet [135]. The motivation behind this choice is that the method can produce good dense optical flow for all kinds of sparse optical flow input, for example FlowField [6], DeepMatch [113], DF [68], CPM [39].

Both two-frame and three-frame pipelines include the same coarse-to-fine procedure based on a continues optimization framework [133] that is the final step of our pipeline.

The performance of the proposed algorithm on MPI Sintel benchmark is confirmed in Table 5.1, where we compare the proposed algorithm TIMCflow with five discrete optical flow methods: FlowFieldsCNN[7], CPM-Flow [39], FullFlow [24], FlowFields [6] and DCFlow [118], two interpolation methods with discrete optical flow initialization: EpicFlow [83] and InterpoNet [135]. One can see that our method is better than the prototype DCFlow method and also outperforms all compared algorithms.

In addition, we also show the results of several recent learning based optical flow estimation methods in this table: PWC-Net [97] , ProFlow [63], Back2FutureFlow [46], MFF [82].

Table 5.1 – Results on the final pass of the MPI-Sintel benchmark for different regions, velocities (s) and distances from motion boundaries (d).

Method	all	matched	unmatched	d0-10	d10-60	d60-140	s0-10	s10-40	s40+
FlowFieldsCNN[7]	5.363	2.303	30.313	4.718	2.020	1.399	1.032	3.065	32.422
CPM-Flow[39]	5.960	2.990	30.177	5.038	2.419	2.143	1.155	3.755	35.592
FullFlow[24]	5.895	2.838	30.793	4.905	2.506	1.913	1.136	3.373	35.136
FlowFields[6]	5.810	2.621	31.799	4.851	2.232	1.682	1.157	3.739	33.890
DCFlow[118]	5.119	2.283	28.228	4.665	2.108	1.440	1.052	3.434	**29.351**
EpicFlow[83]	6.285	3.060	32.564	5.205	2.611	2.216	1.135	3.727	38.021
InterpoNet_ff[135]	5.535	2.372	31.296	4.720	2.018	1.532	1.064	3.496	32.633
ours	**5.049**	**2.094**	**29.134**	**4.738**	**1.812**	**1.221**	**0.922**	**3.226**	29.926
PWC-Net[99]	5.042	2.445	26.221	4.636	2.087	1.475	0.799	2.986	31.070
ProFlow[63]	5.017	2.596	24.736	5.016	2.146	1.601	0.910	2.809	30.715
Back2FutureFlow[46]	8.814	5.031	39.647	7.153	4.880	3.904	1.752	5.961	50.725
MFF[82]	4.566	2.216	23.732	4.664	2.017	1.222	0.893	2.902	26.810

5.6 Experiments

The experiments have been designed to demonstrate the main advantages of the proposed cost formation approach. They are divided into several key parts related to the main algorithm steps in Fig. 5.3 where:

- the robustness of the proposed triple image matching cost in comparison with standard two-image matching cost is evaluated.
- the advantage of using the proposed cost in the energy minimization part of the pipeline is analyzed (corresponding to Step 1 in Fig. 5.3) and the results of our additional simplified pipeline of Fig. 5.6 are reported.
- we motivate our choice for the final outlier handling strategy by analizing the intermediate results after Step 2.
- we compare the results of two different pipelines after flow field interpolation (corresponding to Step 3) and refinement (corresponding to Step 4).
- we report and compare running time of our pipeline in comparison with the two-frame version of our algorithm.

In our experiments we mainly use the final pass of the MPI-Sintel dataset [21] that is a challenging flow evaluation benchmark, which contains long image sequences with large displacements, motion blur, defocus blur and specular reflections. For several additional experiments we also use the KITTI flow 2015 dataset [69] and a part of the Middlebury training dataset [9]. We discuss the results of the proposed algorithm in comparison with state-of-the-art methods on the Sintel dataset. Note that the KITTI dataset differs from the Sintel dataset: the first data set include shading and over-exposure, the second motion blur and dramatic occlusion. As a result, different strategies are necessary to reach state-of-the-art.

Table 5.2 – Cost volume processing results with WTA output: quantitative comparison of two-frames and three-frames using the endpoint error metric for every MPI-Sintel training set sequence. The top part of this table is the all pixels mask, the middle is the non-occluded pixels mask and the bottom is the occluded pixels mask.

	Method	alley-1	alley-2	ambush-2	ambush-4	ambush-5	ambush-6
All	3 frames	2.82	3.82	72.2	58.8	30.2	54.6
pixels	2 frames	4.36	5.79	81.4	66.2	36.6	63.2
Non-occluded	3 frames	2.38	3.18	66.8	52.4	23.6	48.8
pixels	2 frames	3.50	4.73	72.8	56.1	27.2	54.2
Occluded	3 frames	15.5	40.4	88.0	87.9	62.6	76.5
pixels	2 frames	29.8	67.1	109	111	84.2	99.4

	Method	ambush-7	bamboo-1	bamboo-2	bandage-1	bandage-2	cave-2
All	3 frames	5.14	4.70	8.60	5.08	2.74	30.1
pixels	2 frames	8.24	6.92	12.7	7.90	4.06	39.3
Non-occluded	3 frames	3.97	4.02	6.79	3.92	2.47	22.6
pixels	2 frames	6.40	5.74	9.72	5.99	3.61	27.0
Occluded	3 frames	28.3	19.2	35.0	27.2	8.97	63.7
pixels	2 frames	50.2	31.9	60.0	44.7	14.9	96.5

	Method	cave-4	market-2	market-5	market-6	mountain-1	shaman-2
All	3 frames	19.3	4.28	41.7	21.6	4.13	2.83
pixels	2 frames	27.5	6.96	50.3	28.8	6.04	4.56
Non-occluded	3 frames	14.6	3.15	33.2	15.0	3.70	2.58
pixels	2 frames	19.9	4.84	38.2	20.1	5.32	4.05
Occluded	3 frames	55.0	26.9	82.8	53.0	18.6	5.94
pixels	2 frames	86.0	49.0	110	73.3	30.9	10.6

	Method	shaman-3	sleeping-1	sleeping-2	temple-2	temple-3	**Average**
All	3 frames	2.69	1.99	1.24	11.8	27.6	**18.2**
pixels	2 frames	3.88	2.09	1.30	16.4	35.8	**22.6**
Non-occluded	3 frames	2.55	1.99	1.24	9.61	19.3	**15.1**
pixels	2 frames	3.66	2.09	1.29	12.1	23.6	**17.9**
Occluded	3 frames	11.0	1.60	1.51	37.1	61.1	**39.5**
pixels	2 frames	17.4	1.70	1.75	63.2	89.1	**58.0**

5.6.1 WTA output results comparison

We perform experiments with the winner takes all (WTA) output for two different cost formation approaches in Fig. 5.3. In this part, we found that the best results of our approach can be achieved by using $\lambda_1 = 0, \lambda_2 = 0$ and $\lambda_3 = 1$ as parameter setting in Eq. 5.8. In Table 5.2 the comparison between two different cost calculations is shown by using the final pass of the Sintel training data. One can see that our triple image matching cost produces more accurate results and improves the accuracy of the standard cost calculation technique in the occluded area by 32% and by 16% in the non-occluded region. The comparison results confirm the ability of the proposed cost to handle occlusion. Note that the proposed cost formation is more robust than the standard two-image matching cost and that is confirmed by the results in the non-occluded region.

5.6.2 Discrete flow results comparison

The next experiments are performed to show the advantage of using the proposed cost in the energy minimization part of the pipeline (Step 1 in Fig. 5.3). Here we use the SGM approach to minimize energy of the cost volume for three optical flow datasets: the MPI-Sintel, the Middlebury and the KITTI flow 2015 datasets. The Middlebury is represented only by six sequences (Grove2, Grove3, Hydrangea, RubberWhale, Urban2 and Urban3) because other sequences do not provide the ground truth or only two frames are available.

Table 5.3 – Quantitative comparison of two-frames and three-frames using the endpoint error metric with different masks.

Method	MPI-Sintel			KITTI			Middlebury
	all	noc	occ	all	noc	occ	epe
Two-frame	10.25	5.90	32.74	17.04	7.92	53.57	0.6713
Three-frame	7.18	5.39	16.29	14.87	7.16	43.27	0.6609
Three-frame+	5.99	4.26	14.27	12.14	5.56	36.28	0.5909
Three-frame++	4.65	3.01	12.63	6.00	2.13	21.37	0.2279

We prepare Fig. 5.4 to demonstrate the advantage of our approach for visual comparison with the standard two-frame approach. Occluded regions are displayed with shadow in the optical flow ground truth image. One can see that our algorithm is able to estimate flow values in occluded regions, while the two-frame method produces noisy flow values in these regions. For example, the regions in the red bounding box in Fig. 5.4 illustrate our claim. Also our method is more accurate in non-occluded regions (the regions in the blue bounding box). It is important, because non-occluded

regions expand their flow values over image boundaries (the regions in green bounding box).

In this subsection we also perform an experiment to obtain the final dense optical flow without backward flow check and interpolation. In this case, the cost volume pre-processing based on the bilateral filtering is added, like it is done in the paper [73]. Fig. 5.5 illustrates this experiment and shows that the filtering operation in the cost volume can improve the discrete optical flow estimation. Table 5.3 gives the quantitative evaluation results. One can see that for the MPI-Sintel dataset the accuracy of our estimation without the backward flow consistency check increases up to 50% in occluded regions and up to 9% in non-occluded regions in comparision with the two-frame approach, which are the original results of DCFlow algorithm before consistency check. In the case of the cost volume pre-filtering accuracy of the final result (three-frame+ in Table 5.3) increases further by 21% and 12% in non-occluded and occluded regions respectively. In this experiment we also leverage a variational

Table 5.4 – Endpoint error and density of outliers handling results on the final pass of the MPI-Sintel dataset: DCFlow (sparse matching points) and four different outliers handling strategies

Method	all	noc	occ	dens-all(%)	dens-occ(%)
DCFlow(smp)	**2.3808**	**2.1132**	9.4510	74.13	41.22
TIMCflow *Str*1	3.2187	2.4914	10.1091	**84.82**	**66.65**
TIMCflow *Str*2	3.4647	2.7555	**9.1800**	79.54	59.26
TIMCflow *Str*3	3.1035	2.6266	11.6307	81.31	52.37
TIMCflow *Str*4	2.6430	2.3106	10.1220	71.28	40.68

energy minimization post-processing method [133] to obtain our final optical flow results (three-frame++ in Table 5.3). In comparison with the DCFlow results after interpolation and post processing (Table 5.5), our method reaches the same accuracy level in non-occluded regions directly without the consistency check and interpolation. The result of this experiment demonstrates that potentially one can use our approach without the consistency check and interpolation parts.

5.6.3 Results after outliers handling

In the previous subsection it is shown that accuracy of the optical flow estimation with our triple image matching cost formation is considerably higher than with the standard two-image matching cost. In this part, we test the impact of different outlier handling methods for the final optical flow estimation results (Step 3 in Fig. 5.3). The problem is that the popular consistency check procedure usually removes estimated flow values

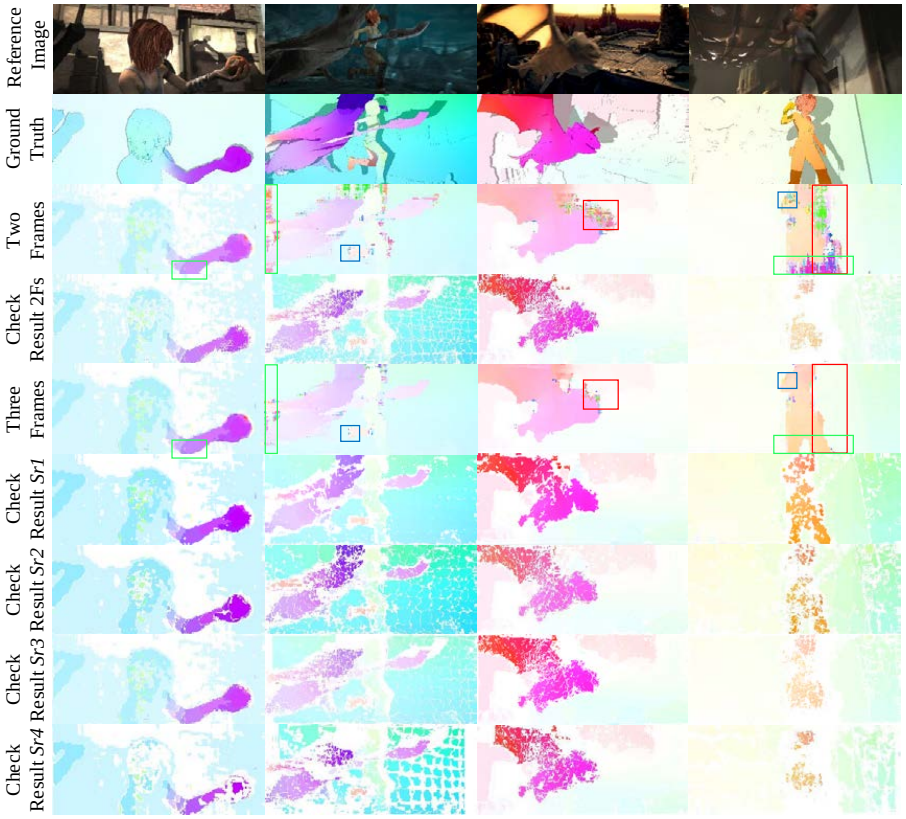


Figure 5.4 – Optical flow results illustration: the first row illustrates the reference image; ground truth with occlusion mask(shadow area) is shown in the second row; the third row illustrates discrete optical flow using two frames; the two-frame approach results after consistency check is shown in the forth row; the fifth row illustrates the optical flow results with three frames; three-frame approach results with different outlier removal strategies are shown in the sixth - ninth rows.

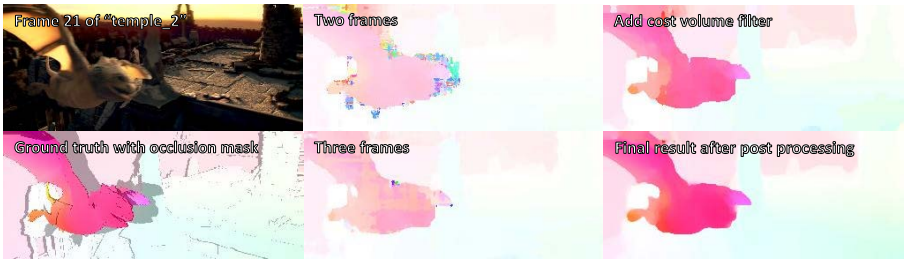


Figure 5.5 – Discrete optical flow results on the MPI-Sintel training dataset for two frames, three frames and three frames with cost volume filtering; final flow values after post-processing.

in occluded regions, thus we cannot capitalize advantages of the flow estimation that are achieved in the previous algorithmic step.

In this subsection we apply several different strategies for outlier handling, which include outlier removal based on the flow field map segmentation and several modifications of the consistency check procedure. These results are summarized in Table 5.4. For *Strategy*1, we use the breadth-first search [94] technique to segment the flow field map and remove the regions with less than 20 pixels. For *Strategy*2, we estimate two different discrete flows relative to the same current frame f^t, but with different sets of λ. We use standard forward and backward flow consistency check to *Strategy*3, but with two different thresholds T_1 and T_2 for consistency check: T_1 is equal to 0.8 for area in which $C^{fk} < C^{bk}$ and T_2 is equal to 3 elsewhere. For *Strategy*4, we use backward flow computed also with three images but shift one frame compared with forward flow. Different outlier handling results can be seen in Fig. 5.4. Formally, the best results among our strategies is achieved with *Strategy*2 and *Strategy*4. However, these strategies produce a more sparse output, thus making the final optical flow estimation worse than the output of *Strategy*3.

5.6.4 Dense optical flow results after interpolation and refinement

We consider two interpolation methods in our experiments: EpicFlow [83] and InterpoNet [135] to get dense initialization flow values for the final variational refinement. The default parameters of EpicFlow and InterpoNet are the same for different outlier handling strategies. In the interpolation part of Table 5.5 one can see that our approach gets the best interpolation results with *Strategy*3 (circle 4 in Fig. 5.3), even for the refinement results (circle 5 in Fig. 5.3). We find that the interpolation result of the InterpoNet method is better than the result obtained with the EpicFlow algorithm,

especially for occluded regions.

Table 5.5 – Interpolation and variational refinement results: quantitative comparison of interpolation and refinement results with DCFlow (sparse matching points) and for four different outliers handling strategies.

	EpicFlow			InterpoNet		
Interpolation	all	noc	occ	all	noc	occ
with DCFlow(smp)	4.71	3.48	11.17	4.60	3.42	10.13
TIMCflow *Str*1	5.19	3.58	12.97	4.54	3.40	9.73
TIMCflow *Str*2	5.32	3.85	12.48	4.82	3.70	9.79
TIMCflow *Str*3	**4.55**	**3.32**	**10.82**	**4.41**	**3.22**	**9.64**
TIMCflow *Str*4	5.32	3.92	12.25	5.10	3.37	10.81
Refinement	all	noc	occ	all	noc	occ
with DCFlow(smp)	4.13	2.91	10.68	4.01	2.81	9.78
TIMCflow *Str*1	4.46	2.92	12.02	3.92	2.78	9.34
TIMCflow *Str*2	4.59	3.15	11.62	4.40	3.27	9.53
TIMCflow *Str*3	**3.99**	**2.73**	**10.61**	**3.82**	**2.61**	**9.30**
TIMCflow *Str*4	4.69	3.28	11.87	4.44	3.06	10.39

5.6.5 Additional algorithm

The intermediate results of our algorithm considerably outperform the DCFlow method, however the final performance gain is not that significant, we propose an additional algorithm in Fig. 5.6 that simplifies the proposed calculation scheme by removing the outlier removal and interpolation steps of the original pipeline. Consequently, this innovation decreases computation complexity. In this case, the results are not better, but comparable with the original results of the two-frame pipeline, however, this decreases the computational complexity of the algorithm.

Figure 5.6 – Proposed additional algorithm based on tripe image matching cost and cost volume filter: outlier handing and interpolation part are not necessary in this simplified pipeline.

5.6.6 Running time

We report and compare running time of our main pipeline (in Fig. 5.3 and additional algorithm (in Fig. 5.6) in Table 5.6. Our three-frame version increases the calculation time minimally. In contrast, our additional algorithm Three-Frame_add algorithm considerably decreases the computational time.

Table 5.6 – Running time of different methods(sec)

Method	Cost Volume	Optimization	Interpolation	refinement	total
Two-Frame	0.35	2.50	0.41	1	4.24
Three-Frame	0.43	2.63	0.41	1	4.47
Three-Frame_add	0.87	1.19	–	1	3.06

5.7 Conclusion

In this paper, we propose a new matching cost formation based on two assumptions: most occlusion regions that are invisible in the forward frame image (relative to the current frame) are visible in the backward frame; the forward flow is approximately equal to the negative value of the backward flow. The assumptions allow us to form the composite matching cost as a combination of two independent forward and backward matching costs. The proposed method allows us to improve the standard two-frame matching technique. Consequently, our approach considerably increases discrete optical flow estimation after the matching cost processing. Experimental results have shown that our TIMCflow pipeline can get better results than two-frame pipeline and reach three first rank positions among nine metrics. In addition, we also propose a simplified pipeline without consistency check and interpolation that can keep comparable accuracy. The running time of our TIMCflow stays at the same level as the two-frame pipeline, and our reduced pipeline shortens running time significantly when compared to the full pipeline. Note that the consistency check procedure usually removes estimated flow values in occluded regions, thus we cannot fully capitalize advantages of the flow estimation that are achieved on the previous algorithmic step of our pipeline, and we plan to improve this aspect of our algorithm in future work.

6 Conclusions and Future Work

6.1 Conclusions

Although neural image compression approaches have demonstrated competitive and even superior rate-distortion performance compared to traditional image coding methods, they still remain unappealing as new generation of image coding formats, mostly due to practical requirements. Compared to traditional image codecs, NIC has important disadvantages such as much higher computational and memory requirements, fixed rate and non-adaptive complexity. Motivated by these practical limitations, in this thesis, we have contributed with solutions towards practical neural image compression.

The methods proposed and the results obtained in this thesis are:

- **Chapter 2: Variable Rate Deep Image Compression with Modulated Autoencoders.** We propose to use an modulated autoencoder to realize variable rate neural image compression. Our method can adapt the features of each layer in both encoder and decoder for different rates and needs to be trained only once with a single model. The experimental results show that the proposed method has better Rate-distortion performance than the previous variable rate method [103], even very close to the upper bound of independent models that need more training time and memory.

- **Chapter 3: Slimmable Compressive Autoencoders for Practical Neural Image Compression.** We show the relation between the rate-distortion performance of the different compression models and their capacity. Based on our observation of this relationship, we propose slimmable compressive autoencoders, where we slim the network to its minimal capacity with the given RD tradeoff. In this way, we can reduce the memory requirements, computation cost and latency, especially in low and middle rates. We also propose a train strategy named λ-scheduling to estimate the optimal RD tradeoffs corresponding to different capacities of the model. The experiment results show that SlimCAE can not only realize variable rate but also adapt computation cost, memory footprint and latency for different rate, while without performance drop than the independent models.

- **Chapter 4: A Novel Framework for Image-to-image Translation and Im-**

age Compression. This chapter proposes and studies the new problem of distributed image-to-image translation. We analyze how image compression influences image-to-image translation, and propose a novel framework to solve the translation and transmission together by introducing NIC techniques in I2I translation. Our method can provide less bits requirements and reduce the latency for distributed I2I translation. In addition, we also propose a unified framework which can work on I2I translation mode or normal neural image compression mode.

- **Chapter 5: Improved Discrete Optical Flow Estimation with Triple Image Matching Cost.** We propose to construct the cost volume with triple image frames by combining two independent forward and backward matching costs. Our method considerably improve discrete optical flow estimation after the matching cost processing, and show an large improvement than the baseline of two image frames. In addition, we also propose a simplified variant that shortens runing time significantly when compared to the full pipeline and keep comparable accuracy at the same time.

6.2 Future work

For future work we are interested in exploring the practical neural video compression continually based on our experience on neural image compression. First of all, we plan to expand our modulated autoencoder and slimmable CAE framework to video compression, and include mechanisms for rate control, which plays a crucial role in video compression and transmission in practice. The extension to video is not trivial, since it involves additional components such as motion prediction/compensation and residual coding, requiring integrating additional modules such as optical flow estimation.

Another direction for our future work is the study of the rate-distortion-complexity problem in neural image and video compression. The latter is particularly challenging, since block matching and optical flow estimation are expensive operations, and require careful analysis.

Finally, we will continue exploring the interplays between neural image compression and other computer vision tasks.

Summary of published works

6.2.1 Main publications

1. **Fei Yang**, Luis Herranz, Joost van de Weijer, José A Iglesias Guitián, Antonio M López and Mikhail G. Mozerov. Variable Rate Deep Image Compression With Modulated Autoencoder. IEEE Signal Processing Letters, 27, 331-335. 2020.

2. **Fei Yang**, Yongmei Cheng, Joost van de Weijer and Mikhail G. Mozerov. Improved Discrete Optical Flow Estimation With Triple Image Matching Cost. IEEE Access, 8, 17093-17102. 2020.

3. **Fei Yang**, Luis Herranz, Yongmei Cheng and Mikhail G. Mozerov. Slimmable Compressive Autoencoders for Practical Neural Image Compression. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 17093–17102, 2021.

4. **Fei Yang**, Yaxing Wang, Luis Herranz, Yongmei Cheng and Mikhail G. Mozerov. A Novel Framework for Image-to-image Translation and Image Compression. Under review.

6.2.2 Other collaborations

1. Mikhail G. Mozerov, **Fei Yang** and Joost van de Weijer. Sparse Data Interpolation Using the Geodesic Distance Affinity Space. IEEE Signal Processing Letters, 26(6), 943-947. 2019.

2. Sudeep Katakol, Luis Herranz, **Fei Yang** and Marta Mrak. DANICE: Domain Adaptation Without Forgetting in Neural Image Compression. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pages 1921–1925, 2021.

3. Shun Yao, **Fei Yang**, Yongmei Cheng and Mikhail G. Mozerov. 3D Shapes Local Geometry Codes Learning with SDF. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops.

Bibliography

[1] JPEG2000 (OpenJPEG). https://pypi.org/project/Glymur/, 2000. [Online; accessed 8-January-2020].

[2] Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool. Generative adversarial networks for extreme learned image compression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 221–231, 2019.

[3] Nasir Ahmed. How i came up with the discrete cosine transform. *Digital Signal Processing*, 1(1):4–5, 1991.

[4] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

[5] Amjad Almahairi, Sai Rajeswar, Alessandro Sordoni, Philip Bachman, and Aaron Courville. Augmented cyclegan: Learning many-to-many mappings from unpaired data. In *International Conference on Machine Learning (ICML)*, 2018.

[6] Christian Bailer, Bertram Taetz, and Didier Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 4015–4023, 2015.

[7] Christian Bailer, Kiran Varanasi, and Didier Stricker. Cnn-based patch matching for optical flow with thresholded hinge embedding loss. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 5, 2017.

[8] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[9] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[10] Johannes Ballé, Valero Laparra, and Eero Simoncelli. Density modeling of images using a generalized normalization transformation. In *4th International Conference on Learning Representations, ICLR 2016*, 2016.

[11] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. Density modeling of images using a generalized normalization transformation. *arXiv preprint arXiv:1511.06281*, 2015.

[12] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

[13] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

[14] Coloma Ballester, Lluis Garrido, Vanel Lazcano, and Vicent Caselles. A tv-l1 optical flow method with occlusion detection. In *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, pages 31–40. Springer, 2012.

[15] Fabrice Bellard. BPG Image Format. http://bellard.org/bpg/, 2014. [Online; accessed 8-January-2020].

[16] Michael J Black and Padmanabhan Anandan. Robust dynamic motion estimation over time. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 296–302. IEEE, 1991.

[17] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[18] Matthew Brown, Gang Hua, and Simon Winder. Discriminative learning of local image descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):43–57, 2011.

[19] Thomas Brox, Andrés Bruhn, and Joachim Weickert. Variational motion segmentation with level sets. In *European Conference on Computer Vision*, pages 471–483. Springer, 2006.

[20] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):500–513, 2011.

[21] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, October 2012.

[22] Chunlei Cai, Li Chen, Xiaoyun Zhang, and Zhiyong Gao. Efficient variable rate image compression with multi-scale decomposition network. *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[23] Chunlei Cai, Li Chen, Xiaoyun Zhang, Guo Lu, and Zhiyong Gao. A novel deep progressive image compression framework. In *2019 Picture Coding Symposium (PCS)*, pages 1–5. IEEE, 2019.

[24] Qifeng Chen and Vladlen Koltun. Full flow: Optical flow estimation by global optimization over regular grids. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4706–4714, 2016.

[25] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Variable rate deep image compression with a conditional autoencoder. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3146–3154, 2019.

[26] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8188–8197, 2020.

[27] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[28] Zhe Gan, Liqun Chen, Weiyao Wang, Yuchen Pu, Yizhe Zhang, Hao Liu, Chunyuan Li, and Lawrence Carin. Triangle generative adversarial networks. In *Annual Conference on Neural Information Processing Systems (NIPS)*, pages 5247–5256, 2017.

[29] B. Glocker, N. Paragios, N. Komodakis, G. Tziritas, and N. Navab. Optical flow estimation with uncertainties through dynamic MRFs. *In CVPR*, pages 1–8, 2008.

[30] Abel Gonzalez-Garcia, Joost van de Weijer, and Yoshua Bengio. Image-to-image translation for cross-domain disentanglement. In *Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1294–1305, 2018.

[31] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014.

[32] Vivek K Goyal. Theoretical foundations of transform coding. *IEEE Signal Processing Magazine*, 18(5):9–21, 2001.

[33] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 3279–3286. IEEE, 2015.

[34] H. Hirschmuller and D. Scharstein. Evaluation of cost functions for stereo matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007.

[35] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 807–814. IEEE, 2005.

[36] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[37] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[38] Yinlin Hu, Yunsong Li, and Rui Song. Robust interpolation of correspondences for large displacement optical flow. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017.

[39] Yinlin Hu, Rui Song, and Yunsong Li. Efficient coarse-to-fine patchmatch for large displacement optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5704–5712, 2016.

[40] Zhihao Hu, Guo Lu, and Dong Xu. Fvc: A new framework towards deep video compression in feature space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1502–1511, 2021.

[41] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *European Conference on Computer Vision (ECCV)*, pages 172–189, 2018.

[42] A.T. Ihler, J.W. Fisher, and A.S. Willsky. Loopy belief propagation: Convergence and effects of message errors. *J. Machine Learning Research*, 6:905–936, 2005.

[43] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE conference on computer vision and pattern recognition (CVPR)*, volume 2, page 6, 2017.

[44] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[45] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[46] Joel Janai, Fatma Güney, Anurag Ranjan, Michael Black, and Andreas Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 690–706, 2018.

[47] Nick Johnston, Elad Eban, Ariel Gordon, and Johannes Ballé. Computationally efficient neural image compression. *arXiv preprint arXiv:1912.08771*, 2019.

[48] Nick Johnston, Damien Vincent, David Minnen, Michele Covell, Saurabh Singh, Troy Chinen, Sung Jin Hwang, Joel Shor, and George Toderici. Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks. In *conference on Computer Vision and Pattern Recognition*, pages 4385–4393, 2018.

[49] Ryan Kennedy and Camillo J Taylor. Optical flow with geometric occlusion estimation and fusion of multiple frames. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 364–377. Springer, 2015.

[50] AH Khan and EL Hines. Integer-weight neural nets. *Electronics Letters*, 30(15):1237–1238, 1994.

[51] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jungkwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *International Conference on Machine Learning (ICML)*, 2017.

[52] Dan Kong and Hai Tao. A method for learning matching errors for stereo computation. In *BMVC*, volume 1, page 2, 2004.

[53] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[54] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Kumar Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *European Conference on Computer Vision (ECCV)*, 2018.

[55] Hsin-Ying Lee, Hung-Yu Tseng, Qi Mao, Jia-Bin Huang, Yu-Ding Lu, Maneesh Singh, and Ming-Hsuan Yang. Drit++: Diverse image-to-image translation via disentangled representations. *International Journal of Computer Vision*, pages 1–16, 2020.

[56] Jooyoung Lee, Seunghyun Cho, and Seung-Kwon Beack. Context-adaptive entropy model for end-to-end optimized image compression. In *International Conference on Learning Representations*, 2018.

[57] J. P. Lewis. Fast template matching. *Vision Interface*, pages 120–123, 1995.

[58] Mu Li, Kede Ma, Jane You, David Zhang, and Wangmeng Zuo. Efficient and effective context-based convolutional entropy modeling for image compression. *IEEE Transactions on Image Processing*, 29:5900–5911, 2020.

[59] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Annual Conference on Neural Information Processing Systems (NIPS)*, pages 700–708, 2017.

[60] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11006–11015, 2019.

[61] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[62] Qi Mao, Hsin-Ying Lee, Hung-Yu Tseng, Siwei Ma, and Ming-Hsuan Yang. Mode seeking generative adversarial networks for diverse image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1429–1437, 2019.

[63] Daniel Maurer and Andrés Bruhn. Proflow: Learning to predict optical flow. *arXiv preprint arXiv:1806.00800*, 2018.

[64] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, and X. Zhang. On building an accurate stereo matching system on graphics hardware. *GPUCV*, 2011.

[65] Youssef Alami Mejjati, Christian Richardt, James Tompkin, Darren Cosker, and Kwang In Kim. Unsupervised attention-guided image-to-image translation. In *Annual Conference on Neural Information Processing Systems (NIPS)*, pages 3693–3703, 2018.

[66] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *conference on Computer Vision and Pattern Recognition*, pages 4394–4402, 2018.

[67] Fabian Mentzer, George Toderici, Michael Tschannen, and Eirikur Agustsson. High-fidelity generative image compression. *arXiv preprint arXiv:2006.09965*, 2020.

[68] Moritz Menze, Christian Heipke, and Andreas Geiger. Discrete optimization for optical flow. In *German Conference on Pattern Recognition*, pages 16–28. Springer, 2015.

[69] Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3d estimation of vehicles and scene flow. In *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015.

[70] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pages 10771–10780, 2018.

[71] David Minnen and Saurabh Singh. Channel-wise autoregressive entropy models for learned image compression. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3339–3343. IEEE, 2020.

[72] M.G. Mozerov. Constrained optical flow estimation as a matching problem. *IEEE Transactions on Image Processing*, 22(5):2044–2055, 2013.

[73] Mikhail G Mozerov and Joost van de Weijer. Accurate stereo matching by two-step energy minimization. *IEEE Transactions on Image Processing*, 24(3):1153–1163, 2015.

[74] David W Murray and Bernard F Buxton. Scene segmentation from visual motion using global optimization. *IEEE transactions on pattern analysis and machine intelligence*, (2):220–228, 1987.

[75] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, pages 807–814, 2010.

[76] Michal Neoral, Jan Šochman, and Jiří Matas. Continual occlusions and optical flow estimation. *arXiv preprint arXiv:1811.01602*, 2018.

[77] Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for conditional image synthesis. In *European Conference on Computer Vision (ECCV)*, 2020.

[78] Majid Rabbani. Jpeg2000: Image compression fundamentals, standards and practice. *Journal of Electronic Imaging*, 11(2):286, 2002.

[79] Hayder M Radha, Mihaela Van der Schaar, and Yingwei Chen. The mpeg-4 fine-grained scalable video coding method for multimedia streaming over ip. *IEEE Transactions on multimedia*, 3(1):53–68, 2001.

[80] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 2. IEEE, 2017.

[81] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[82] Zhile Ren, Orazio Gallo, Deqing Sun, Ming-Hsuan Yang, Erik B Sudderth, and Jan Kautz. A fusion approach for multi-frame optical flow estimation. *arXiv preprint arXiv:1810.10066*, 2018.

[83] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1164–1172, 2015.

[84] Oren Rippel and Lubomir Bourdev. Real-time adaptive image compression. In *International Conference on Machine Learning*, pages 2922–2930, 2017.

[85] Oren Rippel, Sanjay Nair, Carissa Lew, Steve Branson, Alexander G Anderson, and Lubomir Bourdev. Learned video compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3454–3463, 2019.

[86] Amélie Royer, Konstantinos Bousmalis, Stephan Gouws, Fred Bertsch, Inbar Mosseri, Forrester Cole, and Kevin Murphy. Xgan: Unsupervised image-to-image translation for many-to-many mappings. In *International Conference on Machine Learning (ICML)*, 2018.

[87] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[88] Agustín Salgado and Javier Sánchez. Temporal constraints in large optical flow estimation. In *International Conference on Computer Aided Systems Theory*, pages 709–716. Springer, 2007.

[89] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[90] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Overview of the scalable video coding extension of the h. 264/avc standard. *IEEE Transactions on circuits and systems for video technology*, 17(9):1103–1120, 2007.

[91] STANDARDIZATION SECTOR and OF ITU. Information technology–generic coding of moving pictures and associated audio information: Video. *ISO/IEC 13818-2*, 2014.

[92] Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[93] Mark J Shensa et al. The discrete wavelet transform: wedding the a trous and mallat algorithms. *IEEE Transactions on signal processing*, 40(10):2464–2482, 1992.

[94] Jaime Silvela and Javier Portillo. Breadth-first search and its application to image processing problems. *IEEE Transactions on Image Processing*, 10(8):1194–1199, 2001.

[95] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal processing magazine*, 18(5):36–58, 2001.

[96] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

[97] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. *arXiv preprint arXiv:1709.02371*, 2017.

[98] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Models matter, so does training: An empirical study of cnns for optical flow estimation. *arXiv preprint arXiv:1809.05571*, 2018.

[99] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018.

[100] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[101] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[102] David Taubman. High performance scalable image compression with ebcot. *IEEE Transactions on image processing*, 9(7):1158–1170, 2000.

[103] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017.

[104] George Toderici, Sean M O'Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar. Variable rate image compression with recurrent neural networks. *arXiv preprint arXiv:1511.06085*, 2015.

[105] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full resolution image compression with recurrent neural networks. In *conference on Computer Vision and Pattern Recognition*, pages 5306–5314, 2017.

[106] S. Volz, A. Bruhn, L. Valgaerts, and H. Zimmer. Modeling temporal coherence for optical flow. In *2011 International Conference on Computer Vision*, pages 1116–1123, Nov 2011.

[107] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[108] Ting-Chun Wang, Ming-Yu Liu, Andrew Tao, Guilin Liu, Jan Kautz, and Bryan Catanzaro. Few-shot video-to-video synthesis. In *Annual Conference on Neural Information Processing Systems (NIPS)*, 2019.

[109] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *Annual Conference on Neural Information Processing Systems (NIPS)*, 2018.

[110] Xinyao Wang, Liefeng Bo, and Li Fuxin. Adaptive wing loss for robust face alignment via heatmap regression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6971–6981, 2019.

[111] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[112] Joachim Weickert and Christoph Schnörr. Variational optic flow computation with a spatio-temporal smoothness constraint. *Journal of mathematical imaging and vision*, 14(3):245–255, 2001.

[113] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1385–1392. IEEE, 2013.

[114] Manuel Werlberger, Werner Trobin, Thomas Pock, Andreas Wedel, Daniel Cremers, and Horst Bischof. Anisotropic huber-l1 optical flow. In *BMVC*, volume 1, page 3, 2009.

[115] Thomas Wiegand and Heiko Schwarz. *Source coding: Part I of fundamentals of source and video coding*. Now Publishers Inc, 2011.

[116] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.

[117] Paul A Wintz. Transform picture coding. *Proceedings of the IEEE*, 60(7):809–820, 1972.

[118] Jia Xu, René Ranftl, and Vladlen Koltun. Accurate Optical Flow via Direct Cost Volume Processing. In *CVPR*, 2017.

[119] Li Xu, Jiaya Jia, and Yasuyuki Matsushita. Motion detail preserving optical flow estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1744–1757, 2012.

[120] Fei Yang, Yongmei Cheng, Joost Van De Weijer, and Mikhail G Mozerov. Improved discrete optical flow estimation with triple image matching cost. *IEEE Access*, 8:17093–17102, 2020.

[121] Fei Yang, Luis Herranz, Yongmei Cheng, and Mikhail G Mozerov. Slimmable compressive autoencoders for practical neural image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4998–5007, 2021.

[122] Fei Yang, Luis Herranz, Joost Van De Weijer, José A Iglesias Guitián, Antonio M López, and Mikhail G Mozerov. Variable rate deep image compression with modulated autoencoder. *IEEE Signal Processing Letters*, 27:331–335, 2020.

[123] Ren Yang, Luc Van Gool, and Radu Timofte. Perceptual video compression with recurrent conditional gan. *arXiv preprint arXiv:2109.03082*, 2021.

[124] Zili Yi, Hao Zhang, Ping Tan Gong, et al. Dualgan: Unsupervised dual learning for image-to-image translation. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[125] Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1803–1811, 2019.

[126] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[127] Xiaoming Yu, Yuanqi Chen, Shan Liu, Thomas Li, and Ge Li. Multi-mapping image-to-image translation via learning disentanglement. In *Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2990–2999, 2019.

[128] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. *ECCV*, pages 151–158, 1994.

[129] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 4353–4361. IEEE, 2015.

[130] Jure Zbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17(1-32):2, 2016.

[131] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2223–2232, 2017.

[132] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Annual Conference on Neural Information Processing Systems (NIPS)*, pages 465–476, 2017.

[133] Henning Zimmer, Andrés Bruhn, and Joachim Weickert. Optic flow in harmony. *International Journal of Computer Vision*, 93(3):368–388, 2011.

[134] Barret Zoph and Quoc Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017*, 2016.

[135] Shay Zweig and Lior Wolf. Interponet, a brain inspired neural network for optical flow dense interpolation. *arXiv preprint arXiv:1611.09803*, 2016.