LEARNING DEEP NEURAL NETWORKS FOR IMAGE COMPRESSION

MU LI

PhD

The Hong Kong Polytechnic University

2020

THE HONG KONG POLYTECHNIC UNIVERSITY

DEPARTMENT OF COMPUTING

Learning Deep Neural Networks for Image Compression

Mu Li

A Thesis Submitted in Partial Fulfillment of

the Requirements for the Degree of

Doctor of Philosophy

September 2019

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ _____(Signature)

_____Mu Li _____(Name of Student)

ABSTRACT

Data compression is a very basic problem in computer science and has been studied for decades. The contradiction between the increasing amount of data and the limited saving space arises soon after the invention of the computer and has never been completely solved until today. After the population of the Internet, the limited bandwidth and the increasing data to be transported became another significant contradiction. With the limited saving space and bandwidth, a method to compress the data into a smaller size is very valuable to the whole computer society. Recently, social media becomes a hot topic in our daily life. Media like image and video is now a major data type. Besides, the deep learning methods have shown unprecedented success and powerful fitting ability in many computer vision and natural language processing problems. In this thesis, we choose to develop better image compression methods with powerful deep learning toolkits.

Image compression methods are divided according to the decoded images. One branch is the lossless image compression method which requires the decompressed images to be the same as the original images. The other branch is the lossy image compression method that allows the decompressed images to be different from the original images. With small scarification on the image quality, the lossy image compression methods could compress the image into a much smaller size. In practice, most of the popular image compression standards are lossy image compression methods. Lossy image compression methods usually are modeled as a rate-distortion optimization problem. Two issues should be considered when building a deep image compression method. The first issue is quantization. Quantization functions are generally step functions with zero gradients at almost all the regions except for several points where the gradient is infinite. All the neural networks before the quantization operation could not be optimized with the back-propagation algorithm. Another issue is how

to model the discrete entropy of the codes.

In the first job, we analyze the informative content in the image and build a content weighted lossy image compression framework with deep networks. Considering that in an image with an eagle flying in the blue sky, the informative part, i.e., the eagle, should be more important than the sky. Thus, when the bits used to code the image are limited, it is reasonable to allocate more bits to the informative important parts and fewer bits to code the unimportant parts instead of allocating the same bits for all the parts evenly. We introduce a side information network to summarize the informative importance of different parts of the image as the importance map and allocate the different number of bits to different parts correspondingly. And the sum of the importance map is adopted as the upper bound of the discrete entropy of the codes. For the quantization, a binarization function is adopted. And a continuous proxy function of quantization function is introduced for back-propagation to tackle the gradient problem. The whole framework consists of an analysis transform, a synthesis transform, and a side importance map network. The analysis transform takes the image as input and generates the code representations which are further quantized into discrete codes. And the codes are decoded by the synthesis transform to produce the decoded image. The model is end-to-end optimized on a subset of the ImageNet dataset and tested on Kodak dataset where it outperforms the image compression standards like JPEG and JPEG2O0O by SSIM index and generates better visual results.

The first job is still inferior to the state-of-art image compression standards like BPG. We analyze the shortcomings of the first job and improve it as follows. First, a DenseBlock is introduced to build the encoder and decoder. Secondly, a channel-wise learnable quantization function is introduced by minimizing the quantization error between the proxy function and the quantization function. With smaller quantization error, the gradient produced by the proxy function would be more accurate. Especially, when the quantization error is 0, the quantization function is the same as the proxy function. The gradient estimated by the proxy function is the real gradient. Finally, we introduce a 3D mask convolutional network for post

entropy coding. In the previous job, a small 3D block around a target code is extracted as the context to predict the discrete probability table of the code. Compared to the previous job, the mask CNN could employ a larger context and is computational more efficient due to the share computation. With the above improvements, our job could outperform the state-of-art image compression methods especially at low bit rates and achieve visually much better results. And we further apply the framework for task-driven image compression with task-driven distortion loss.

In the third job, we focus on entropy modeling, generalize the mask CNN proposed in the second job, and introduce a general context-based convolutional network (CCN) for efficient and effective context-based entropy modeling. The CCN is more general and can be applied for any context and coding order with the given property. The previous mask CNN could predict the probability of all the codes in parallel in encoding but have to process the codes in serial order in decoding due to the limitation of the context. For better efficiency in decoding, we proposed a 3D zigzag scanning order for the 3D code block generated by analysis transform together with a code dividing technique to cut the codes into different groups. By removing the dependency among the code in the same group, the introduce context can be used in CCNs for parallel decoding. Without a clear drop in the effectiveness, the proposed special context-based CCN can speed up the decoding process by a lot. We test the CCN for lossy and lossless image compression. For lossy image compression, we directly apply a CCN on binarized grayscale image planes to predict the Bernoulli distribution of each code. For lossy image compression, without further hypothesis on the probabilistic distribution of the codes, we adopt a mixture of Gaussian (MoG) distributions to predict the distribution of the codes whose parameters are estimated with CCNs. The discrete entropy built on the MoGs is further used as the rate loss to guide the end-to-end optimization of the transforms and the CCN based entropy model. On both lossy image compressions, the proposed CCN based entropy modeling outperforms all the current lossless image compression standards. As for lossy image compression, the proposed methods achieve state-of-art performance in

low bit rate region.

The traditional convolutional networks could only adopt some local information in its receptive filed for computation, and the information outside the receptive filed is usually ignored. Due to the structural limitation, the CCN can only apply local context for entropy modeling. The global context and non-local similarity are naturally discarded. In the fourth job, we dig out the non-local similarity of the codes inner the context and exploit this prior in context-based entropy modeling. The CCNs proposed in the third job are adopted to handle the local context. And a non-local attention block is introduced to combine the local representation produced by the CCNs and the non-local estimation generated by the content related weights from the global context. Also, a UnetBlock is introduced for the synthesis and analysis transforms. The width of the network, i.e., the minimum number of filters in the network, is supposed to be important in determining the performance for low distortion models. The introduced UnetBlock can help increase the width of the transforms with manageable computational consumption and time complexity. With the UnetBlock and the context-based non-local entropy modeling, the model is end-to-end optimized on images collected from the Flickr. We test the model on Kodak and Tecnick datasets and find that both of the non-local entropy modeling and the UnetBlock are effective in improving the performance and the whole model can achieve the state-of-art performance not only at low bit rate region but also high bit rate region. Among the four jobs, the fourth job achieves the best performance.

To summarize, we have done four jobs for lossy image compression with deep convolutional networks. The first two of them focus on the content variant image compression framework. And the last two jobs are more general and aim to build better entropy modeling which could be used for any other image compression jobs. With the four jobs, we have achieved state-of-art performance on lossy image compression tasks.

Keywords: Image Compression, Importance Map, Convolutional Networks, Context-based

Entropy Modelling, Non-local.

PUBLICATIONS ARISING FROM THE THESIS

Published Papers

1. **Mu Li**, Wangmeng Zuo, Shuhang Gu, Debin Zhao, and David Zhang, Learning convolutional networks for content-weighted image compression, In Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. (Citation: 121)

Unpublished Papers

1. **Mu Li**, Kede Ma, Jane You, David Zhang, and Wangmeng Zuo, Efficient and Effective Context-Based Convolutional Entropy Modelling for Image Compression, submitted to IEEE Transactions on Image Processing (TIP).Under Minor revision

2. **Mu Li**, Wangmeng Zuo, Shuhang Gu, Jane You, and David Zhang. Learning Content-Weighted Deep Image Compression, submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI). Under Minor revision.

3. **Mu Li**, Wangmeng Zuo, Bob Zhang, Jane You, and David Zhang. Learning Context-Based Non-local Entropy Modelling for Image Compression, submitted to IEEE Transactions on Image Processing (TIP).

ACKNOWLEDGEMENTS

First and foremost, I want to thank my supervisors, Prof. David Zhang and Prof. Jane You, for their guidance, support and encouragement in my study. They taught me not only how to become a good researcher but also how to be a good person.

I would like to express my gratitude to Prof. Wangmeng Zuo, from whom I learnt the whole process of doing research. He also help me a lot in revising my papers.

I also want to give my thanks to Jinxing LI, Hongwei Yong, Keze Wang, Kede Ma, Shuhang Gu, Lingxiao Yang, Xixi Jia, and Kai Zhang for the happy memories in the lab and their kind support and meaningful suggestions in my research.

Finally, I want to specially appreciate my parents and my sister for their endless love, sacrifice, support, and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

Data compression [72] is a very basic topic in computer science. It aims to represent the information of the data with fewer bits. Due to the insufficient storage of the computer and the limited bandwidth of data link, how to compress the data into a smaller size is a crucial problem at the very beginning of the emergence of the computer. Even though the storage increase rapidly according to Moore's law [49], the data needed to be saved or transmitted also grows correspondingly. After the population of the personal computer, the media, including digital images and videos, became a major data type for compression.

Digital images usually are very large. For example, a 4K color image without compression, *i.e.*, 4096×2160, would occupy $25,920$ KB for saving, which is even larger than the size of all the text in a normal book. Considering that the spatially nearby pixels inner an image are closely related to each other, the natural images have a lot of redundant information which could be compressed by a lot. By analyzing the redundancy, a simple image compression method could easily compress the images by 10 times without a clear visual difference. Thus, image compression is very useful in saving storage and transmit bandwidth.

Recently, the social media, like Facebook and YouTube, are very popular in our daily life, the amount of media like image and videos are growing explosively. And another hot topic, artificial intelligence, also depends on the images collected by cameras. With a large number of images, better image compression methods are still needed. The last decade has witnessed the unprecedented success of deep learning which shows to have powerful fitting ability in many computer vision and natural language processing problems. In this thesis, we aim to develop better image compression methods with deep neural networks.

1.1 Lossy and Lossless Image Compression

According to the decoded image, image compression methods are divided into two branches. One branch is the lossless image compression method which can perfectly reconstruct the original image from the codes. The other branch is the lossy image compression method where the decoded image is allowed to be different from the original image. And the difference between the original image and the decoded image is defined as distortion.

Rate is one of the important metric to evaluate the image compression methods. Usually, the rate is evaluated by compression ratio or bits per pixel (bpp). The compression ratio is defined as,

$$\text{Compression Ratio} = \frac{\text{Size of the uncompressed image}}{\text{Size of the compressed file}}, \tag{1.1}$$

and the bits per pixel is defined as,

$$\text{bpp} = \frac{\text{Size of the compressed file in bits}}{\text{Number of pixels of the uncopmressed image}}. \tag{1.2}$$

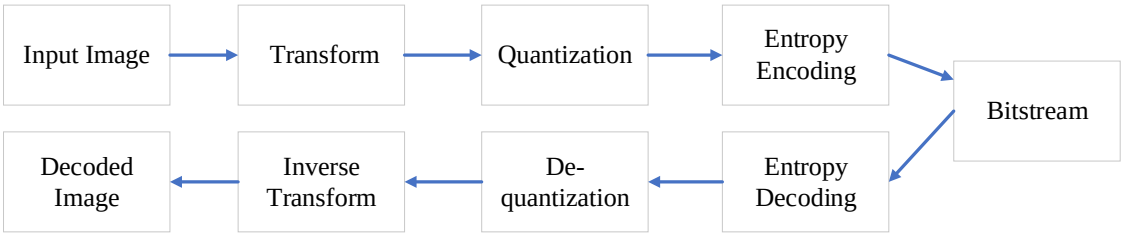
For lossless image compression methods, bigger compression ratio (small bits per pixel) indicates better image compression methods. For lossy image compression methods, things are a little complex in evaluating the performance of the methods. Besides the rate, another important metric, distortion should also be considered. A most simple and widely used metric for distortion is mean square error, which is defined as,

$$\text{MSE} = \frac{1}{HW}\|\mathbf{x} - \mathbf{y}\|_2^2, \tag{1.3}$$

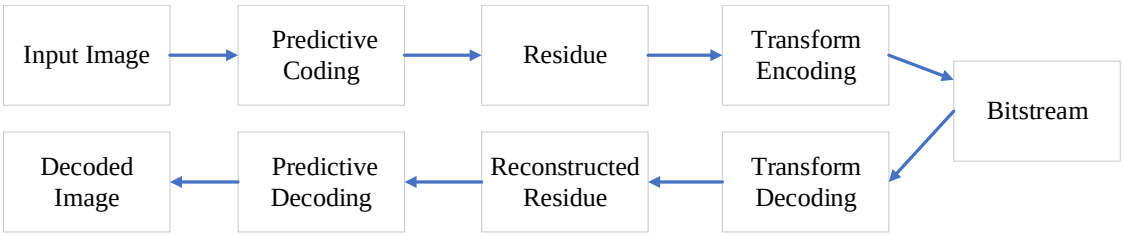
where $x_{i,j}$ and $y_{i,j}$ are separately the pixel of the input image and the decoded image at the 2D location of (i, j). Corresponding to MSE, another metric Peak Signal-to-Noise Ratio (PSNR) is also a common distortion metric.

$$\text{PSNR} = 10\log_{10}\frac{255^2}{MSE}. \tag{1.4}$$

However, the above distortion metrics are simple but fail to consider the characteristics of the human visual system. As a result, such metrics can not perfectly indicate the visual quality of the image. Some better metrics like structural similarity index (SSIM) [76]



(a)



(b)

Figure 1.1: Illustration of the two image coding framework. (a) transform coding framework. (b) hybrid coding framework.

and multi-scale structural similarity index (MS-SSIM) [77] which are more consistent with the human visual system are introduced to evaluate the distortion.

Taking both the rate and distortion into account, the lossy image compression methods are usually evaluated by a joint rate-distortion curve. Without noticeable distortion, lossy image compression methods can achieve a much larger compression ratio than the lossless image compression methods can do. Thus, lossy image compression standards, such as JPEG and JPEG2000, are widely used in our daily life.

1.2 Transform Coding and Hybrid Coding

For image compression methods, the goal is to remove the redundant information in the image. With different strategies, mainly two frameworks are designed for image compression. The first and most adopted framework is the transform coding framework, which usually follows the pipeline of transforming, quantization, entropy encoding to generate the bitstream for the compressed file, and decodes the compressed file to get the decompressed image with entropy decoding, inverse transforming. Figure 1.1 (a) shows the framework of the transform

coding framework. Most of the well-known image coding standards, including JPEG [73] and JPEG2000 [64], adopted the transform coding framework.

As illustrated in Figure 1.1 (b), another framework, hybrid coding framework, is a mixture of predictive coding and transforming coding. Firstly, a predictive coding step is adopted to predict the pixels from context and then code the residue between the input image and the prediction with the transform coding system. The first step aims to remove the redundancy of the image and the second step aims to remove the redundancy of the residue. The image coding system, BPG (HEVC intra coding) [13], follows this framework, and produce overwhelm performance compared with JPEG and JPEG2000.

1.3 Traditional Image Compression Methods

As a very basic problem in computer science, image compression has been studied for decades. And many well-know image compression standards have been proposed for better performance. In this thesis, we choose to briefly review three representative methods, *i.e.*, JPEG, JPEG2000 and BPG (HEVC Intra Coding).

1.3.1 JPEG

The JPEG standard [73] is designed by the Joint Photographic Expert Group for still image compression. It included several models including baseline model, lossless model, progressive model, and hierarchical model. Among the models, the most popular model is the baseline model, which is even the most popular still image compression standard. The baseline model follows the transform coding framework. It first divides the images into 8×8 patches and adopts the DCT transform [6] to process the 8×8 image patches. The 64 DCT coefficients after transformation are further quantized with a uniform scalar quantizer. Finally, the Huffman coding [31] is adopted to mapping the coefficients to bitstreams for saving. The lossless model is quite different, which makes use of a predictive coding scheme based on the nearest neighbors and 7 different predictors. The residue is directly entropy coded with the Huffman coding.

1.3.2 JPEG2000

JPEG2000 is another popular image compression standard designed for improved compression performance. Compared with JPEG, JPEG2000 has properties including progressive decoding, random access, lossless to lossy compression, and tolerance for errors. JPEG 2000 also follows the transform coding framework. Instead of using DCT transform, JPEG2000 adopts the discrete wavelet transform (DWT) [1, 8]. Besides, context modeling together with the arithmetic coding [79] is introduced for entropy coding. And a post-compression rate allocation scheme is designed for better compression performance. With the DWT transform and the efficient code-stream organization scheme, JPEG2000 can do progressive decoding, which means that JPEG2000 can construct a lower quality version image with only a smaller part of the whole image. With more parts of the file received, the quality of the decoded image will increase correspondingly. Another significant feature is the ROI image compression scheme. With a given ROI map, the regions of interest are coded with better quality than the other parts of the image.

1.3.3 HEVC Intra Coding

BPG [13], also known as the HEVC intra coding [35], follows the hybrid coding framework which consists of predictive coding built on spatial sample prediction, transform coding and post-processing. Different from the block dividing scheme in JPEG, BPG adopts a quadtree-based coding structure for block dividing. Besides, the agnostic angular prediction based on 33 prediction directions and planar prediction for smooth sample surfaces is jointly adopted to predict the image blocks. After predictive coding, the residue will be further compressed with the transform coding which shares the same transform coding pipeline as HEVC inter coding. This pipeline consists of transform, quantization, entropy coding, removing blocking effects, and applying sample adaptive offsets.

1.4 Deep Learning Based Image Compression Methods

As a popular tool-kit in recent years, deep learning have been well investigated in computer vision problems including image classification [34], visual tracking [74], object detection [57], denoising [81, 83], and super-resolution [20], and natural language processing problems including machine translation [16], question answering [42, 50], and speech recognition [25, 29]. The unprecedented success of deep learning throws light on designing better image compression methods. Currently, nearly all the deep learning methods follow the transform coding framework and adopt deep networks as transforms which is further optimized for a joint rate-distortion objective loss function. In designing a deep image compression framework, two issues, *i.e.*, quantization, and entropy modeling, should be considered. In this section, we will first explore the quantization and entropy modeling strategies used in deep image compression methods and then briefly review the deep image compression methods.

1.4.1 Quantization

A major problem in end-to-end lossy image compression is that the gradients of the quantization function are zeros almost everywhere, making gradient descent-based optimization ineffective. Based on the strategies of alleviating the zero-gradient problem of quantization, DNN-based lossy image compression methods can be divided into different categories.

From a signal processing perspective, the quantizer can be approximated with an additive i.i.d. uniform noise, which has the same width as the quantization bin [26]. The desired property of this approximation is that the resulting density function is a continuous relaxation of the probability mass function of $\boldsymbol{y}$ [11]. Given the features to be quantized denoted as z, the quantization operation $Q(z)$ is defined as

$$Q(z) = z + \mathbf{u}. \tag{1.5}$$

Here, $\mathbf{u}$ is sampled from a uniform noise distribution. In testing, z is directly rounded to its nearest integer.

Another line of research introduced more continuous functions (without the zero-gradient problem) to approximate the quantization function. The step quantizer is used in the forward pass, while its continuous proxy is used in the backward pass [67]. For example, $Q(z) = [z]$ is used for forward-propagation and $\hat{Q}(z) = z$ is adopted for back-propagation. Here, $[.]$ is the round operation.

1.4.2 Entropy Modelling

Learnt DNN-Based lossy image compression methods are usually modeled as a joint rate-distortion optimization problem. Estimating the entropy of the codes which depend on certain potential probabilistic distribution functions (PDFs) is one of the most important issues in building a deep image compression framework. Early methods [2, 11, 58, 67] only adopt DNNs for transforms and directly suppose that all the codes are independent and follows the same PDF for easy to handle entropy models. Without accurate estimation of the entropy, the performance of these methods is not satisfactory enough compared with traditional image compression standards like JPEG2000 and BPG. It is known to all that natural images have continuous prior. The codes generated from the image with a non-linear analysis transform still keep some strong statistical redundancies, which leave a lot of space for better entropy modeling with extra information, *i.e.*, context [37, 38, 46, 48, 69] or hyperprior [12] of the codes. For better entropy modeling, DNNs started to be used to estimate the PDFs of the codes from code context or hyperprior summarized from the code.

1.4.3 Overview

In the encoder-decoder framework, both recurrent neural network (RNN) and conventional network (CNN) based models have been developed for lossy image compression. In [68], Toderici et al. suggest a RNN architecture to compress the residual of a 32×32 images progressively. Subsequently, they [69] present a new variant of a gated recurrent unit (GRU) as well as content-based residual scaling for RNN-based full-resolution image compression. Three improvements are further introduced in [32], i.e., improved network architecture, spatially adaptive bit allocation, and SSIM-weighted loss. In contrast to our pioneer work as

shown in chapter 2 and 3, they [32] simply adopt a handcrafted spatially adaptive bit allocation post-processing scheme.

Using variational auto-encoder (VAE), Ballé et al. [11] adopt a uniform noise approximation for modeling quantization error, and present a continuous and differentiable proxy of the rate-distortion loss. Supposing that all the codes in one feature map are independent, a linear piece-wise probability density function (PDF) is learned for each channel to estimate the entropy of the codes. In [12], they further introduce a scale hyperprior as side information for capturing spatial dependency and then model the entropy of the codes conditioned on the learned hyperprior. Furthermore, Minnen et al. [48] utilize a single PixelCNN layer for modeling autoregressive priors and combine it with hyperprior for boosting rate-distortion performance.

Convolutional auto-encoders have also been studied and applied to lossy image compression. Theis et al. [67] provide a continuous upper bound of entropy rate and replace rounding function with its smooth approximation in backward propagation. For modeling the distribution of encoder features, they adopt Gaussian scale mixtures. Agustsson et al. [2] introduce a soft relaxation of the quantization and entropy, and adopt a soft-to-hard annealing scheme in training. Rippel et al. [58] incorporate a deep auto-encoder with pyramidal decomposition and adaptive code length regularization to develop a real-time high performance compression system. Adversarial loss [24] is also considered in [4, 58] to generate visually realistic decoding images for low bit rates. Soon after our pioneer work as shown in chapter 2 and concurrently with our work in chapter 3, Mentzer et al. [46] introduce a masked convolutional network for capturing spatial dependency in entropy model, and suggest a learning scheme by alternatingly updating the entropy network and auto-encoder in training.

1.5 Contributions

- As the first several jobs on deep image compression framework, we introduce a deep framework to learn content weighted spatially variant transformations. Different from

the traditional transform coding framework, the deep transforms modeled with convolutional networks is inevitable. Distortion arises from the transform. As each channel in the code is a representation of the input image, stacking more channels will help reduce the distortion brought by the deep transforms. We propose to learn a content related importance map from the image for a joint rate-distortion optimization. The learned importance map allocates more channels for the informative important regions to help reduce the extra distortion and keep important details such as edges. With the whole number of codes is limited, our scheme could allocate more channels for the informative important region and few channels for flattening regions.

- Better network structure, learnable multi-value adaptive quantization function, better learning strategy for importance map, and a mask convolutional network for context-based entropy modeling are introduced to improve the learned content weighted image compression framework. And the framework is further used for task-driven image compression.

- A general context-based convolutional network (CCN) is proposed for context modeling. Any contexts and scanning orders that fulfill with the requirement can be modeled with the CCNs. And a special code dividing scheme together with a partial context is introduced to accelerate the decoding efficiency by a lot without a noticeable performance drop.

- We introduce non-local similarity in the code representations for context modeling and get state-of-art image compression performance.

1.6 Dataset

In chapter 2, we follow Ballé et al. [11], adopts images from the ImageNet as the training data for optimizing the deep image compression framework. However, the images from the ImageNet are compressed with JPEG standard with a low quality. Compression artifacts such as ringing, blurring, and blocking, are witnessed in the training data. To remove the effection from the compression artifacts, we crawled 10,000 high-quality images from the

photo-sharing website Flickr and downsample the image for 3 times and save them with the lossless image compression standard, PNG. The collected images are used in the last three chapters. For evaluating the performance of the proposed methods, we adopt Kodak Photo CD dataset and the Tecnick dataset. There are 24 images with the size of 752×496 (496×752) in Kodak and 100 images with the size of 1200×1200 in Tecnick

1.7 Conclusion

The rest of the thesis is organized as follows. In Chapter 2, we propose a content weight image compression framework. And in Chapter 3, the framework is extended with several improvements, which is further applied for task-driven image compression where the special tasks also contribute to the learning of importance map. In Chapter 4, a general context-based deep entropy model and an accelerating strategy are proposed for modeling the entropy of image compression frameworks. In Chapter 5, we introduce the non-local similarity of the codes and a U-net structure. The thesis is concluded in Chapter 6.

CHAPTER 2
LEARNING CONTENT-WEIGHTED DEEP IMAGE COMPRESSION

Lossy image compression is generally formulated as a joint rate-distortion optimization problem to learn encoder, quantizer, and decoder. Due to the non-differentiable quantizer and discrete entropy estimation, it is very challenging to develop a convolutional network (CNN)-based image compression system. In this chapter, motivated by that the local information content is spatially variant in an image, we suggest that: (i) the bit rate of the different parts of the image is adapted to local content, and (ii) the content-aware bit rate is allocated under the guidance of a content-weighted importance map. The sum of the importance map can thus serve as a continuous alternative of discrete entropy estimation to control the compression rate. The binarizer is adopted to quantize the output of the encoder and a proxy function is introduced for approximating binary operation in backward propagation to make it differentiable. The encoder, decoder, binarizer and importance map can be jointly optimized in an end-to-end manner. And a convolutional entropy encoder is further presented for lossless compression of importance map and binary codes. In low bit-rate image compression, experiments show that our system significantly outperforms JPEG and JPEG 2000 by structural similarity (SSIM) index, and can produce the much better visual result with sharp edges, rich textures, and fewer artifacts.

2.1 Introduction

Image compression is a fundamental problem in computer vision and image processing. With the development and popularity of high-quality multimedia content, lossy image compression has been becoming more and more essential in saving transmission bandwidth and hardware storage. An image compression system usually includes three components, *i.e.*encoder, quantizer, and decoder, to form the codec. The typical image encoding stan-

dards, *e.g.*, JPEG [73], and JPEG 2000 [64], generally rely on handcrafted image transformation and separate optimization on codecs and thus are suboptimal for image compression. Moreover, JPEG and JPEG 2000 perform poor for low rate image compression and may introduce visible artifacts such as blurring, ringing, and blocking [64, 73].

Recently, deep convolutional networks (CNNs) have achieved unprecedented success in versatile vision tasks [3, 20, 23, 34, 40, 53, 81, 83]. As to image compression, CNN is also expected to be more powerful than JPEG and JPEG 2000 by considering the following reasons. First, for image encoding and decoding, flexible nonlinear analysis and synthesis transformations can be easily deployed by stacking multiple convolution layers. Second, it allows us to jointly optimize the nonlinear encoder and decoder in an end-to-end manner. Several recent advances also validate the effectiveness of deep learning in image compression [11, 67, 68, 69]. However, there are still several issues to be addressed in CNN-based image compression. In general, lossy image compression can be formulated as a joint rate-distortion optimization to learn encoder, quantizer, and decoder. Even the encoder and decoder can be represented as CNNs and optimized via back-propagation, the learning with non-differentiable quantizer is still a challenging issue. Moreover, the whole compression system aims to jointly minimize both the compression rate and distortion, where the entropy rate should also be estimated and minimized in learning. As a result of quantization, the entropy rate defined on discrete codes is also a discrete function and requires continuous approximation.

In this chapter, we present a novel CNN-based image compression framework to address the issues raised by quantization and entropy rate estimation. The existing deep learning-based compression models [11, 68, 69] allocate the same number of codes for each spatial position, and the discrete code used for decoder has the same length with the encoder output. That is, the length of the discrete code is spatially invariant. However, it is generally known that the local informative content is spatially variant in an image or video [82]. Thus, the bit rate should also be spatially variant to adapt to local informative content. To this end, we introduce a content-weighted importance map to guide the allocation of local bit rate. Given an input image $\mathbf{x}$, let $\mathbf{e} = E(\mathbf{x}) \in \mathbb{R}^{n \times h \times w}$ be the output of encoder network,

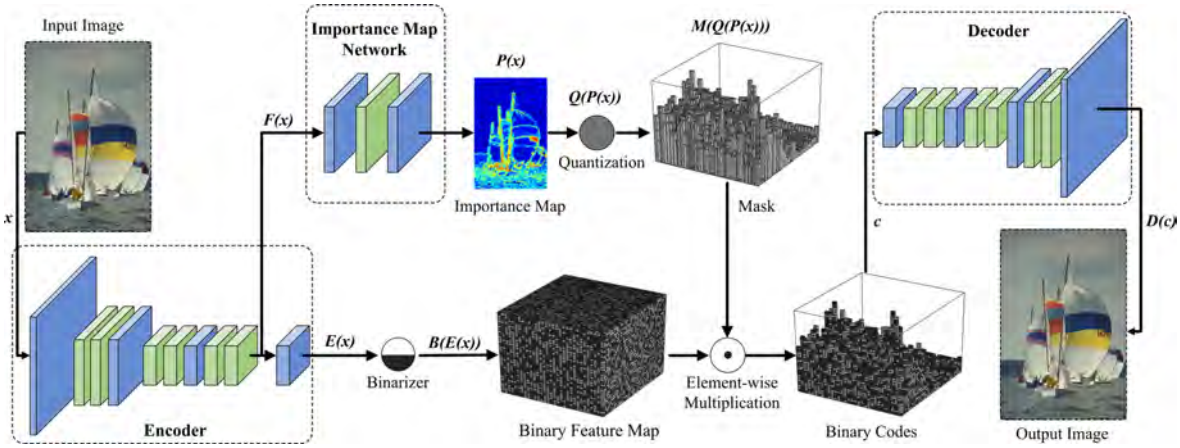


Figure 2.1: Illustration of the CNN architecture for content-weighted image compression.

which includes n feature maps with size of $h \times w$. $\mathbf{p} = P(\mathbf{x})$ denotes the $h \times w$ non-negative importance map. Specifically, when $\frac{l-1}{L} \leq \mathbf{p}_{i,j} < \frac{l}{L}$, we will only encode and save the first $\frac{nl}{L}$-th bits $\{\mathbf{e}_{1ij}, ..., \mathbf{e}_{\frac{nl}{L}ij}\}$ at spatial location (i, j). Here, L is the number of the importance level, and $\frac{n}{L}$ is the number of bits for each importance level. The other bits $\{\mathbf{e}_{(\frac{nl}{L}+1)ij}, ..., \mathbf{e}_{nij}\}$ at (i, j) are automatically set to 0 and need not to be saved into the codes. In this way, we can allocate more bits to the region with rich content, which is very helpful in preserving texture details with less sacrifice of bit rate. Moreover, the sum of the importance map $\sum_{i,j} \mathbf{p}_{i,j}$ naturally serves as a continuous estimation of the compression rate and can be directly adopted as a compression rate controller.

Benefited from the importance map, we do not require to use any entropy rate estimation in training the encoder and decoder, and can adopt a simple binarizer for quantization. The binarizer sets those features with the sigmoid outputs which are higher than 0.5 to 1 and the others to 0. Inspired by the binarized CNN [17, 55, 86], we introduce a proxy function to approximate the binary operation in backward propagation. As shown in Figure 2.1, our compression framework consists of four major components: convolutional encoder, importance map network, binarizer, and convolutional decoder. With the introduction of the continuous importance map and proxy function, all the components can be jointly optimized in an end-to-end manner.

Note that we do not include any entropy rate estimates in the training of the compres-

sion system. And the local spatial context of the codes is not utilized. Therefore, we design a convolutional entropy coder to predict the current code from its context, and apply it to the context-adaptive binary arithmetic coding (CABAC) framework [44] to further compress the binary codes and importance map.

Our whole framework is trained on a subset of the ImageNet database [19] and tested on the Kodak dataset. In low bit-rate image compression, our system achieves much better rate-distortion performance than JPEG and JPEG 2000 in terms of both SSIM metric and visual quality. More remarkably, the compressed images by our system are visually more pleasing with sharp edges, rich textures, and fewer artifacts. Compared with other CNN-based systems [11], ours performs favorably in retaining texture details while suppressing visual artifacts.

To sum up, the main contribution of this chapter is to introduce the content-weighted importance map and binary quantization into the image compression system. The importance map not only can be used to substitute the entropy rate estimate in joint rate-distortion optimization but also can be adopted to guide the local bit rate allocation. With binary quantization and the proxy function, our compression system can be end-to-end trained, and obtain notable improvement on visual quality over JPEG and JPEG 2000.

2.2 Content-weighted Image Compression

As illustrated in Figure 2.1, our content-weighted image compression framework is composed of four components, *i.e.*convolutional encoder, binarizer, importance map network, and convolutional decoder. Given an input image $\mathbf{x}$, the convolutional encoder defines a nonlinear analysis transformation by stacking convolution layers and outputs $E(\mathbf{x})$. The binarizer $B(E(\mathbf{x}))$ assigns 1 to the encoder outputs which are higher than 0.5, and 0 to the others. The importance map network takes the intermediate feature maps of the encoder as an input and yields the content-weighted importance map $P(\mathbf{x})$. The rounding function is adopted to quantize $P(\mathbf{x})$ and then a mask $M(P(\mathbf{x}))$ which has the same size of $B(E(\mathbf{x}))$ is generated with the guidance of the quantized $P(\mathbf{x})$. The binary code is then trimmed based

on $M(P(\mathbf{x}))$. Finally, the decoder defines a nonlinear synthesis transformation to produce a decoding result of $\hat{\mathbf{x}}$. In the following, we first introduce the four components and then present the formulation and learning of our model.

2.2.1 Components and Gradient Computation

Convolutional encoder and decoder

Both the encoder and decoder in our framework are fully convolutional networks and can be trained by back-propagation. The encoder network consists of three convolution layers and three residual blocks. Following [27], each residual block has two convolution layers. Analogous to [40] in single-image super-resolution, we remove the batch normalization operations from the residual blocks, and empirically find that it helps suppress visual compression artifacts in smooth areas. The input image $\mathbf{x}$ is first convolved with 128 filters with size 8×8 and stride 4 and followed by one residual block. The feature maps are then convolved with 256 filters with size 4×4 and stride 2 and followed by two residual blocks to output the intermediate feature maps $F(\mathbf{x})$. Finally, $F(\mathbf{x})$ is convolved with m filters with size 1×1 to yield the encoder output $E(\mathbf{x})$. It should be noted that we set $n = 64$ for low compression rate models with less than 0.5 bpp, and $n = 128$ otherwise.

The network architecture of decoder $D(\mathbf{c})$ is symmetric to that of the encoder, where $\mathbf{c}$ is the code of an image $\mathbf{x}$. To upsample the feature maps, we adopt the depth to space operation mentioned in [69]. Please refer to our project webpage[1] for more details on the network architecture of the encoder and decoder.

Binarizer

Since sigmoid nonlinearity is adopted in the last convolution layer of the encoder, the encoder output $\mathbf{e} = E(\mathbf{x})$ should be in the range of $[0, 1]$. e_{ijk} denotes an element in $\mathbf{e}$. The binarizer is defined as

$$B(e_{ijk}) = \begin{cases} 1, & \text{if } e_{ijk} > 0.5, \\ 0, & \text{if } e_{ijk} \leq 0.5. \end{cases} \tag{2.1}$$

[1] http://www2.comp.polyu.edu.hk/∼15903062r/content-weighted-image-compression.html

However, the gradient of the binarizer function $B(e_{ijk})$ is zero almost everywhere except that it is infinite when $e_{ijk} = 0.5$. In the back-propagation algorithm, the gradient is computed layer by layer with the chain rule. Thus, such setting makes any layers before the binarizer (*i.e.*, the whole encoder) never be updated during training.

Fortunately, some recent works on binarized neural networks (BNN) [17, 55, 86] have studied the issue of propagating gradient through binarization. Based on the straight-through estimator on the gradient [17], we introduce a proxy function $\tilde{B}(e_{ijk})$ to approximate $B(e_{ijk})$. Here, $B(e_{ijk})$ is still used in forward propagation calculation, while $\tilde{B}(e_{ijk})$ is used in back-propagation. Inspired by BNN, we adopt a piecewise linear function $\tilde{B}(e_{ijk})$ as the proxy of $B(e_{ijk})$,

$$\tilde{B}(e_{ijk}) = \begin{cases} 1, & \text{if } e_{ijk} > 1, \\ e_{ijk}, & \text{if } 1 \leq e_{ijk} \leq 0, \\ 0, & \text{if } e_{ijk} < 0. \end{cases} \tag{2.2}$$

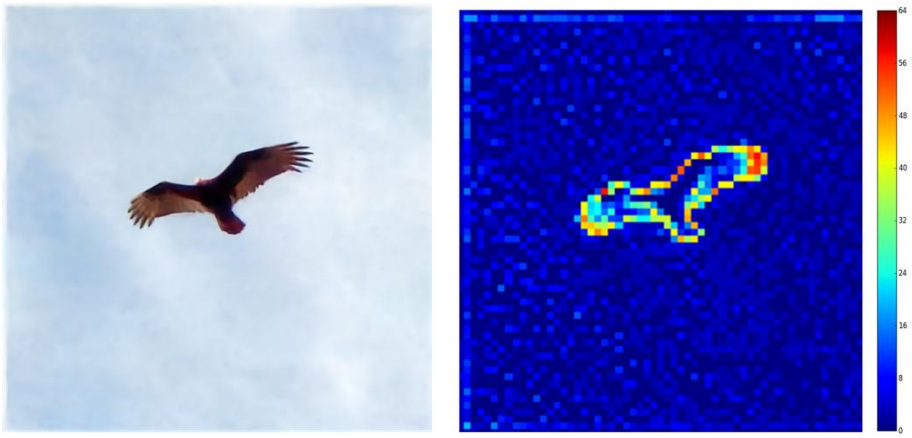
Then, the gradient of $\tilde{B}(e_{ijk})$ can be easily obtained by,

$$\tilde{B}'(e_{ijk}) = \begin{cases} 1, & \text{if } 1 \leq e_{ijk} \leq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{2.3}$$

Importance map

In [11, 67], the code length after quantization is spatially invariant, and entropy coding is then used to further compression the code. The difficulty in compressing different parts of an image should be different. The smooth regions in an image are easier to be compressed than those with salient objects or rich textures. Thus, fewer bits should be allocated to the smooth regions while more bits should be allocated to the regions with complex structures and details. For example, given an image with an eagle flying in the blue sky in Figure 2.2, it is reasonable to allocate more bits to the eagle and fewer bits to the blue sky. Moreover, when the whole code length for an image is limited, such an allocation scheme can also be used for rate control.

We introduce a content-weighted importance map for bit allocation and compression rate control. It is a feature map with only one channel, and its size should be the same as that of the encoder output. The value of the importance map is in the range of $(0, 1)$. An



Source Image Importance Map

Figure 2.2: Illustration of importance map. The regions with sharp edges or rich textures generally have higher values and should be allocated more bits.

importance map network is deployed to learn the importance map from an input image of $\mathbf{x}$. It takes the intermediate feature maps $F(\mathbf{x})$ from the last residual block of the encoder as input, and uses a network of three convolution layers to produce the importance map $\mathbf{p} = P(\mathbf{x})$.

Denote by $h \times w$ the size of the importance map $\mathbf{p}$, and n the number of feature maps of the encoder output. In order to guide the bit allocation, we should first quantize each element in $\mathbf{p}$ to an integer no more than n, and then generate an importance mask $\mathbf{m}$ with the size of $n \times h \times w$. Given an element p_{ij} in $\mathbf{p}$, the quantizer to importance map is defined as,

$$Q(p_{ij}) = l - 1, \text{if } \frac{l-1}{L} \leq p_{ij} < \frac{l}{L}, l = 1, \ldots, L. \tag{2.4}$$

where L is the importance levels and $(n \ mod \ L) = 0$. Each important level is corresponding to $\frac{n}{L}$ bits. As mentioned above, $p_{ij} \in (0, 1)$. Thus, $Q(p_{ij})$ has only L types of different quantity values i.e., $0, \ldots, L-1$. It should be noted that, $Q(p_{ij}) = 0$ indicates that zero bit will be allocated to this location, and all its information can be reconstructed based on its context in the decoding stage. In this way, the importance map can not only be treated as an alternative of entropy rate estimation but also naturally take the context into account.

With $Q(\mathbf{p})$, the importance mask $\mathbf{m} = M(\mathbf{p})$ can then be obtained by,

$$\mathbf{m}_{kij} = \begin{cases} 1, & \text{if } k \leq \frac{n}{L} Q(p_{ij}), \\ 0, & \text{else.} \end{cases} \tag{2.5}$$

The final coding result of the image $\mathbf{x}$ can then be represented as $\mathbf{c} = M(\mathbf{p}) \circ B(\mathbf{e})$, where $\circ$ denotes the element-wise multiplication operation. Note that the quantized importance map $Q(\mathbf{p})$ should also be considered in the code. Thus all the bits with $\mathbf{m}_{kij} = 0$ can be safely excluded from $B(\mathbf{e})$. Therefore, instead of n, only $\frac{n}{L} Q(p_{ij})$ bits are needed for each location (i, j). Besides, in video coding, just noticeable distortion (JND) models [82] have also been suggested for spatially variant bit allocation and rate control. Different from [82], the importance map is learned from training data via joint rate-distortion optimization.

Finally, in back-propagation, the gradient $\mathbf{m}$ with respect to p_{ij} should be computed. Unfortunately, due to the quantization operation and mask function, the gradient is zero almost everywhere. To address this issue, we rewrite the importance map m as a function of p,

$$\mathbf{m}_{kij} = \begin{cases} 1, & \text{if } \lceil \frac{kL}{n} \rceil < L p_{ij}, \\ 0, & \text{else} \end{cases} \tag{2.6}$$

where $\lceil . \rceil$ is the ceiling function. Analogous to binarizer, we also adopt a straight-through estimator of the gradient,

$$\frac{\partial \mathbf{m}_{kij}}{\partial p_{ij}} = \begin{cases} L, \text{ if } L p_{ij} - 1 \leq \lceil \frac{kL}{n} \rceil < L p_{ij} + 1, \\ 0, \text{ else.} \end{cases} \tag{2.7}$$

2.2.2 Model formulation and learning

Model formulation

In general, the proposed content-weighted image compression system can be formulated as a rate-distortion optimization problem. Our objective is to minimize the combination of the distortion loss and rate loss. A tradeoff parameter γ is introduced for balancing compression rate and distortion. Let $\mathcal{X}$ be a set of training data, and $\mathbf{x} \in \mathcal{X}$ be an image from the set. Therefore, the objective function our model is defined as

$$\mathcal{L} = \sum_{\mathbf{x} \in \mathcal{X}} \{ \mathcal{L}_D(\mathbf{c}, \mathbf{x}) + \gamma \mathcal{L}_R(\mathbf{x}) \} \tag{2.8}$$

where $\mathbf{c}$ is the code of the input image $\mathbf{x}$. $\mathcal{L}_D(\mathbf{c}, \mathbf{x})$ denotes the distortion loss and $\mathcal{L}_R(\mathbf{x})$ denotes the rate loss, which will be further explained as follows.

Distortion loss. Distortion loss is used to evaluate the distortion between the original image and the decoding result. Although better results may be obtained by assessing the distortion in the perceptual space, we simply use the squared ℓ_2 error to define the distortion loss,

$$\mathcal{L}_D(\mathbf{c}, \mathbf{x}) = \|D(\mathbf{c}) - \mathbf{x}\|_2^2. \tag{2.9}$$

Rate loss. Instead of entropy rate, we define the rate loss directly on the continuous approximation of the code length. Suppose the size of encoder output $E(\mathbf{x})$ is $n \times h \times w$. The code by our model includes two parts: (i) the quantized importance map $Q(\mathbf{p})$ with the fixed size $h \times w$; (ii) the trimmed binary code with the size $\frac{n}{L}\sum_{i,j} Q(p_{ij})$. Note that the size of $Q(\mathbf{p})$ is constant to the encoder and importance map network. Thus $\frac{n}{L}\sum_{i,j} Q(p_{ij})$ can be used as rate loss.

Due to the effect of quantization $Q(p_{ij})$, the function $\frac{n}{L}\sum_{i,j} Q(p_{ij})$ cannot be optimized by back-propagation. Thus, we relax $Q(p_{ij})$ to its continuous form, and use the sum of the importance map $\mathbf{p} = P(\mathbf{x})$ as rate loss,

$$\mathcal{L}_R^0(\mathbf{x}) = \sum_{i,j} (P(\mathbf{x}))_{ij}. \tag{2.10}$$

For better rate control, we can select a threshold r, and penalize the rate loss in Eqn. (2.10) only when it is higher than r. Then we define the rate loss in our model as,

$$\mathcal{L}_R(\mathbf{x}) = \begin{cases} \sum_{i,j} (P(\mathbf{x}))_{ij} - r, & \text{if } \sum_{i,j} (P(\mathbf{x}))_{ij} > r \\ 0, & \text{otherwise.} \end{cases} \tag{2.11}$$

The threshold r can be set based on the code length for a given compression rate. By this way, our rate loss will penalize the code length higher than r, and makes the learned compression system achieve the comparable compression rate around the given one.

Learning

Benefited from the relaxed rate loss and the straight-through estimator of the gradient, the whole compression system can be trained in an end-to-end manner with an ADAM solver [33].

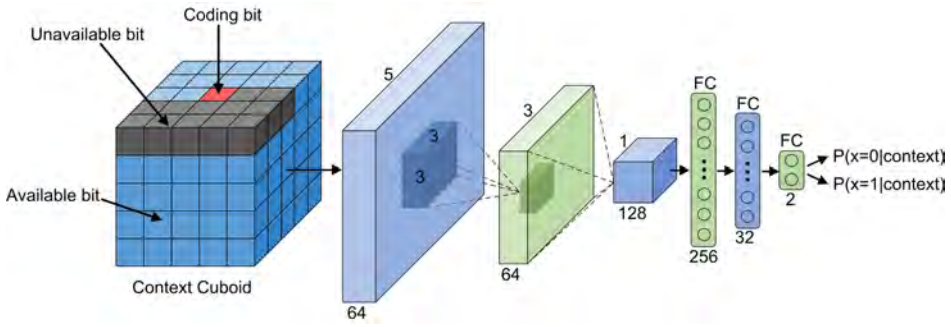


Figure 2.3: The CNN for convolutional entropy encoder. The red block represents the bit to predict; dark blocks mean unavailable bits; blue blocks represent available bits.

We initialize the model with the parameters pre-trained on the training set $\mathcal{X}$ without the importance map. The model is further trained with the learning rate of $1e^{-4}$, $1e^{-5}$ and $1e^{-6}$. In each learning rate, the model is trained until the objective function does not decrease.

2.3 Convolutional entropy encoder

Due to no entropy constraint is included, the entropy of the code generated by the compression system in Sec. 2.2 is not maximal. This provides some leeway to further compress the code with lossless entropy coding. Generally, there are two kinds of entropy compression methods, *i.e.*Huffman tree, and arithmetic coding [79]. Among them, arithmetic coding can exhibit a better compression rate with a well-defined context and is adopted in this work.

2.3.1 Encoding binary code

The binary arithmetic coding is applied according to the CABAC [44] framework. Note that CABAC is originally proposed for video compression. Let $\mathbf{c}$ be the code of n binary bitmaps, and $\mathbf{m}$ be the corresponding importance mask. To encode $\mathbf{c}$, we modify the coding schedule, redefine the context, and use CNN for probability prediction. As to the coding schedule, we simply code each binary bit map from left to right and row by row and skip those bits with the corresponding important mask value of 0.

Context modeling. Denote by c_{kij} a binary bit of the code $\mathbf{c}$. We define the context of c_{kij} as $CNTX(c_{kij})$ by considering the binary bits both from its neighborhood and from

the neighboring binary code maps. Specifically, $\mathit{CNTX}(c_{kij})$ is a $5 \times 5 \times 4$ cuboid. We further divide the bits in $\mathit{CNTX}(c_{kij})$ into two groups: the available and unavailable ones. The available ones represent those can be used to predict c_{kij}. While the unavailable ones include: (i) the bit to be predicted c_{kij}, (ii) the bits with the importance map value 0, (iii) the bits out of boundary and (iv) the bits currently not coded due to the coding order. Here we redefine $\mathit{CNTX}(c_{kij})$ by: (1) assigning 0 to the unavailable bits, (2) assigning 1 to the available bits with value 0, and (3) assigning 2 to the available bits with value 1.

Probability prediction. One usual method for probability prediction is to build and maintain a frequency table. As to our task, the size of the cuboid is too large to build the frequency table. Instead, we introduce a CNN model for probability prediction. As shown in Figure 2.3, the convolutional entropy encoder $En(\mathit{CNTX}(c_{kij}))$ takes the cuboid as input, and outputs the probability that the bit c_{kij} is 1. Thus, the loss for learning the entropy encoder can be written as,

$$\begin{aligned}\mathcal{L}_E = \sum_{i,j,k} m_{kij} \{ & c_{kij} \log_2(En(\mathit{CNTX}(c_{kij}))) \\ & +(1 - c_{kij}) \log_2(1 - En(\mathit{CNTX}(c_{kij})))\} \,. \end{aligned} \tag{2.12}$$

where $\mathbf{m}$ is the importance mask. The convolutional entropy encoder is trained using the ADAM solver on the contexts of binary codes extracted from the binary feature maps generated by the trained encoder. The learning rate decreases from $1e^{-4}$ to $1e^{-6}$ as we do in Sec. 2.2.

2.3.2 Encoding quantized importance map

We also extend the convolutional entropy encoder to the quantized importance map. To utilize binary arithmetic coding, several binary code maps are adopted to represent the quantized importance map. The convolutional entropy encoder is then trained to compress the binary code maps.

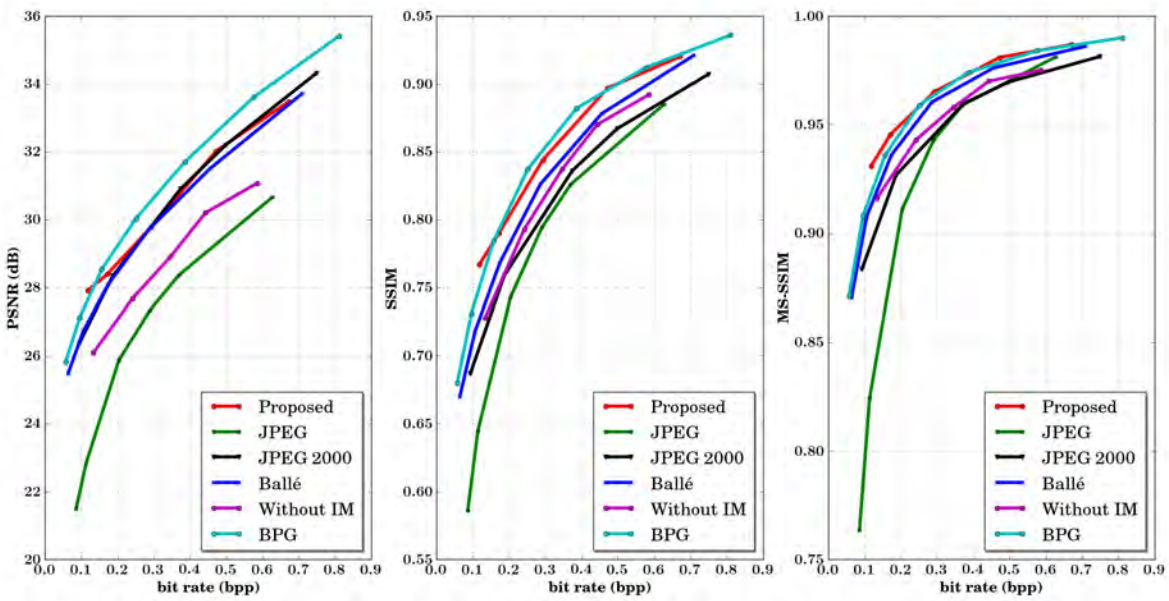


Figure 2.4: Comparison of the rate-distortion curves by different methods: (a) PSNR, (b) SSIM, and (c) MS-SSIM. "Without IM" represents the proposed method without importance map.

2.4 Experiments

Our content-weighted image compression models are trained on a subset of ImageNet [19] with about $10,000$ high quality images. We crop these images into 128×128 patches and make use of these patches to train the network. After training, we test our model on the Kodak PhotoCD image dataset with the metrics for lossy image compression. The compression rate of our model is evaluated by the metric bits per pixel (bpp), which is calculated as the total amount of bits used to code the image divided by the number of pixels. The image distortion is evaluated with Multi-Scale Structure Similarity (MS-SSIM), Peak Signal-to-Noise Ratio (PSNR), and the structural similarity (SSIM) index. For the time complexity, it takes about 0.48 second to compress an image in Kodak dataset.

In the following, we first introduce the parameter setting of our compression system. Then both quantitative metrics and visual quality evaluation are provided. Finally, we further analyze the effect of importance map and convolutional entropy encoder on the compression system.

2.4.1 Parameter setting

In our experiments, we set the number of binary feature maps n according to the compression rate, *i.e.*64, when the compression rate is less than 0.5 bpp and 128 otherwise. Then, the number of importance levels is chosen based on m. For $n = 64$ and $n = 128$, we set the number of importance level L to be 16 and 32, respectively. Moreover, different values of the tradeoff parameter γ in the range $[0.0001, 0.2]$ are chosen to get different compression rates. We note that, γ larger than 0.2 will lead to models with quit large distortions, which are meaning less in real applications. For the choice of the threshold value r, we just set it as r_0hw for $n = 64$ and $0.5r_0hw$ for $n = 128$. r_0 is the wanted compression rate represented with bit per pixel (bpp).

2.4.2 Quantitative evaluation

For quantitative evaluation, we compare our model with JPEG [73], JPEG 2000 [64], BPG and the CNN-based method by Ballé *et al*. [11]. Among the different variants of JPEG, the optimized JPEG with 4:2:0 chroma sub-sampling is adopted. For a fair comparison, all the results by Ballé [11], JPEG, and JPEG2000 on the Kodak dataset are downloaded from `http://www.cns.nyu.edu/~lcv/iclr2017/`.

Using MS-SSIM [77], SSIM [76] and PSNR as performance metrics, Figure 2.4 gives the rate-distortion curves of these five methods. In terms of PSNR, BPG has the best performance. And the results by JPEG 2000, Ballé [11] and ours are very similar but are much higher than that by JPEG. In terms of SSIM and MS-SSIM, our system has similar performance with BPG and outperforms all the other three competing methods, including JPEG, JPEG 2000, and Ballé [11]. Due to SSIM and MS-SSIM is more consistent with human visual perception than PSNR, these results indicate that our system performs favorably in terms of visual quality.

2.4.3 Visual quality evaluation

We further compare the visual quality of the results by JPEG, JPEG 2000, Ballé [11], BPG and our system in low compression rate setting. Figure 2.5 shows the original images and the results produced by the five compression systems. Visual artifacts, *e.g.*, blurring, ringing, and blocking, usually are inevitable in the compressed images by traditional image compression standards such as JPEG and JPEG 2000, while blurring and ringing effect can still be observed from the results by BPG. And these artifacts can also be perceived in the second and third columns of Figure 2.5. Even Ballé [11] is effective in suppressing these visual artifacts. In Figure 2.5, from the results produced by Ballé [11], we can observe the blurring artifacts in row 1, 2, 3, and 5, the color distortion in row 2 and 5, and the ringing artifacts in row 2. By contrast, the results produced by our system exhibit much less noticeable artifacts and are visually much more pleasing.

From Figure 2.5, Ballé [11] usually produces the results by blurring the strong edges or over-smoothing the small-scale textures. Specifically, in row 5 most details of the necklace have been removed by Ballé [11]. One possible explanation may be that before entropy encoding it adopts a spatially invariant bit allocation scheme. It is natural to see that more bits should be allocated to the regions with strong edges or detailed textures while less to the smooth regions. By contrast, an importance map is introduced in our system to guide spatially variant bit allocation. Moreover, instead of handcrafted engineering, the importance map is end-to-end learned to minimize the rate-distortion loss. As a result, our model is very promising in keeping perceptual structures, such as sharp edges and detailed textures.

2.4.4 Experimental analyses on important map

To assess the role of importance map, we train a baseline model by removing the importance map network from our framework. Both entropy and importance map based rate loss are not included in the baseline model. Thus, the compression rate is controlled by modifying the number of binary feature maps. Figure 2.4 also provides the ratio-distortion curves of the baseline model. We can see that, the baseline model is inferior to JPEG 2000 and Ballé [11]

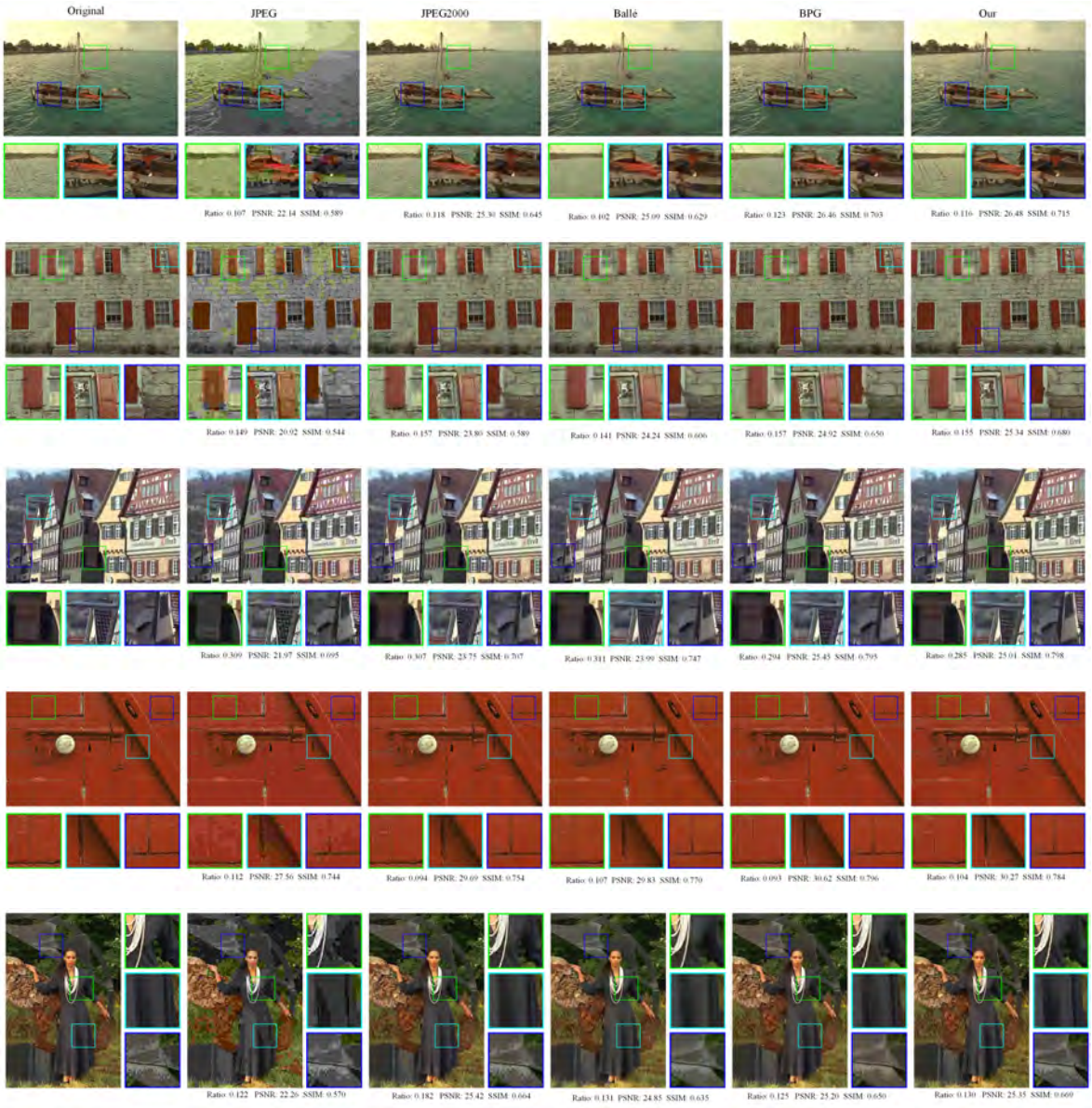


Figure 2.5: Images produced by different compression systems at different compression rates. From the left to right: groundtruth, JPEG, JPEG 2000, Ballé [11], BPG and ours. Our model achieves the best visual quality at each rate, demonstrating the superiority of our model in preserving both sharp edges and detailed textures. (Best viewed on screen in color)

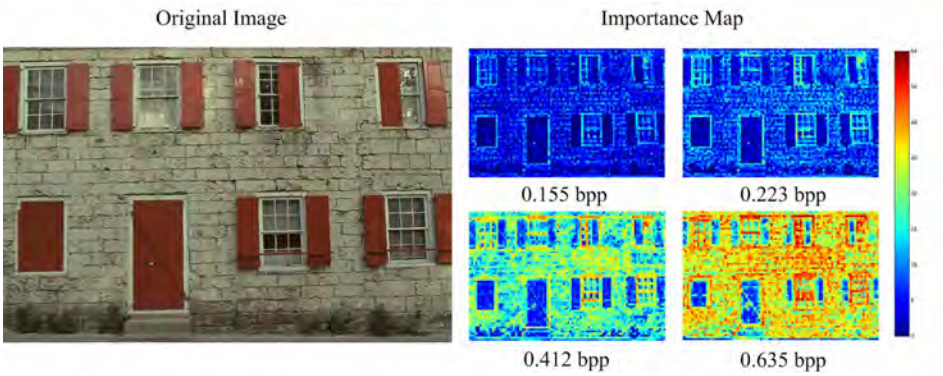


Figure 2.6: The important maps obtained at different compression rates. The right color bar shows the palette on the number of bits.

in terms of MS-SSIM, PSNR, and SSIM, thereby validating the necessity of importance map for our model. Using the image in row 5 of Figure 2.5, the compressed images by our model with and without importance map are also shown on our project webpage. More detailed textures and better visual quality can be obtained by using the importance map.

Figure 2.6 shows the importance map obtained at different compression rates. We can see that when the compression rate is low, due to the overall bit length is very limited, the importance map only allocates more bits to salient edges. With the increasing of compression rate, more bits will be allocated to weak edges and mid-scale textures. Finally, when the compression rate is high, small-scale textures will also be allocated with more bits. Thus, the importance map learned in our system is consistent with human visual perception, which may also explain the superiority of our model in preserving the structure, edges, and textures.

2.4.5 Entropy encoder evaluation

The model in Sec. 2.2 does not consider the entropy of the codes, allowing us to further compress the code with a convolutional entropy encoder. Here, two groups of experiments are conducted. First, we compare four variants of our model: (i) the full model, (ii) the model without entropy coding, (iii) the model by only encoding binary codes, and (iv) the model by only encoding importance map. From Figure 2.7(a), both the binary codes and importance map can be further compressed by using our convolutional entropy encoder. And our full model can achieve the best performance among the four variants. Second, we com-

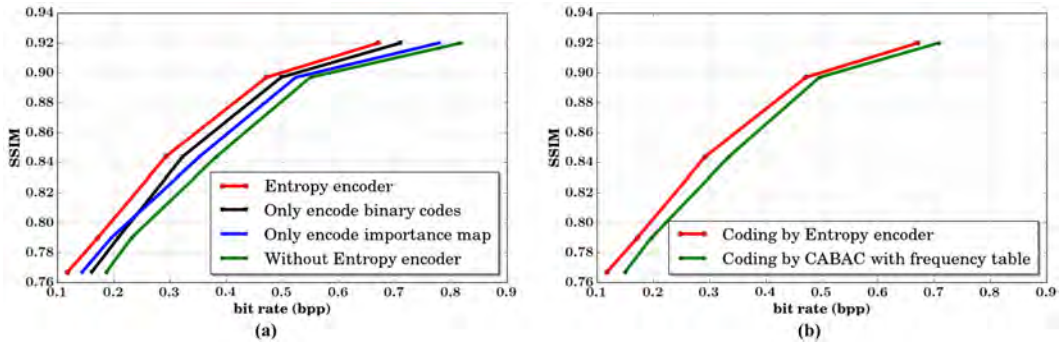


Figure 2.7: Performance of convolutional entropy encoder: (a) for encoding binary codes and importance map, and (b) by comparing with tradition CABAC.

pare our convolutional entropy encoder with the traditional content-based arithmetic coding (CABAC) with a small context (*i.e.*the 5 bits near the bit to encode). As shown in Figure 2.7(b), our entropy encoder can take larger context into account and performs better than CABAC. Besides, we also note that our method with either CABAC or convolutional encoder can outperform JPEG 2000 in terms of SSIM.

2.5 Conclusion

A CNN-based system is developed for content weighted image compression. With the importance map, we suggest a non-entropy based loss for rate control. Spatially variant bit allocation is also allowed to emphasize the salient regions. Using the straight-through binary estimator, our model can be trained in an end-to-end manner. A convolutional entropy encoder is introduced to further compress the binary codes and the importance map. Experiments clearly show the superiority of our model in retaining structures and removing artifacts, leading to favorably visual quality.

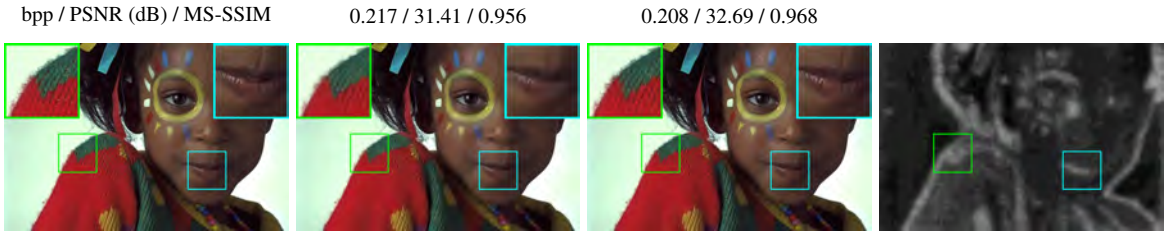
CHAPTER 3

IMPROVED CONTENT-WEIGHTED DEEP IMAGE COMPRESSION

In this chapter, we extend the content-weighted image compression framework with the following improvements: improved network structure for the transforms, learnable multi-value adaptive quantization function, better learning strategy for importance map, and a novel convolutional network for post entropy coding. With the improvement, the proposed framework can produce much better results than that of chapter 2. Furthermore, we imply the framework for task-driven image compression, where the importance map is only related to the content of the image but also the performance of the specific tasks, such as objective detection or face recognition.

3.1 Introduction

Inspired by the unprecedented success of deep learning, deep image compression models recently have received considerable attention from the vision and learning communities. Traditional image encoding standards, e.g., JPEG [73], JPEG 2000 [64], and BPG (intra-frame encoding of HEVC) [65], generally adopt handcrafted transform and separate optimization on codecs, thus are limited in compression performance and are inflexible in adapting to image content and tasks. In comparison, deep networks provide a flexible and end-to-end manner to learn nonlinear analysis and synthesis transforms jointly by optimizing rate-distortion performance, thereby expectantly surpassing existing codecs by compression performance and visual quality. Moreover, recent years have witnessed the consistent progress and pursuit for the acquisition and sharing of higher definition images and videos. And the population of smartphones and the Internet increases the burden on storage and network bandwidth, which further makes it demanding to develop better image compression methods.



(a) Original (b) Ballé et al. [11] (c) Ours(MSE) (d) Importance map

Figure 3.1: Decoding images of deep compression approach Ballé et al. [11] and our content-weighted image compression method.

Learning-based lossy image compression is usually formulated as a joint rate-distortion optimization problem, and cannot be readily solved by deep networks. On the one hand, the quantization operation generally is indispensable to generate discrete codes. However, its gradient is zero almost everywhere except it is infinite for several threshold points, making it challenging to jointly optimize encoder and decoder via back-propagation. To handle this issue, several continuous proxy methods have been presented, including variational relaxation [11, 68], smoothed [67] and soft-to-hard approximation [2]. On the other hand, for joint rate-distortion optimization, a rate loss usually is introduced for modeling the entropy of quantized codes, and also requires continuous approximation [2, 11, 67].

Albeit their significant progress, existing learning-based image compression methods are still limited in modeling and exploiting spatial variation and dependency of image content. First, spatially invariant bit length allocation is generally adopted in existing methods, whereas undoubtedly the content of an image is spatially variant. That is, the regions with complex and salient structures generally are more essential to constitute an image. For example, it can be seen from Fig. 3.1 that Ballé et al. [11] fail to recover the details of the mouth and sweater at lower bits per pixel (bpp). With the guidance of the importance map (see Fig. 3.1(d)) learned from the image content, our method can assign more bits to encode the areas with more details and structural information and thus can generate visually better outputs (see Fig. 3.1(c)). Second, the entropy is usually calculated by assuming that quantized codes follow some specific form of distribution. The mismatch between the assumption and the real distribution will inevitably hurt compression performance. Last but not least, the codes after quantization are still spatially dependent. Ballé et al. [11] simply adopt the context-

based adaptive binary arithmetic coding framework (CABAC) [44] for entropy encoding. Considering the practical feasibility of maintaining conditional probability tables, CABAC only exploits the two nearest codes as context. Nonetheless, better compression performance can be attained by incorporating a larger context in entropy encoding.

To address the above issues, we in this chapter take both spatial variation and dependency of image content into account and present a content-weighted encoder-decoder model for deep image compression (i.e., CWIC). For handling spatially variant image content, the encoder of our CWIC involves an encoder subnet, a learned quantization operation, and an importance map subnet. In particular, the quantization operation is learned to quantize each element from one of the n feature maps into T discrete levels by minimizing the quantization error. And the importance map subnet is deployed to estimate the informative importance $p_{i,j}$ of local image content at location (i, j). Specifically, when $\frac{l}{L} \leq p_{i,j} < \frac{l+1}{L}$, we only encode and save the first $\frac{nl}{L}$ channels at spatial location (i, j), where L is the number of the importance level. The learned importance map can be exploited to produce the importance masks $\mathbf{m}$ for guiding locally adaptive bit rate allocation. Codes with the importance mask $m_{kij} = 1$ are saved, while those $m_{kij} = 0$ are ignored. Thus, we can allocate more codes to the region with rich and salient structures and less codes to the smooth region, thereby benefiting the reconstruction of texture details with less sacrifice of bit rate (see Fig. 3.1(c)). Moreover, the summation of importance map $\sum_{k,i,j} m_{kij}$ naturally serves as an estimation of the compression rate, and our CWIC model can be learned without any assumption on the distribution of quantized codes.

For exploiting local spatial context, we adopt the arithmetic coding framework, and present a convolutional entropy prediction model to predict the current symbol from its context for quantized codes as well as importance map. Existing methods generally suffer from either storage or computational burden, and are limited in large context modeling and compression performance. To tackle this dilemma, we present a trimmed convolutional network for arithmetic encoding (TCAE), where convolutional kernels are specially trimmed to respect the compression order and context dependency. Then, the probability prediction of all symbols can be efficiently performed in one single forward pass via a fully convolutional

network. By stacking several trimmed convolution layers, TCAE can model a large context while maintaining computational efficiency. Furthermore, an inclined TCAE model is presented to divide the codes from a 3D code map into several inclined planes. Parallel decoding can then be safely conducted to the symbols inner each inclined plane, thereby significantly speeding up the decoding process.

Experiments are conducted to evaluate our CWIC model on the Kodak PhotoCD image dataset[1] and the Tecnick dataset[2]. In terms of MS-SSIM, our CWIC clearly outperforms existing image encoding standards, i.e., JPEG [64], JPEG 2000 [73] and BPG [65], and several deep image compression methods, e.g., [11, 32, 46, 58]. In terms of PSNR, our CWIC performs on par with BPG [65] and surpasses the other competing methods. As for visual quality, our CWIC is promising in retaining fine salient details and suppressing visual artifacts in comparison to the competing methods.

This chapter is a substantial extension of our pioneer work in chapter 2 [38]. Compared with [38], an improved network structure combining dense blocks is introduced in encoder and decoder. And binary quantization is substituted by a learned channel-wise multi-value quantization for adaptive discretion of encoder feature. Moreover, a two-stage relaxation scheme is adopted for better learning of the importance map subnet. Finally, we introduce a TCAE as well as an inclined TCAE to effectively and efficiently model large context in arithmetic coding. The contributions of this work are summarized as follows:

- A content-weighted encoder-decoder model is introduced for lossy image compression. Here, a learned channel-wise quantization is deployed for the discretion of the encoder features, an importance map subnet is introduced to guide locally adaptive bit allocation, the summation of the generated importance mask is used as an estimation of compression rate, and a two-stage relaxation scheme is deployed for learning importance map subnet.

[1] http://r0k.us/graphics/kodak/
[2] https://testimages.org/

- A TCAE network is presented for large context modeling in arithmetic encoding. With trimmed convolution, the conditional probability of quantized codes can be efficiently predicted via the fully convolutional network. And an inclined TCAE model is further introduced to accelerate the decoding process.

- Experiments show that our method is effective in recovering salient structures and rich details while suppressing visual artifacts at lower bpp. Moreover, our method performs favorably in comparison to existing image encoding standards [64, 65, 73] and deep models [11, 32, 46, 58].

The remainder of this chapter is organized as follows. Sections 3.2 and 3.3 respectively present our CWIC and TCAE models for handling the spatial variation and dependency issues in image compression. Section 3.4 gives the experimental results, and Section 3.6 provides some concluding remarks.

3.2 Content-weighted Image Compression

In this section, we present a content-weighted encoder-decoder network for image compression (i.e., CWIC). To begin with, we first describe the network structure of our CWIC, including encoder, decoder, and importance map subnets. Then, distortion and rate losses are defined on the decoding image as well as importance map. Finally, to ease the difficulty caused by quantization in CWIC, continuous relaxations are introduced for learning the encoder and importance map subnets.

3.2.1 Network Architecture

As illustrated in Fig. 3.2, our CWIC network consists of three subnetworks, i.e., encoder, importance map and decoder subnets. In particular, the encoder is further divided into the shared and encoding-specific parts. To generate discrete codes, quantization operations are deployed to the outputs of encoder and importance map subnets. In the following, we introduce the main network components, quantization operations, and encoding/decoding procedure.

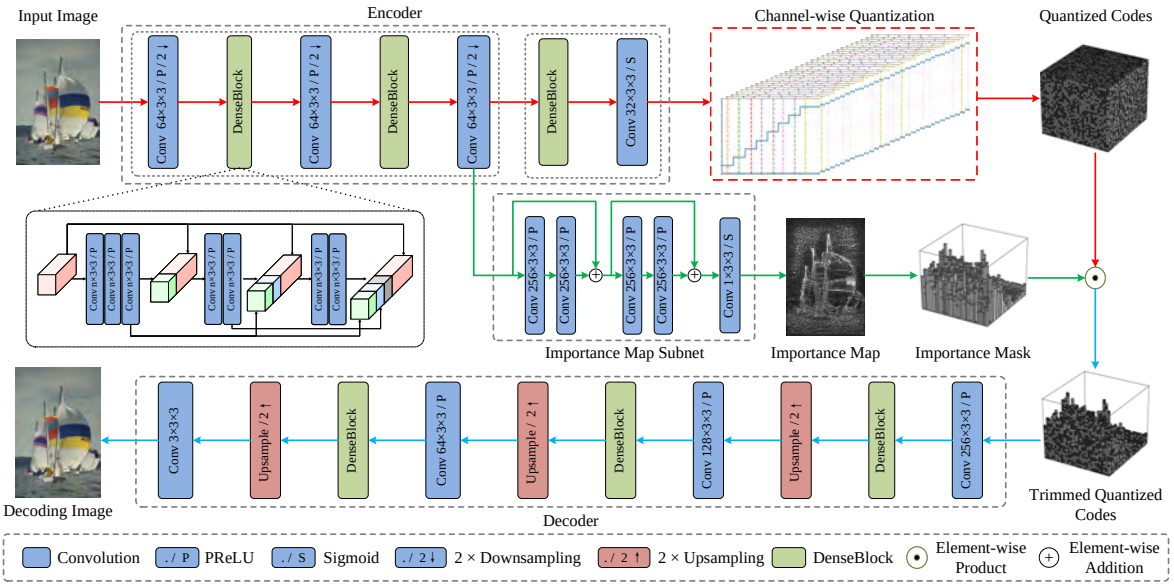


Figure 3.2: Illustration of our content weighted image compression model. The whole framework involves an encoder, a learned channel-wise multi-valued quantization, an importance map subnet, and a decoder. The encoder produces 32 feature maps which are further quantized by the channel-wise multi-valued quantization function to generate quantized codes. The importance map subnet estimates the informative importance of local image content and generate an importance map with only 1 channel. With the quantized importance map, an importance mask is further generated for guiding spatially variant bit rate allocation. By multiplying quantized codes with importance mask in an element-wise manner, the trimmed quantized codes are produced as the input of the decoder to generate the decoding image.

Encoder and decoder subnets

Given an image $\mathbf{x}$, the encoder subnet $E(\mathbf{x})$ is comprised of a shared part E_s and an encoding-specific part E_p. Concretely, E_s has three convolution layers with stride 2. And the feature map channels of the three layers are 64, 128, and 256, respectively. Moreover, a dense block is deployed right after each of the first two strided convolution layers to increase the nonlinearity of the encoder. For E_p, it only contains one dense block. From Fig. 3.2, each dense block involves three sub-blocks, where the first sub-block consists of three convolution layers and each of the other two sub-blocks consists of two convolution layers. Following DenseNet [30], skip connections are introduced from any sub-block to all successive sub-blocks to improve information flow and ease the training of deep networks, thereby benefiting compression performance. Analogous to [40] in SISR, we remove the batch nor-

malization operations from the sub-blocks and empirically find that it helps suppress visual compression artifacts in smooth areas. After the dense block in E_p, we further add an extra convolution layer with sigmoid nonlinearity to reduce the channels to 32 and constrain the output within the interval $(0, 1)$. For simplicity, 3×3 convolution is adopted in all convolution layers.

The structure of the decoder subnet $D(\mathbf{c})$ is a mirror of the encoder. In particular, the convolution layer with stride 2 in the encoder subnet is substituted by an upsampling convolution layer, which involves a basic convolution layer followed by a depth-to-width operation [69] in the decoder. And the last convolution layer produces the decoding image with 3 channels and linear activation is used.

Importance map subnet

In general, an image conveys spatially variant informative content. From Fig. 3.3, the regions with the house are more salient and content-intensive, while the regions with the sky are simple and contain little informative content. Most deep image compression methods [2, 11, 67] allocate spatially invariant code length and exploit entropy coding to further compress the codes. Although entropy coding can encode quantized codes into bit streams with different length, it is deployed after quantization operation, and cannot recover the information loss caused by spatially invariant bit allocation in quantization. As a result, such solutions usually are inferior in preserving the salient structure and fine details at lower bpp.

To alleviate this issue, we suggest utilizing spatially variant bit allocation which can be more advantageous by emphasizing salient structures of the image. As shown in Fig. 3.3, it is reasonable to allocate more bits to region *house* and fewer bits to region *sky*. Therefore, we introduce an importance map subnet to produce importance map from the input image. It can be seen from Fig. 3.3(b) that, the importance map provides a reasonable estimation of the informative importance of local image content, and can be exploited to guide locally adaptive bit rate allocation. In comparison to spatially invariant bit rate allocation, we can add more channels of feature maps and incorporate with importance map to preserve the more salient structure and fine details without the increase of code length. Note that the

(a) Original image (b) Importance map (0.213 bpp)

Figure 3.3: Illustration of the importance map. The regions with a sharp edge and rich texture generally have higher values and should be allocated with more bits.

code length at each position is controlled by the importance map. Thus, the summation of quantized importance map, i.e., importance mask, can serve as an estimation of compression rate, and be computed without any assumption on the distribution of quantized codes. It is worth noting that just noticeable distortion (JND) models [82] have also been suggested for spatially variant bit allocation and rate control in video coding. In contrast to JND [82], the importance map is learned from training data via rate-distortion optimization.

The architecture of the importance map subnet is shown in Fig. 3.2. In particular, it takes the intermediate feature map $E_s(\mathbf{x})$ as input, involves two residual blocks [28] and an extra convolution layer with sigmoid nonlinearity to produce the 1-channel importance map $\mathbf{p} = P(E_s(\mathbf{x}))$ which has the same spatial size, i.e., $h \times w$, as the encoder feature map $\mathbf{e}$ with the values in the range $(0, 1)$.

Quantization

Both the encoder feature map $\mathbf{e}$ and importance map $\mathbf{p}$ are continuous values in the range $(0, 1)$, and quantization is required. For $\mathbf{e}$, we adopt the channel-wise multi-valued quantization Q parameterized by $\Theta_k = \{s_{k,0}, \ldots, s_{k,t}, \ldots, s_{k,T-1}\}$, where the $s_{k,t}$ denotes non-

negative weights representing a quantization interval, and T is the number of quantization levels. With Θ_k, the t-th quantization center $q_{k,t}$ of the k-th channel is defined as,

$$q_{k,t} = \sum\nolimits_{t'=0}^{t} s_{k,t'} \ . \tag{3.1}$$

The quantized level $t^*(e_{kij})$ of an element e_{kij} of the k-th channel can be obtained by,

$$t^* = \arg\min_t \|e_{kij} - q_{k,t}\|^2, \ \ t = 0, \ldots, T-1. \tag{3.2}$$

Then, e_{kij} is discretely represented as $Q(e_{kij}) = q_{k,t^*}$, and its quantization index is represented as $QL(e_{kij}) = t^*$.

As for importance map $\mathbf{p}$, we define the following quantization function to quantize the importance value p_{ij} at position (i, j),

$$QI(p_{ij}) = l, \text{such as } \frac{l}{L} \leq p_{ij} < \frac{l+1}{L} \ , \ l \in \{0, \ldots, L-1\}, \tag{3.3}$$

where L is the number of quantization levels for importance map. We note that the quantized importance map is also required to be stored in our encoding scheme. Denote by n the number of channels of encoder feature map $\mathbf{e}$. Without loss of generality, we assume that $(n \mod L) = 0$.

For guiding spatially variant bit allocation, we further introduce a binary importance mask $M(\mathbf{p})$ with the same size as the quantized encoder feature map $Q(\mathbf{e})$. In particular, the (k, i, j)-th element m_{kij} of $\mathbf{m} = M(\mathbf{p})$ is defined as,

$$m_{kij} = \begin{cases} 1, & \text{if } k < \frac{n}{L} QI(p_{ij}), \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

Guided by $M(\mathbf{p})$, all the codes with $m_{kij} = 0$ are discarded from $Q(\mathbf{e})$. When $QI(p_{ij}) = 0$, no code needs to be stored at position (i, j), and all of its information is predicted from its context. To sum up, instead of $n \times h \times w$, only $\sum_i \sum_j \frac{n}{L} QI(p_{ij}) = \sum_{k,i,j} m_{kij}$ codes from $Q(\mathbf{e})$ need to be stored, and the summation of importance mask can thus be used as an indicator of compression rate.

Procedure of encoding and decoding

Finally, we summarize the procedure of encoding and decoding based on the encoder, importance map, and decoder subnets. Given an input image $\mathbf{x}$, the shared part of encoder subnet is first deployed to generate intermediate feature map $E_s(\mathbf{x})$. Then, both the encoder-specific part of encoder subnet and the importance map subnet take $E_s(\mathbf{x})$ as input to produce encoder feature map $\mathbf{e} = E_p(E_s(\mathbf{x}))$ and importance map $\mathbf{p} = P(E_s(\mathbf{x}))$, respectively.

By quantizing $\mathbf{e}$ and $\mathbf{p}$, we obtain the discrete encoder representation $Q(\mathbf{e})$, quantized importance map $QI(\mathbf{p})$, and binary importance mask $M(\mathbf{p})$. The encoding result of $\mathbf{x}$ can then be represented as $\mathbf{z} = M(\mathbf{p}) \circ Q(\mathbf{e})$, where $\circ$ denotes the element-wise product. The corresponding quantization index of $\mathbf{z}$ is represented as $\mathbf{o} = M(\mathbf{p}) \circ QL(\mathbf{e})$. Then $\mathbf{o}$ and $QI(\mathbf{p})$ are stored as the codes of $\mathbf{x}$. In the decoding stage, $\mathbf{z}$ is reconstructed by $z_{kij} = q_{k,o_{kij}}$ if $m_{kij} = 1$, otherwise $z_{kij} = 0$. and the decoder subnet takes $\mathbf{z}$ as input to obtain the decoding image $D(\mathbf{z})$.

3.2.2 Loss Functions

In general, both distortion and rate losses should be included in modeling the objective of content-weighted image compression. Moreover, a quantization loss is also introduced to guide the learning of channel-wise multi-valued quantization. In the following, we explain these loss functions and give the overall model objective.

Distortion loss. Distortion loss is used to measure the distortion between the input image and decoding image. Concretely, we consider two types of distortion metrics. The first is based on the mean squared error (MSE),

$$\mathcal{L}_{\text{MSE}}(\mathbf{z}, \mathbf{x}) = \frac{1}{3HW} \|D(\mathbf{z}) - \mathbf{x}\|_2^2 \,, \tag{3.5}$$

where H and W are the height and width of $\mathbf{x}$, respectively. The other is based on the multi-scale structural similarity (MS-SSIM) [77],

$$\mathcal{L}_{\text{MS-SSIM}}(\mathbf{z}, \mathbf{x}) = 100\,(1 - \text{MS-SSIM}\,(D(\mathbf{z}), \mathbf{x}))\ . \tag{3.6}$$

In our implementation, $\mathcal{L}_{\text{MS-SSIM}}$ is adopted as the default distortion loss $\mathcal{L}_D$, and we denote our method with $\mathcal{L}_{\text{MSE}}$ as Ours(MSE).

Rate loss. Benefited from importance map subnet, we define the rate loss directly on approximate code length. Suppose the size of encoder feature map $E(\mathbf{x})$ is $n \times h \times w$. The code by our model includes two parts: (i) quantized importance map $QI(\mathbf{p})$ with the size $h \times w$; (ii) the trimmed code with the size $\frac{n}{L}\sum_{i,j} QI(p_{ij})$. Note that the size of $QI(\mathbf{p})$ is constant given an image size. Thus $\frac{n}{L}\sum_{i,j} QI(p_{ij})$ can be used as an indicator of code length.

For better rate control, we introduce a threshold r based on the expected code length for a given compression rate, and penalize the rate loss only when it is higher than r. Then, we define the rate loss in our model as,

$$\mathcal{L}_R(\mathbf{x}) = \max\left\{0, (\frac{n}{L}\sum\nolimits_{i,j} QI(p_{ij}) - r \cdot nhw)\right\} \ . \tag{3.7}$$

By this way, rate loss only penalizes the code length higher than $r \cdot nhw$, making the learnt compression system exhibit a comparable compression rate around the given one.

Considering that the number of trimmed codes is exactly equal to the number of 1s in the importance mask, the ratio loss can be equivalently rewritten as:

$$\mathcal{L}_R(\mathbf{x}) = \max\left\{0, (\sum\nolimits_{k,i,j} m_{kij} - r \cdot nhw)\right\} \ . \tag{3.8}$$

Quantization loss. For better quantization of encoder feature map, we employ Θ_k to parameterize the multi-valued quantization for the k-th channel and incorporate a quantization loss for minimizing the squared ℓ_2 error caused by quantization,

$$\mathcal{L}_{Quant} = \frac{1}{nhw}\sum\nolimits_{k,i,j} \|Q(e_{kij}) - e_{kij}\|^2. \tag{3.9}$$

Model objective. Let $\mathcal{X}$ be a set of training data, and $\mathbf{x} \in \mathcal{X}$ be an image from the set. The overall learning objective is then defined as the combination of distortion, rate, and quantization losses,

$$\mathcal{L} = \sum_{\mathbf{x}\in\mathcal{X}} \{\mathcal{L}_D(\mathbf{z}, \mathbf{x}) + \gamma\mathcal{L}_R(\mathbf{x}) + \eta\mathcal{L}_{Quant}\}, \tag{3.10}$$

where γ and η are tradeoff parameters for balancing the three loss terms. Considering that quantization loss is not directly related with the rate-distortion performance, we empirically set $\eta = 1$ and $\mathcal{L}_{Quant}$ is deployed to only update quantization parameters in training.

3.2.3 Relaxation of Quantization for Model Learning

As noted above, due to the quantization operations, the conventional back-propagation algorithm is not applicable to learn the encoder and importance map subnets. To circumvent this issue, two relaxation approaches are presented. To relax the quantization of the feature map, a proxy function based on a straight-through estimator is introduced to approximate the channel-wise quantization in backward propagation. To relax the quantization of importance map, a two-stage relaxation scheme is adopted to train importance map subnet.

Relaxation and learning of channel-wise multi-valued quantization

The gradients of the learned channel-wise multi-valued quantization function are zeros almost everywhere and are infinite at several threshold points. Being non-differentiable inevitably restricts the backward propagation of gradients from the decoder to encoder and gives rise to the difficulty in learning the deep image compression system. As a result, any layers before the quantization function (i.e., the whole encoder) are never updated during training.

Fortunately, some recent works on binarized neural networks (BNN) [17, 55, 86] have studied the issue of propagating gradient through binarization, which can also be extended to relax multi-valued quantization. Based on the straight-through estimator on the gradient [17], we introduce a linear proxy $\tilde{Q}(e_{kij})$ to approximate $Q(e_{kij})$,

$$\tilde{Q}(e_{kij}) = e_{kij}. \tag{3.11}$$

In particular, $Q(e_{kij})$ is still adopted in forward propagation, while $\tilde{Q}(e_{kij})$ is only used in backward-propagation. The gradient of $\tilde{Q}(e_{kij})$ can then be easily obtained by,

$$\tilde{Q}'(e_{kij}) = 1. \tag{3.12}$$

Albeit that the proxy function can ease the difficulty of model learning, its effectiveness actually depends on the values of e_{kij}s in training. When the quantization error is 0, $Q(e_{kij})$ equals to $\tilde{Q}(e_{kij})$, and it is safe to use $\tilde{Q}(e_{kij})$ as proxy function. Moreover, even the equality does not hold, the quantization loss in Eqn. (3.9) can constrain that $\tilde{Q}(e_{kij})$ approximates $Q(e_{kij})$. Thus, it is reasonable to use $\tilde{Q}(e_{kij})$ as proxy of the learnt quantization function in practice.

Initialization and re-initialization in learning. By assuming that encoder feature follows a uniform distribution in the range $[0, 1]$, we simply initialize $s_{k,t}$s as $s_{k,0} = \frac{1}{2T}$ and $s_{k,t} = \frac{1}{T}$ for $t > 0$. However, we empirically find that such initialization scheme may suffer from the dead point problem. For example, when all $Q(e_{kij})$s from the k-th channel are lower than q_{k,t_0}, the gradients with respect to $s_{k,t_0}, \ldots, s_{k,T-1}$ will be always zero, and the last few quantization levels, i.e., $q_{k,t_0}, \ldots, q_{k,T-1}$, will never be optimized and used during training. For handling this issue, we store the histogram of $q_{k,t}$s of each mini-batch. When all the counts of $q_{k,t}$s with $t \geq t_0$ are zero in 50 successive mini-batches, we re-initialize the weights $s_{k,t} = \frac{s_{k,t_0-1}}{T-t_0+1}$ for $t = t_0 - 1, \ldots, T - 1$. As a result, $Q(e_{kij})$s with $e_{kij} > q_{k,t_0-2}$ are more likely to be quantized to the last few quantization levels in future training, and the non-increasing quantization loss can also be guaranteed.

Relaxation and learning of importance map

Analogous to channel-wise quantization, a straight-through estimator can also be used to relax the quantization of importance map which includes the generation of both quantized importance map and binarized importance mask. However, we empirically find that such a solution works well only when binarization is adopted to quantize the encoder feature map. For better learning importance map subnet in a general setting, we introduce a two-stage relaxation scheme, Here, an alternative loss is used in the first stage to update $QI(p_{ij})$, and another alternative loss is adopted in the second stage to update the importance map subnet.

In the first stage, given the current code $\mathbf{z}^*$, by approximating $Q(e_{kij}) - Q(e^*_{kij})$ with

$e_{kij} - e^*_{kij}$, the distortion loss $\mathcal{L}_D$ is relaxed with its Taylor expansion w.r.t. $\mathbf{z}^*$,

$$\begin{aligned} \mathcal{L}'_D &= \mathcal{L}_D(\mathbf{z}^*) + \sum_{k,i,j} m_{kij} \frac{\partial \mathcal{L}_D}{\partial z_{kij}}|_{\mathbf{z}=\mathbf{z}^*}(e_{kij} - e^*_{kij}) \\ s.t. \quad & |e_{kij} - e^*_{kij}| \leq \xi \end{aligned} \tag{3.13}$$

where ξ is a small positive value, and it is empirically set as $\xi = 0.1$. By replacing the $\mathcal{L}_D$ with the proxy $\mathcal{L}'_D$, we define the proxy function for $QI(\mathbf{p})$ as,

$$\mathcal{L}'(QI(\mathbf{p})) = \mathcal{L}'_D + \gamma \mathcal{L}_R. \tag{3.14}$$

It is worth noting that, the minimization of $\mathcal{L}'(QI(\mathbf{p}))$ w.r.t. $QI(\mathbf{p})$ can be decomposed into $h \times w$ subproblems on $QI(p_{ij})$. The loss function for each $QI(p_{ij})$ can be represented as,

$$\mathcal{L}'_{ij} = \begin{cases} \sum_{k=0}^{n-1} m_{kij} t_{kij}, & \text{if } \sum_{k,i,j} m_{kij} < r \cdot nhw \\ \sum_{k=0}^{n-1} m_{kij}(t_{kij} + \gamma) - r \cdot n, & \text{otherwise} \end{cases} \tag{3.15}$$

where $t_{kij} = (e_{kij} - e^*_{kij}) \frac{\partial \mathcal{L}_D}{\partial z_{kij}}|_{\mathbf{z}=\mathbf{z}^*}$ with $|e_{kij} - e^*_{kij}| \leq \xi$. Note that $\mathcal{L}'_{ij}$ is a function of both e_{kij} and $QI(p_{ij})$. First, we only consider $\mathcal{L}'_{ij}$ w.r.t. e_{kij}. Due to that m_{kij} is non-negative, it is obvious that the minimum of t_{kij} should be $-\xi \left| \frac{\partial \mathcal{L}_D}{\partial z_{kij}}|_{\mathbf{z}=\mathbf{z}^*} \right|$. Then, given $t_{kij} = -\xi \left| \frac{\partial \mathcal{L}_D}{\partial z_{kij}}|_{\mathbf{z}=\mathbf{z}^*} \right|$, the minimization of $\mathcal{L}'_{ij}$ w.r.t. $QI(p_{ij})$ can be rewritten as:

$$\mathcal{L}'_{ij} = \begin{cases} -\xi \sum_{k=0}^{n-1} m_{kij} \left| \frac{\partial \mathcal{L}_D}{\partial z_{kij}}|_{\mathbf{z}=\mathbf{z}^*} \right|, & \text{if } \sum_{k,i,j} m_{kij} < r \cdot nhw \\ -\xi \sum_{k=0}^{n-1} m_{kij}(\left| \frac{\partial \mathcal{L}_D}{\partial z_{kij}}|_{\mathbf{z}=\mathbf{z}^*} \right| - \frac{\gamma}{\xi}) - r \cdot n, & \text{otherwise} \end{cases} \tag{3.16}$$

It is noted that $QI(p_{ij})$ only has L possible values, i.e., $QI(p_{ij}) \in \{0, \ldots, L-1\}$. Then, we define the L different importance masks as $\{s_l = (\underbrace{1, \ldots, 1}_{nl/L}, \underbrace{0, \ldots, 0}_{n-nl/L}) \mid l = 0, \ldots, L-1\}$. Thus, we simply test all possible $QI(p_{ij})$ values to find the optimal one for minimizing $\mathcal{L}'_{ij}$,

$$l^* = \arg\min_l \mathcal{L}'_{ij}. \tag{3.17}$$

In the second stage, we introduce a continuous proxy based on l^* for updating importance map subnet,

$$\mathcal{L}_{imp} = \alpha |l^* - p_{i,j} \cdot L|, \tag{3.18}$$

where α is a trade-off parameter and we set it to be 0.001 in our implementation. Thus, the gradient w.r.t. $p_{i,j}$ can be obtained by,

$$\frac{\partial \mathcal{L}_{imp}}{\partial p_{i,j}} = \begin{cases} -\alpha, & \text{if } p_{i,j} < l^*/L \\ \alpha, & \text{if } p_{i,j} > l^*/L \\ 0, & \text{otherwise.} \end{cases} \tag{3.19}$$

3.2.4 Implementation and Learning

In our implementation, we set the channel number of code maps $n = 32$. For the learnt channel-wise quantization function, we set the number of quantized values $T = 8$. As for the importance map, the number quantization levels L is set to 16. Without importance map and entropy coding, the whole code maps corresponds to a compression rate of 1.5 bpp. By setting specific threshold value r, our CWIC framework can be adapted to different compression rates without changing the network structure. For the setting of r, we simply let $r = \frac{2}{3}r_0$, where $r_0 \in \{0.1, 0.2, 0.3, 0.45, 0.6, 0.8, 1.00\}$ is the expected compression rate. The parameter γ controls the tradeoff between distortion and rate losses. According to $r_0 \in \{0.1, 0.2, 0.3, 0.45, 0.6, 0.8, 1.00\}$, we set $\gamma \in \{1 \times 10^{-3}, 5 \times 10^{-4}, 2 \times 10^{-4}, 1 \times 10^{-4}, 5 \times 10^{-5}, 2 \times 10^{-5}, 1 \times 10^{-5}\}$ to make the gradient of $\mathcal{L}_R$ term comparable to that of $\mathcal{L}_D$ term during training.

The whole CWIC model is trained using the ADAM solver [33]. We initialize the model with the parameters pre-trained on the training set $\mathcal{X}$ without the importance map subnet. The model is further trained with the learning rate of 1×10^{-4}, 1×10^{-5} and 1×10^{-6}. The smaller learning rate is adopted until the objective with the larger one keeps non-decreasing in five successive epochs.

3.3 Trimmed Convolutional Network for Arithmetic Encoding

The code $\mathbf{o}$ and quantized importance map $QI(\mathbf{p})$ by the above CWIC model are still spatially dependent and can be further losslessly compressed. For the code $\mathbf{o}$, there are two kinds of 0s, i.e., the 0 in the quantization index and the 0 generated by the element-wise product with importance mask. Nonetheless, the former 0s are informative, while the later should be ignored in entropy prediction but still can be used as the context of the other symbols. To distinguish these two kinds of 0s, we simply adopt $\mathbf{o}' = (\mathbf{o} + 1) \circ \mathbf{m}$ in our implementation.

For lossless compression, arithmetic coding [79] predicts the probability of the current symbol to be encoded from its context and is proved to be the optimal coding in approximating entropy-based compression rate [60]. Thus, we adopt the arithmetic coding

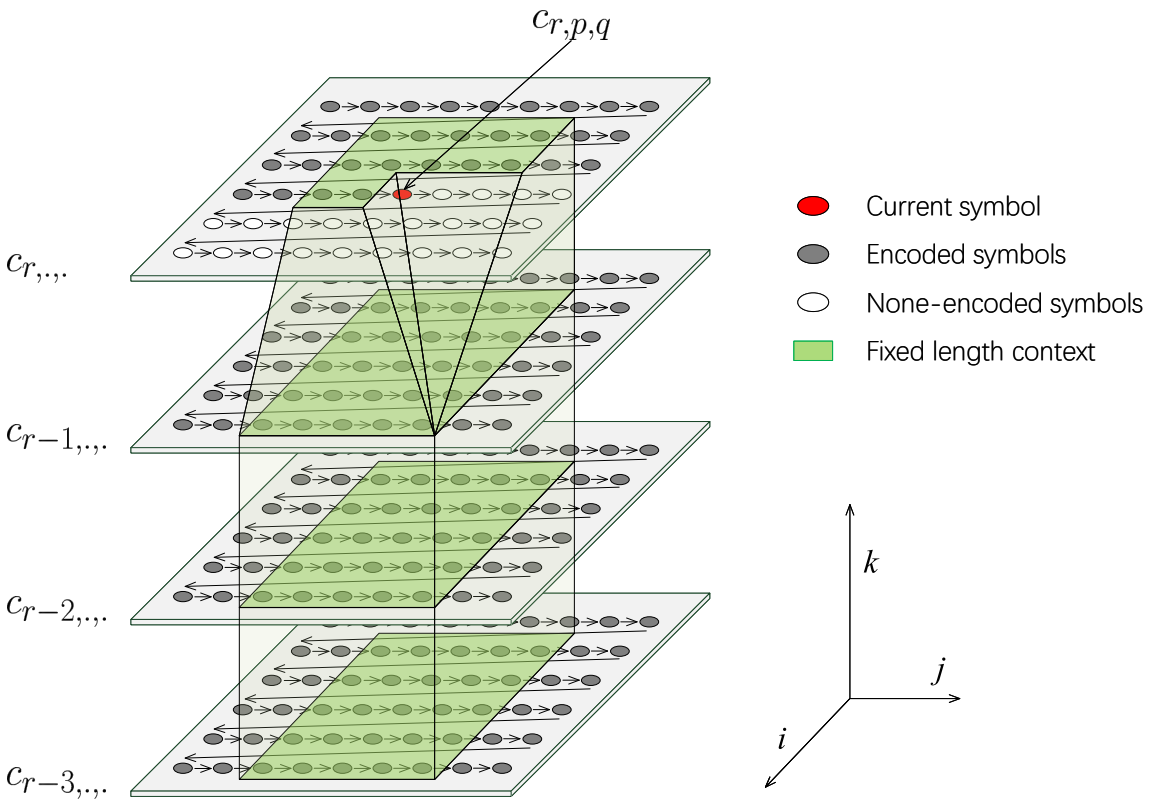


Figure 3.4: Coding schedule and context of 3D cuboid. The arrows indicate the encoding order of the cuboid. The green areas are the fixed length context of the symbol $c_{r,p,q}$ with $h_t = 2$ and $w_t = 2$. The red circle represents the current symbol $c_{r,p,q}$, and the gray (white) circles represent the encoded (non-encoded) symbols.

framework, and present a trimmed convolutional network model for efficient modeling of large context.

3.3.1 Coding Schedule and Context of 3D Cuboid

To begin with, we note that both $\mathbf{o}'$ and $QI(\mathbf{p})$ can be represented as a 3D cuboid $\mathbf{C} = \{c_{k,i,j} | 0 \leq k \leq n-1, 0 \leq i \leq h-1, \text{and } 0 \leq j \leq w-1\}$. Before TCAE, we first introduce the coding schedule and two types of context based on $\mathbf{C}$. As illustrated in Fig. 3.4, beginning at $c_{0,0,0}$, we adopt the following order to encode $\mathbf{C}$: (i) $c_{k,i,j+1}$ is encoded after $c_{k,i,j}$ until $j = w-1$; (ii) when $j = w-1$, $c_{k,i+1,0}$ is encoded after $c_{k,i,w-1}$ until $i = h-1$; (iii) when $j = w-1$ and $i = h-1$, $c_{k+1,0,0}$ is encoded after $c_{k,h-1,w-1}$.

For a position (r, p, q), we define its full context as, $\text{CTX}(c_{r,p,q}) = \{c_{k,i,j} | \{k <$

$r\}\vee\{k=r,i<p\}\vee\{k=r,i=p,j<q\}\}$, i.e., all the gray circle in Fig. 3.4. Unfortunately, the length of the full context $\mathrm{CTX}(c_{r,p,q})$ is unfixed and varies by the position (r,p,q), making it difficult to learn a CNN-based probability prediction model based on $\mathrm{CTX}(c_{r,p,q})$.

Naturally, the context spatially close to the current symbol $c_{r,p,q}$ plays a more important role in probability prediction. Taking these aspects into account, we give a fixed length context defined as $\mathrm{CTX}_f(c_{r,p,q})=\{c_{k,i,j}|\{r-c_t\le k<r,|i-p|\le h_t,|j-q|\le w_t\}\vee\{k=r,p-h_t\le i<p,|j-q|\le w_t\}\vee\{k=r,i=p,q-w_t\le j<q\}\}$. Furthermore, considering that different channels are not totally independent, we empirically suggest to set a larger c_t or even include all channels in context modeling. Fig. 3.4 gives an example of $\mathrm{CTX}_f(c_{r,p,q})$ with $h_t=2$ and $w_t=2$ for intuitive illustration.

In our pioneer work [38], we extract a $(c_t+1)\times(2h_t+1)\times(2w_t+1)$ cuboid $\mathrm{CTX}'_f(c_{r,p,q})=\{c_{k,i,j}|\{r-c_t\le k\le r,|i-p|\le w_t,|j-q|\le h_t\}$ with $c_t=3,h_t=2,w_t=2$. For context modeling, the non-encoded symbols in $\mathrm{CTX}'_f(c_{r,p,q})$ are replaced with 0s. Then, a convolutional entropy prediction model of three convolution layers followed by three fully connected layers is introduced to predict $c_{r,p,q}$ from its context cuboid $\mathrm{CTX}'_f(c_{r,p,q})$.

In [38], the contexts and non-encoded symbols are dynamically changed along with the encoding process. Thus, convolutional entropy prediction model requires to perform probability prediction independently without shared computation, thereby remaining computationally expensive. Consequently, even though the introduction of non-encoded symbols is necessary for context modeling, it also brings new difficulties to exploit fully convolutional network (FCN) for shared computation. Next, we will present a group of trimmed convolutions to circumvent the inefficiency issue.

3.3.2 Trimmed Convolution

The fixed-length context in convolutional entropy encoder has two appealing properties, which can be exploited to perform probability prediction via trimmed convolutions. (i) Given the current symbol $c_{r,p,q}$, the positions of all non-encoded symbols are fixed. (ii) The default value for all non-encoded symbols is also a fixed number, and without loss of generality, we

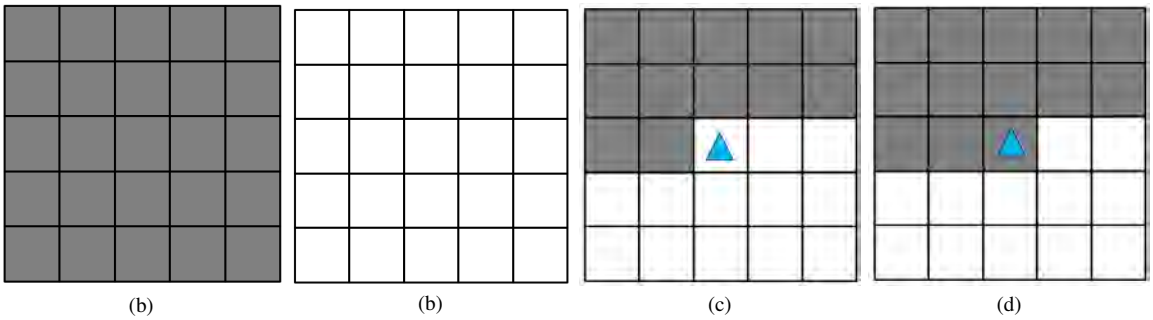


Figure 3.5: Mask planes with respect to w_t for trimmed convolution kernels with the size of 5×5. The gray value denotes 1 and the white value denotes 0. The blue triangle represents the position of the codes to be encoded with respect to the mask. (a) $k < t$, (b) $k > t$, (c) $k = t$ for the input layer, (d) $k = t$ for the hidden layers.

can set it to be 0.

We first begin with the definition and analysis of standard convolution operator. Denote by $\mathbf{w}^0$ a group of n convolution kernels $\mathbf{w}^0 = \{w_{t,k,i,j} | -w_0 \leq j \leq w_0, -h_0 \leq i \leq h_0, 0 \leq k < n, 0 \leq t < n\}$. Then the convolution result $(\mathbf{C} * \mathbf{w}^0)$ at the location (r, p, q) can be written as,

$$(\mathbf{C} * \mathbf{w}^0)(r, p, q) = \sum_{k,l-i=p,m-j=q} c_{k,l,m} w^0_{r,k,i,j}. \tag{3.20}$$

where $*$ denotes the convolution operator. However, such convolution treats the context and non-encoded symbols equally and cannot be applicable to context modeling.

We then present our trimmed convolution by taking the properties of non-encoded symbols into account. From Property (ii), we can keep voxel values unchanged, and introduce zeros in convolutional kernel $\mathbf{w}^0$ to exclude the effect of non-encoded symbols in convolution. From Property (i), the positions of non-encoded symbols are fixed and predefined w.r.t. $\mathbf{w}^0$, thereby allowing us to employ a mask $\hat{\mathbf{m}}$ of $\{0, 1\}$ for correctly setting zeros. Here, $\hat{m}_{k,i,j}$ is defined as 1 if $c_{k,p+i,q+j}$ is encoded before $c_{k,p,q}$, and 0 otherwise. Trimmed convolution is then defined as,

$$\mathbf{C}^1 = \mathbf{C} * (\hat{\mathbf{m}} \circ \mathbf{w}^0), \tag{3.21}$$

where $\circ$ denotes the element-wise product. With trimmed convolution, we can safely exclude the effect of non-encoded symbols in context modeling while maintaining the efficiency of FCN for predicting probabilities of all voxels.

In the following, we first use single convolution kernel as an example to explain the settings of $\hat{\mathbf{m}}$, which are different for the input layer and the hidden layers. For the input layer, when predicting the probability of $c_{r,p,q}$, both $c_{r,p,q}$ and the symbols encoded after $c_{r,p,q}$ should be masked out in trimmed convolution. Following the definition of context, we define the mask $\hat{\mathbf{m}}^0$ for the input layer as,

$$\hat{m}^0_{tkij} = \begin{cases} 1, \text{if } \{k<t\} \vee \{k=t, i<0\} \vee \{k=t, i=0, j<0\} \ , \\ 0, \text{otherwise} \ . \end{cases} \tag{3.22}$$

When it comes to the hidden layer $\mathbf{C}^d$ ($d \geq 1$), we note that the feature $c^d_{r,p,q}$ only conveys the context information of $\mathbf{c}_{r,p,q}$ and should not be excluded in the successive context modeling. Therefore, we modify the definition of the mask $\hat{\mathbf{m}}^d$ ($d \geq 1$) for hidden layer as,

$$\hat{m}^d_{tkij} = \begin{cases} 1, \text{if } \{k<t\} \vee \{k=t, i<0\} \vee \{k=t, i=0, j\leq 0\} \ , \\ 0, \text{otherwise.} \end{cases} \tag{3.23}$$

Using the convolution kernel $\mathbf{w}$ with the size of $5\times5\times n$ as an example, Fig. 3.5 illustrates the representative mask planes w.r.t. the kernel planes $\mathbf{w}_{t,k,\cdot,\cdot}$. As shown in Fig. 3.5(a) (Fig. 3.5(b)), when $k < t$ ($k > t$), the k-th mask plane is a matrix of 1s (0s) for both the input and hidden layers. When $k = t$, the center position should be masked out in the mask plane for the input layer (see Fig. 3.5(c)), but can be retained for the hidden layers (see Fig. 3.5(d)).

Multi-group trimmed convolution. The trimmed convolution in Eqn. (3.21) only uses one group of convolution kernels in each layer, which is still limited in complicated probability prediction. Thus, we extend the trimmed convolution to the multi-group form. Suppose there are g_{in} groups of feature maps $\mathcal{C}^d = \{\mathbf{C}^{d,0}, ..., \mathbf{C}^{d,g_{in}-1}\}$ in the d-th layer and g_{out} groups of feature maps $\mathcal{C}^{d+1} = \{\mathbf{C}^{d+1,0}, ..., \mathbf{C}^{d+1,g_{out}-1}\}$ in the $(d+1)$-th layer. Each group of feature map has the same size with the input cuboid $\mathcal{C}$. The group trimmed convolution is defined as,

$$\mathbf{C}^{d+1,g'} = \sum_{g=0}^{g_{in}-1} \mathbf{C}^{d,g} * (\hat{\mathbf{m}}^d \circ \mathbf{w}^{d,g,g'}), \tag{3.24}$$

where $\mathbf{C}^{d,g}$ denotes the g-th group of feature map in $\mathcal{C}^d$, $\mathbf{C}^{d+1,g'}$ denotes the g'-th group of feature map in $\mathcal{C}^{d+1}$. $\hat{\mathbf{m}}^d$ is the mask for the d-th layer, and $\mathbf{w}^{d,g,g'}$ is the convolution kernel to connect $\mathbf{C}^{d,g}$ and $\mathbf{C}^{d+1,g'}$.

3.3.3 TCAE and Learning Objective

Our proposed TCAE is constructed by stacking several 5×5 trimmed convolution layers to enlarge the context and increase the nonlinearity of the model. Given all the model parameters $\mathcal{W} = \{\mathbf{w}^{d,g,g'}\}$, the output of TCAE can be written as $F(\mathbf{C}; \mathcal{W}) = \{(F(\mathbf{C}; \mathcal{W}))^b_{r,p,q} | b = 0, \ldots, m-1\}$. Here, $(F(\mathbf{C}; \mathcal{W}))^b_{r,p,q}$ denotes the predicted probability of $c_{r,p,q} = b$, and m is the number of quantization levels of the input code map. Using $\mathbf{o}'$ as an example, we adopt the code length after arithmetic encoding as the learning objective,

$$\ell(\mathcal{W}; \mathbf{C}) = \sum_{r,p,q} \sum_{b=0}^{m-1} -m_{rpq} s(c_{r,p,q}, b) \log_2 \left(F(\mathbf{C}; \mathcal{W})\right)^b_{r,p,q} \tag{3.25}$$

where $s(c_{r,p,q}, b) = 1$ when $c_{r,p,q} = b$, and $s(c_{r,p,q}, b) = 0$ otherwise. m_{rpq} is an importance mask to exclude those codes with the $m_{rpq} = 0$ according to Eqn. (3.4) during training.

3.3.4 Inclined TCAE

The above TCAE can only accelerate the encoding process. In the decoding stage, the codes should still be decoded in a sequential order and cannot be speeded up with GPU and parallel computation. To alleviate this issue, we present an inclined TCAE model by introducing another kind of coding schedule and context. Concretely, we divide the 3D cuboid $\mathbf{C}$ into $n + h + w - 2$ inclined planes, where the t-th inclined plane is defined as $CB_t(\mathbf{C}) = \{c_{k,i,j} | k+i+j = t\}$ $(t = 0, 1, \ldots, n+h+w-3)$. In inclined TCAE, the context of the current symbol $c_{r,p,q}$ is then defined as $\text{CTX}_b(c_{r,p,q}) = \{CB_0(\mathbf{C}), CB_1(\mathbf{C}), \ldots, CB_{r+p+q-1}(\mathbf{C})\}$. In terms of coding schedule, we simply begin with $CB_0(\mathbf{C})$ and then gradually encode $CB_t(\mathbf{C})$ after $CB_{t-1}(\mathbf{C})$.

To accelerate the encoding process, we define the mask for the input and hidden layers as follows,

$$\hat{m}^0_{tkij} = \begin{cases} 1, \text{if } i + k + j < 0 \ , \\ 0, \text{otherwise} \ . \end{cases} \tag{3.26}$$

$$\hat{m}^d_{tkij} = \begin{cases} 1, \text{if } i + k + j \leq 0, \\ 0, \text{otherwise} \ . \end{cases} \tag{3.27}$$

Benefited from the new coding schedule and context, all $c_{r,p,q}$s in $CB_t(\mathbf{C})$ share the same

context and can be decoded in parallel, which can then be utilized to speed up decoding process with GPU parallel computation.

3.3.5 Implementation and Learning

Two inclined TCAE models are deployed for context modeling of the code $\mathbf{o}'$ and quantized importance map $QI(\mathbf{p})$, respectively.

As for network structure, our inclined TCAE is comprised of 2 trimmed convolution layers, 3 residual blocks with each consisting of 2 trimmed convolution layers, and a final trimmed convolution layer followed by a softmax function. For context modeling of $\mathbf{o}'$, 8 groups are used for the first 8 trimmed convolution layers and 9 groups are used for the last layer. For context modeling of $QI(\mathbf{p})$, 32 groups are used for the first 8 layers and 16 groups for the last layer.

To train inclined TCAE, we adopt the ADAM solver [33]. The model is trained with the learning rate of 3×10^{-4}, 1×10^{-4}, 3.33×10^{-5} and 1.11×10^{-5}. The smaller learning rate is adopted until the objective with the larger learning rate keeps non-decreasing in five successive epochs.

3.4 Experiments

In this section, we compare our full CWIC method with both existing image encoding standards and state-of-the-art deep image compression models. Several ablation studies are also given to assessing the effect of importance map, channel-wise multi-valued quantization, and inclined TCAE. The pre-trained models will be available at https://github.com/limuhit/CWIC.

3.4.1 Experimental Setup

By setting different r values, we train 7 models based on MS-SSIM distortion loss and 7 models based on MSE loss. All the models are trained with 10,000 high-quality images crawled from the photo-sharing website *Flickr*. Each image is downsampled to its 1/3 size

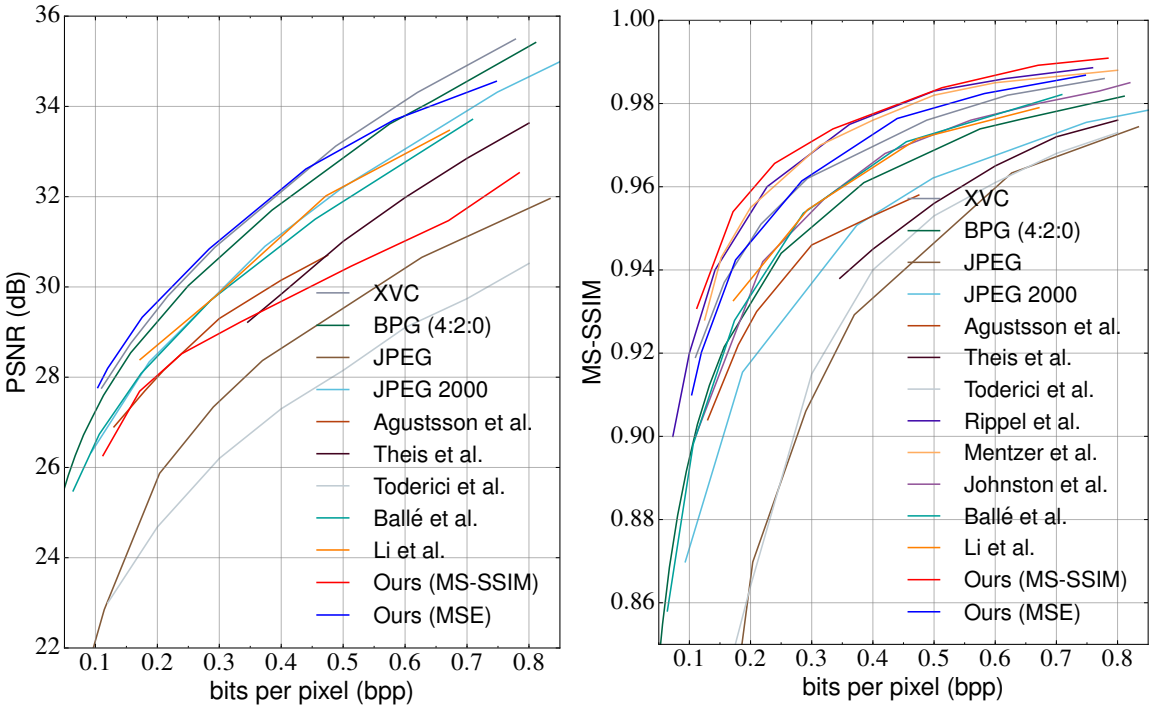
for removing the possible compression artifacts caused by JPEG compression and save the downsampled image with the lossless *PNG* format. Finally, 500,000 patches with a size of 384×384 are randomly cropped from the 10,000 images for training. For performance evaluation, we adopt two datasets, i.e., Kodak PhotoCD and Tecnick. The compression rate of our model is evaluated by bits per pixel (bpp), which is calculated as the total amount of bits used to code the image divided by the number of pixels. The image distortion is evaluated by the Multi-Scale Structural Similarity (MS-SSIM) and the Peak Signal-to-Noise Ratio (PSNR).

3.4.2 Quantitative Evaluation

Using MS-SSIM and PSNR as performance metrics, we evaluate the rate-distortion performance of our CWIC, existing image encoding standards, and state-of-the-art deep image compression models. In terms of image encoding standards, we consider JPEG [73], JPEG 2000 [64], and BPG [13]. Among different variants of JPEG, the optimized JPEG with 4:2:0 chroma subsampling is adopted[1]. JPEG 2000 is based on the optimized implementation in MATLAB 2015, and the implemented BPG is based on the 4:2:0 chroma format[2]. In terms of deep image compression models, their source codes generally are not available. Following the strategy adopted in [46], we carefully digitalize and collect rate-distortion curves from the related literatures [2, 58, 67] in our comparative experiments.

Fig. 3.6 shows the rate-distortion curves of competing methods on the Kodak datasets. In terms of MS-SSIM, we compare our method with JPEG, JPEG 2000, BPG, Ballé et al. [11], Rippel et al. [58], Theis et al. [67], Johnston et al. [32], Toderici et al. [68], Agustsson et al. [2] and Mentzer et al. [46]. In terms of PSNR, we exclude Rippel et al. and Mentzer et al. due to that the PSNR result is not reported in their paper [46, 58]. From Fig. 3.6(a), Ours(MS-SSIM) performs on par with Rippel et al. [58] and outperforms the other methods by MS-SSIM. It is worth noting that, Ours(MSE) also exhibits competitive MS-SSIM performance, which is only inferior to Rippel et al. [58] and Mentzer et al. [46] among

[1] http://libjpeg.sourceforge.net/
[2] https://bellard.org/bpg/



(a) PSNR

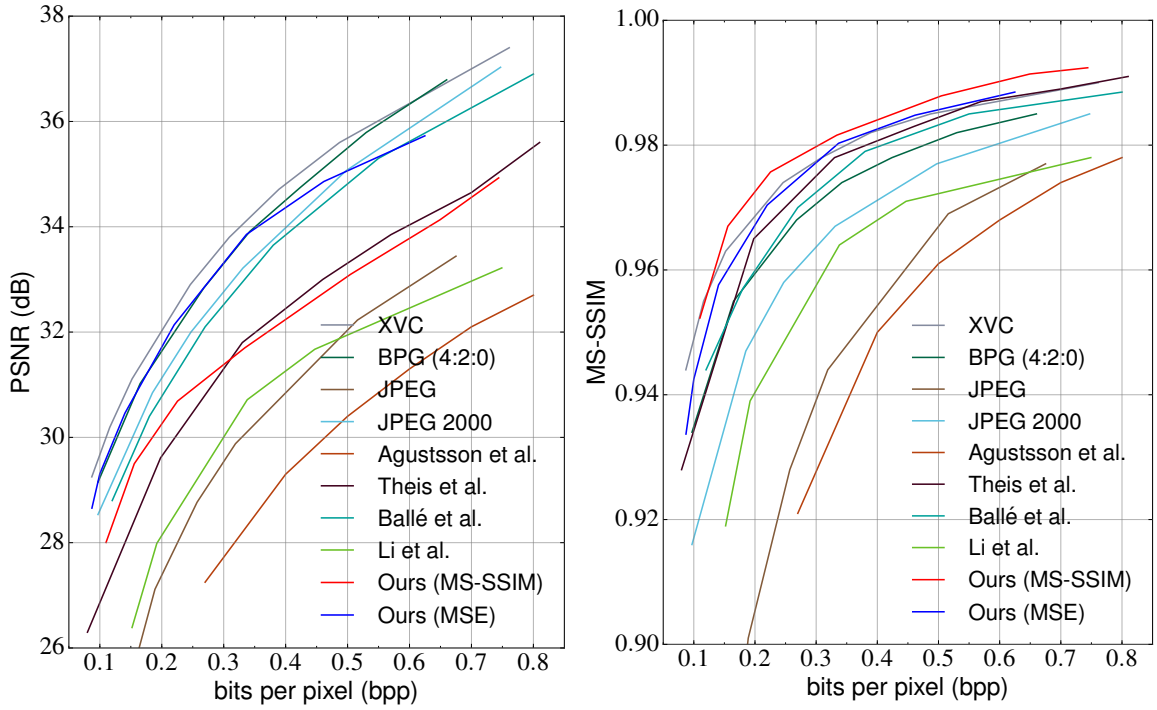
(b) MS-SSIM

Figure 3.6: Rate-distortion curves of different compression algorithms w.r.t. (a) PSNR and (b) MS-SSIM on the Kodak PhotoCD image dataset.

the competing methods, probably being explained by that the use of importance map benefits the reconstruction of salient and structural information at lower bpp. From Fig. 3.6(b), Ours(MSE) is comparable with BPG, and is much better than the other methods. Furthermore, we give the rate-distortion curves of competing methods on Tecnick [9, 10] in Fig. 3.7, and get the similar observations with the Kodak dataset. It is noted that the results of Rippel et al. [58], Theis et al. [67], Agustsson et al. [2] and Mentzer et al. [46] are unavailable on Tecnick.

3.4.3 Visual Quality Evaluation

Quantitative evaluation is conducted to assess the visual quality of decoding images by different methods. Among deep models, most existing methods except Ballé et al. [11] do not provide either source codes or decoding images. Among image coding standards, JPEG 2000 and BPG are superior to JPEG by quantitative metrics. Thus, we compare Ours(MS-SSIM) with JPEG 2000, BPG, Ballé et al. [11] in our experiment.



(a) PSNR (b) MS-SSIM

Figure 3.7: Rate-distortion curves of different compression algorithms w.r.t. (a) PSNR and (b) MS-SSIM on the Tecnick dataset.

Fig. 3.8 shows the decoding images of competing methods on five Kodak images. Visual artifacts, e.g., blurring and ringing, can still be observed from the results of JPEG 2000 and BPG. Ballé et al. [11] is effective in suppressing ringing artifacts, but is limited in handling small-scale details, and also suffers from blurring and smoothing effect at lower bpp. In contrast, the results by our method exhibit much less noticeable artifacts and are visually much more pleasing. More importantly, due to the introduction of importance map based bit length allocation, our method is more effective in retaining the salient structure and fine details in comparison to the competing methods.

Fig. 3.9 shows the visual comparison between the proposed methods optimized by MS-SSIM and MSE, respectively. At lower bpp, Ours(MSE) performs well in preserving sharp strong edges and smooth textures, while Ours(MS-SSIM) is superior in keeping small-scale textures and weak edges. However, at higher bpp, Ours(MS-SSIM) fails to reconstruct parts of small-scale edges, e.g., the eyelash in the bottom-right of Fig. 3.9. One possible explanation is that MS-SSIM is designed for measuring multi-scale similarity. As a result, the

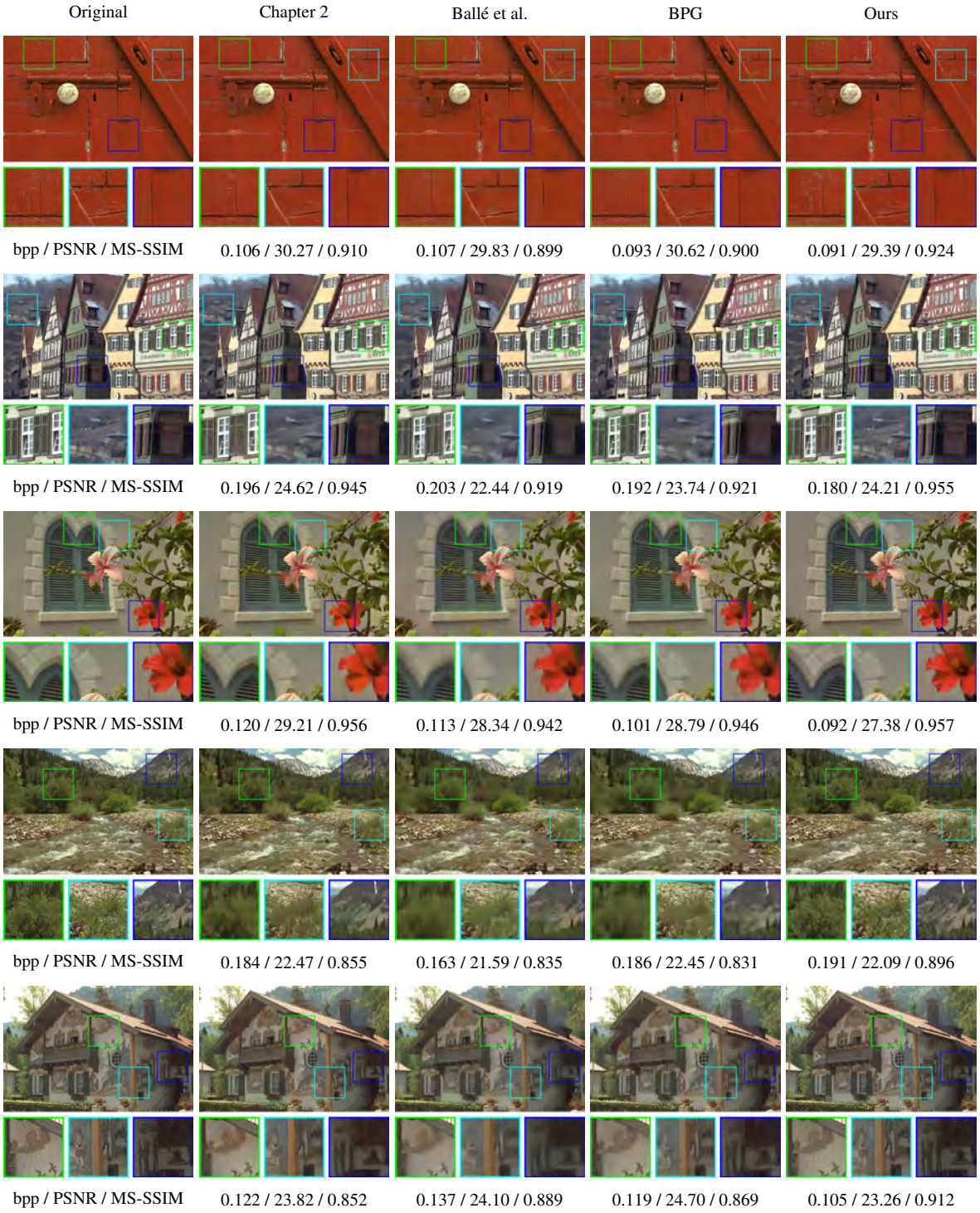


Figure 3.8: Decoding images produced by different compression systems. From the left to right: ground-truth, the results of Chapter 2, Ballé et al. [11], BPG and Ours(MS-SSIM). In general, our model achieves the best visual quality, demonstrating the superiority of our model in preserving both sharp edges and detailed textures. (Best viewed on screen in color)

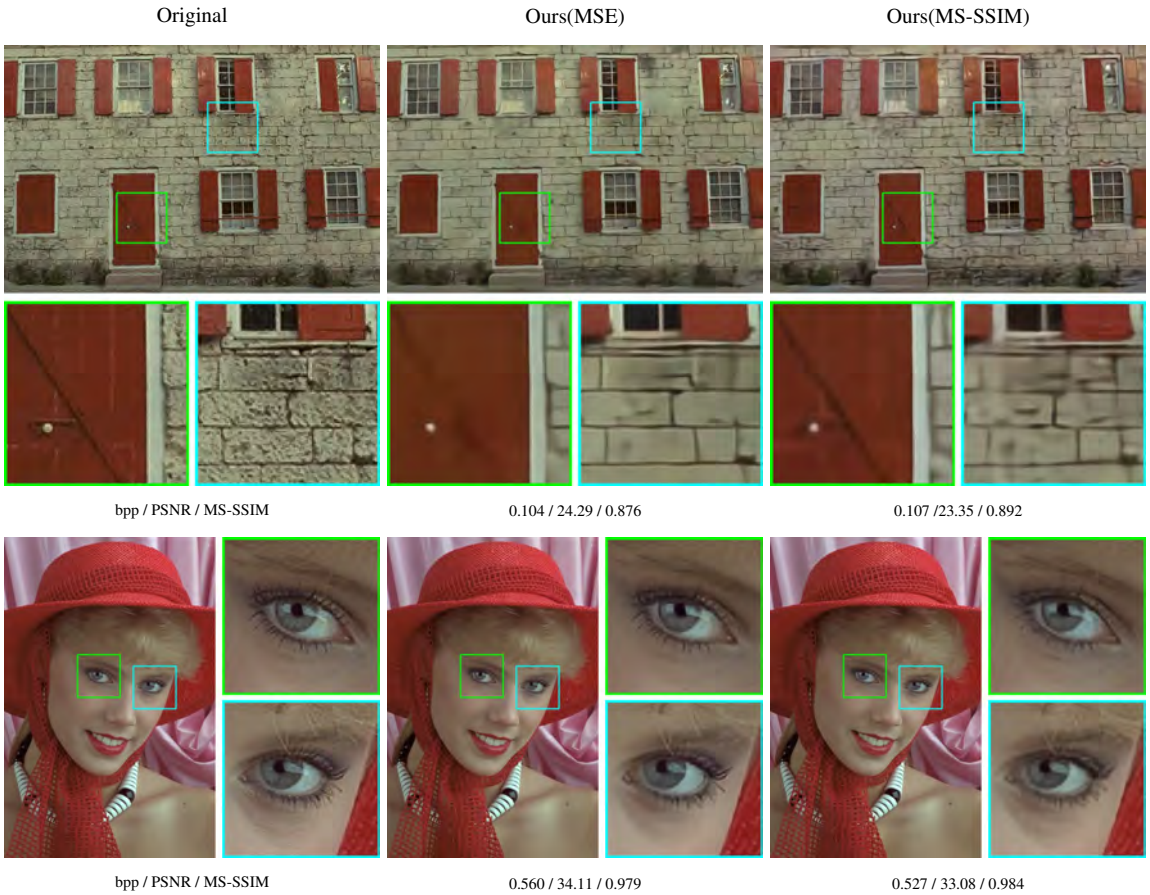


Figure 3.9: Decoding images produced by our models optimized with MSE and MS-SSIM, respectively. Ours(MS-SSIM) exhibits better textures at lower bpp but may slightly obscure small sharp edges.

small edges are usually ignored at a large scale, which inevitably diminishes the contribution of small-scale edges in the metric.

3.4.4 Ablation Studies

In this section, we separately test the effect of three components, i.e., the channel-wise multi-valued quantization, importance map, and TCAE, with ablation studies. For a fair comparison, we simply reuse all the parameters for training the 7 CWIC model in Sec. 3.4.1. $\mathcal{S} = \{\gamma, r, \mathbf{w}_e^0, \mathbf{w}_d^0\}$ denotes a set of parameters to train a CWIC. Here, $\mathbf{w}_e^0$ and $\mathbf{w}_d^0$ are separately the initial weights of encoder and decoder. In the experiments, all the ablation variant models are trained on the 7 parameter sets, i.e. $\mathcal{S}_1, \ldots, \mathcal{S}_7$, by MS-SSIM distortion loss and tested on Kodak dataset.

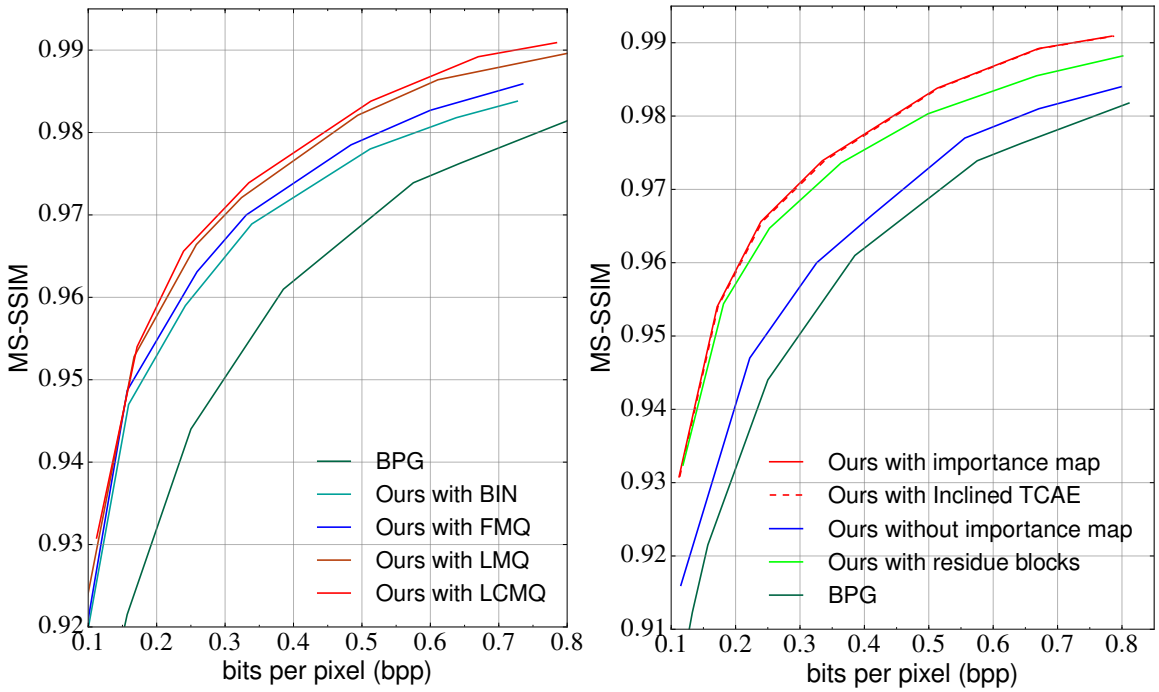
Channel-wise multi-valued quantization

We consider three other variants for the learnt channel-wise multi-valued quantization (LCMQ), i.e., (1) the learnt multi-value quantization (LMQ) with all the channels sharing the same quantization function, (2) the fixed multi-valued quantization (FMQ) with all the quantization levels are fixed and (3) the binarization function (BIN) used in [38]. The quantization levels for all the variants except for BIN are set to be 8. For FMQ, we adopt the uniform multi-valued quantization used to initialize LCMQ in Sec. 3.2.3. For BIN, 1 bit instead of 3 bits is used to represent code in $\mathbf{c}$ before entropy coding, and we increase the number of channels of $\mathbf{o}$ from 32 to 96 to compensate for the total number of bits to represent $\mathbf{o}$.

Table 3.1: Quantization error of four quantization functions, i.e., LCMQ, LMQ, FMQ and BIN, on 7 parameter sets.

Set	LCMQ	LMQ	FMQ	BIN
$\mathcal{S}_1$	0.97×10^{-3}	1.12×10^{-3}	2.21×10^{-3}	3.29×10^{-2}
$\mathcal{S}_2$	0.92×10^{-3}	1.01×10^{-3}	1.78×10^{-3}	3.13×10^{-2}
$\mathcal{S}_3$	0.79×10^{-3}	1.03×10^{-3}	1.93×10^{-3}	3.24×10^{-2}
$\mathcal{S}_4$	0.78×10^{-3}	0.93×10^{-3}	2.09×10^{-3}	3.15×10^{-2}
$\mathcal{S}_5$	0.92×10^{-3}	1.00×10^{-3}	1.89×10^{-3}	3.07×10^{-2}
$\mathcal{S}_6$	1.01×10^{-3}	1.20×10^{-3}	2.08×10^{-3}	3.21×10^{-2}
$\mathcal{S}_7$	0.93×10^{-3}	1.03×10^{-3}	1.81×10^{-3}	3.18×10^{-2}
AVE	0.90×10^{-3}	1.05×10^{-3}	1.97×10^{-3}	3.18×10^{-2}

By replacing LCMQ with each of the three other variants, we re-train the CWIC



(a) Quantization (b) Others

Figure 3.10: Rate-distortion curves for ablation studies on Kodak. (a) comparison of four quantization variants, i.e., LCMQ, LMQ, FMQ and BIN. (b) comparison of other variants of our method.

model on the 7 parameter sets with the MS-SSIM distortion loss. Two metrics, i.e., distortion of the decoding images and quantization error, are reported based on the Kodak dataset. The quantization error is defined as the MSE between the output of encoder $\mathbf{e}$ and the quantized code $Q(\mathbf{e})$. Table 3.1 lists the quantization error of the four quantization functions. Unsurprisingly, the BIN in [38] obtains the largest quantization error due to that it has only two quantization levels. Among the multi-valued quantization functions, LCMQ and LMQ get much lower quantization error than FQM, indicating that the learning of quantization is indeed helpful in decreasing quantization error when the quantization levels are the same. Nonetheless, LCMQ achieves the lowest quantization error on all the 7 parameter sets, demonstrating the usefulness of the learned quantization function for each channel.

Fig. 3.10(a) shows the rate-distortion curves of our CWIC with the four quantization variants on the Kodak dataset. It can be seen that LCMQ gets the best performance followed by LMQ, while BIN exhibits the worst performance. We note that the rate-distortion results are consistent with the quantization error, where lower quantization error corresponds to

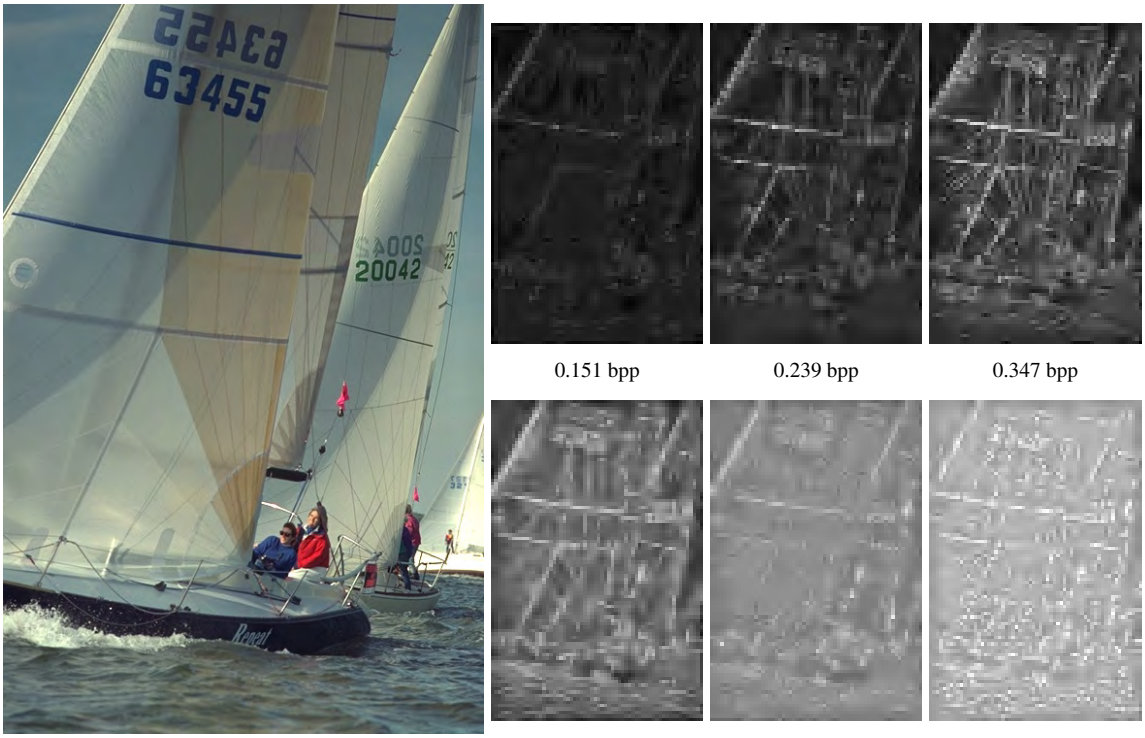
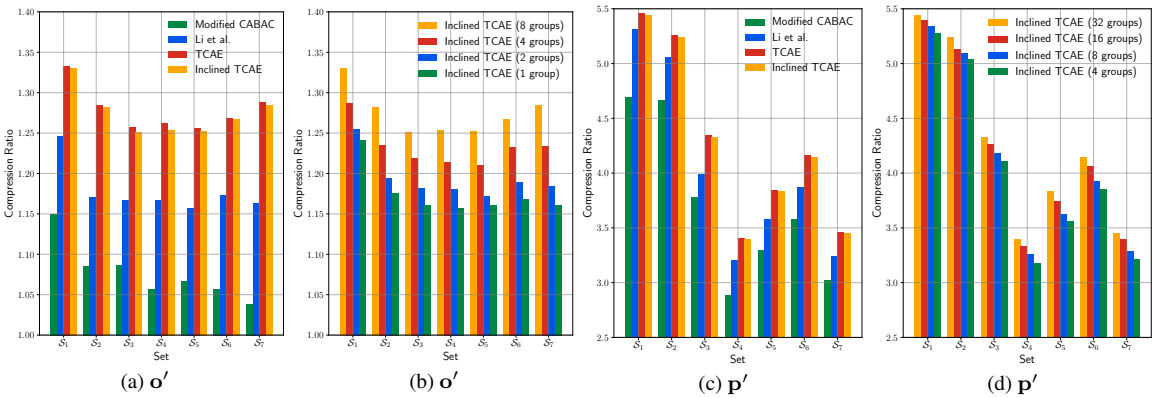


Figure 3.11: Visualization of the importance maps at 6 kinds of bpps. Left: ground-truth. Right: importance maps ranging from 0.151 to 0.814 bpp.



(a) $\mathbf{o}'$ (b) $\mathbf{o}'$ (c) $\mathbf{p}'$ (d) $\mathbf{p}'$

Figure 3.12: Lossless compression ratio of entropy prediction models. The data used to test the entropy prediction models are generated by our CWIC with 7 different parameter sets. (a) and (c) respectively show the results of the four entropy prediction models on the code $\mathbf{o}'$ and the quantized importance map $\mathbf{p}'$. (b) and (d) respectively show the results of inclined TCAE with different number of groups on $\mathbf{o}'$ and $\mathbf{p}'$.

lower distortion. Therefore, the quantization loss in Sec. 3.2.2, which is introduced to minimize quantization error, can be expectantly beneficial to rate-distortion performance and to ease the gradient issue of quantization in back-propagation.

Dense block

To assess the effect of dense blocks, we compare our method with its counterpart by replacing each dense block with three residual blocks. To keep the same network depth, the first residual block has three convolution layers and the following two residual blocks have two convolution layers. As shown in Fig. 3.10(b), our method exhibits better compression performance than the counterpart with residual blocks, especially at higher bit rates.

Importance map

To assess the effect of importance map, we introduce a baseline model by removing the importance map subnet from CWIC, and set the compression rate by adjusting the number of channels n in the code $\mathbf{c}$. In the experiments, we adopt $n = 4, 8, 12, 16, 20, 24$. As shown in Fig. 3.10 (b), the introduction of importance map can result in much better performance, clearly demonstrating the effectiveness of the spatially variant bit length allocation scheme. It is also interesting to note that, due to the learned channel-wise multi-valued quantization, our CWIC can also outperform BPG by MS-SSIM.

To reveal the role of importance map, we visualize the importance maps of a representative image at 6 kinds of bpps. The importance maps are mapped from the range of [0,1] to [0,255] and shown as gray images. From Fig. 3.11, it can be observed that at lower bpp the learned importance map only allocates more codes to the strong edges. Along with the increase of bit rate, more codes will be allocated to weak edges and mid-scale textures. With the further increase of bit rate, small-scale textures such as the wave start to be allocated with more codes. Thus, the learned importance map is consistent with human visual perception, which also explains the superiority of our model in preserving the structure, edges, and textures.

Inclined TCAE for encoding and decoding

For entropy modeling, we compare TCAE and inclined TCAE with two counterparts, i.e., our pioneer work (convolutional entropy prediction model) [38] and a modified CABAC [44] to compress the code $\mathbf{o}' = (1 + \mathbf{o}) \circ \mathbf{m}$ and the quantized importance map $\mathbf{p}' = QI(\mathbf{p})$. In [38], a $4 \times 5 \times 5$ cuboid is extracted for the symbol $o'_{r,p,q}$ with the importance mask $m_{r,p,q} = 1$, while a $1 \times 5 \times 5$ cuboid is extracted for symbols in $\mathbf{p}'$. For CABAC [44], only two nearby symbols, i.e., $o'_{k,i-1,j}$ and $o'_{k,i,j-1}$, are considered as the context of $o'_{k,i,j}$. In our modification, we further consider the relation across channels and also include the symbol $o'_{k-1,i,j}$ into the context. For the quantized importance map with only 1 channel, the context is kept the same as CABAC.

Fig. 3.12 (a) and (c) show the lossless compression ratios of the four context modeling methods on $\mathbf{o}'$ and the quantized importance map $\mathbf{p}'$. In particular, our TCAE and inclined TCAE achieve comparable compression ratio and outperform the two counterparts. Moreover, convolutional entropy prediction [38] is also superior to modified CABAC. Using the code $\mathbf{o}'$ as an example, the context sizes of TCAE, inclined TCAE, convolutional entropy prediction [38], and modified CABAC are $k \times 37 \times 37$, $k \times 37 \times 37$, $4 \times 5 \times 5$ and 3, respectively. Here, k is the channel index of $\mathbf{o}'$. By comparing the four context modeling methods, it can be seen that larger context generally is beneficial to better entropy prediction.

In Fig. 3.12 (b) and (d), we test inclined TCAE with different number of groups of feature maps, i.e., 1, 2, 4 and 8 groups for the code $\mathbf{o}'$ and 4, 8, 16 and 32 groups for the quantized importance map $\mathbf{p}'$. Naturally, more groups do benefit the performance of inclined TCAE but also increase the model parameters and running time. In our implementation, we adopt 8 groups for $\mathbf{o}'$ and 32 groups for $\mathbf{p}'$. It is worth noting that, on the parameter sets $\mathcal{S}_1$ and $\mathcal{S}_2$, the symbols in the quantized importance maps mainly are some small values, i.e., 0, 1, 2, 3. Consequently, the quantized importance map is small by entropy and results in larger compression ratio.

Table 3.2: Running time (s) of different entropy prediction models on the codes and quantized importance maps generated by CWIC trained on seven different parameter sets.

Set	Task	Modified CABAC		Chapter 2		TCAE		Inclined TCAE	
		$\mathbf{o}'$	$\mathbf{p}'$	$\mathbf{o}'$	$\mathbf{p}'$	$\mathbf{o}'$	$\mathbf{p}'$	$\mathbf{o}'$	$\mathbf{p}'$
$\mathcal{S}_1$	Encoding	0.001	0.00003	0.092	0.021	0.021	0.005	0.021	0.005
	Decoding	0.001	0.00003	32.3	6.73	202.8	6.58	0.923	0.161
$\mathcal{S}_2$	Encoding	0.001	0.00003	0.178	0.021	0.021	0.005	0.021	0.005
	Decoding	0.001	0.00003	58.5	6.73	202.8	6.58	0.923	0.161
$\mathcal{S}_3$	Encoding	0.001	0.00003	0.368	0.021	0.021	0.005	0.021	0.005
	Decoding	0.001	0.00003	123.2	6.73	202.8	6.58	0.923	0.161
$\mathcal{S}_4$	Encoding	0.001	0.00003	0.483	0.021	0.021	0.005	0.021	0.005
	Decoding	0.001	0.00003	163.5	6.73	202.8	6.58	0.923	0.161
$\mathcal{S}_5$	Encoding	0.001	0.00003	0.665	0.021	0.021	0.005	0.021	0.005
	Decoding	0.001	0.00003	225.1	6.731	202.8	6.58	0.923	0.161
$\mathcal{S}_6$	Encoding	0.001	0.00003	0.782	0.021	0.021	0.005	0.021	0.005
	Decoding	0.001	0.00003	250.6	6.73	202.8	6.58	0.923	0.161
$\mathcal{S}_7$	Encoding	0.001	0.00003	0.931	0.021	0.021	0.005	0.021	0.005
	Decoding	0.001	0.00003	288.2	6.73	202.8	6.58	0.923	0.161

Running time

Using a computer with a Intel(R) Xeon(R) Processor E5-2620 v4, 64 GB of RAM and a NVIDIA TITAN Xp GPU, we test the running time (in seconds, s) of our method on the Kodak dataset with the image size 752×496 (or 496×752). The running time to generate the code $\mathbf{o}$ and to reconstruct input image from $\mathbf{o}$ are 0.024 and 0.032 s, respectively.

We further test the running time of lossless encoding and decoding of the code $\mathbf{o}'$ and the quantized importance map $\mathbf{p}'$. In particular, we consider four models, i.e., modified CABAC, convolutional entropy prediction model [38], TCAE, and inclined TCAE. The running time of modified CABAC is tested on CPU, while the running time of the other three models is accelerated with GPU in the caffe framework.

Table 3.2 lists the running time of the 4 lossless compression models. For both encoding and decoding, modified CABAC is the fastest model, but fails to exploit large context. In terms of encoding, TCAE and inclined TCAE are the second-fastest methods and are effective in large context modeling. However, when taking decoding into account, TCAE should decode the symbols in a sequential order and remains computationally inefficient. Benefited from the inclined plane based context, inclined TCAE is able to parallel decode the symbols within each inclined planes, and thus can be more than $100\times$ faster than TCAE for decoding. In comparison, convolutional entropy prediction model [38] only utilizes limited size of

context (i.e., $4 \times 5 \times 5$), is much inefficient for encoding, and is only comparable to TCAE for decoding. Furthermore, the running time of convolutional entropy prediction model [38] gradually increases from $\mathcal{S}_1$ to $\mathcal{S}_7$ when encoding and decoding $\mathbf{o}'$ but keeps the same when encoding and decoding $\mathbf{p}'$. This result can be ascribed to that convolutional entropy prediction model [38] independently handles each symbol to encode. From $\mathcal{S}_1$ to $\mathcal{S}_7$, more and more 1s are generated in the importance mask, which indicates that more symbols in $\mathbf{o}'$ are required to be processed by convolutional entropy prediction model [38]. As for $\mathbf{p}'$, all the symbols should be processed, and thus the parameter set does not affect running time.

3.5 Task Driven Image Compression

3.5.1 Model the Task Driven Objective Function

The proposed lossy image compression framework aims to do spatially variant bit allocation for different parts of the image with a leaned content weighted importance map. Without further constraints, the importance map is learned to the distortion of the decoded image. However, for some specific tasks, such as object detection [22, 23, 56, 57, 71] and human face recognition [7, 14, 70, 80, 85], the distortion is not the only metric. We further consider the region of interest (ROI) into account and modify the proposed content weighted image compression for task-driven image compression.

For the task-driven image compression, the regions related to detection or recognition performance are much more important than others and should be allocated more bites. Two constraints are introduced in this thesis to increase the importance of the ROI region in the importance map. The first constraint is strait-forward and modeled on the image space. We directly adopt a weighted mean square error distortion loss with a larger weight for the ROI region. The modified distortion loss is defined as,

$$\mathcal{L}_{\text{WMSE}}(\mathbf{z}, \mathbf{x}) = \frac{1}{3HW} \|\omega \cdot (D(\mathbf{z}) - \mathbf{x})\|_2^2, \tag{3.28}$$

where ω is the ROI related weight with the same size of the input image $\mathbf{x}$. ω is a $H \times W$ 2D ROI map. For the RGB color image, each channel of $\mathbf{x}$ shares the same ω. The value

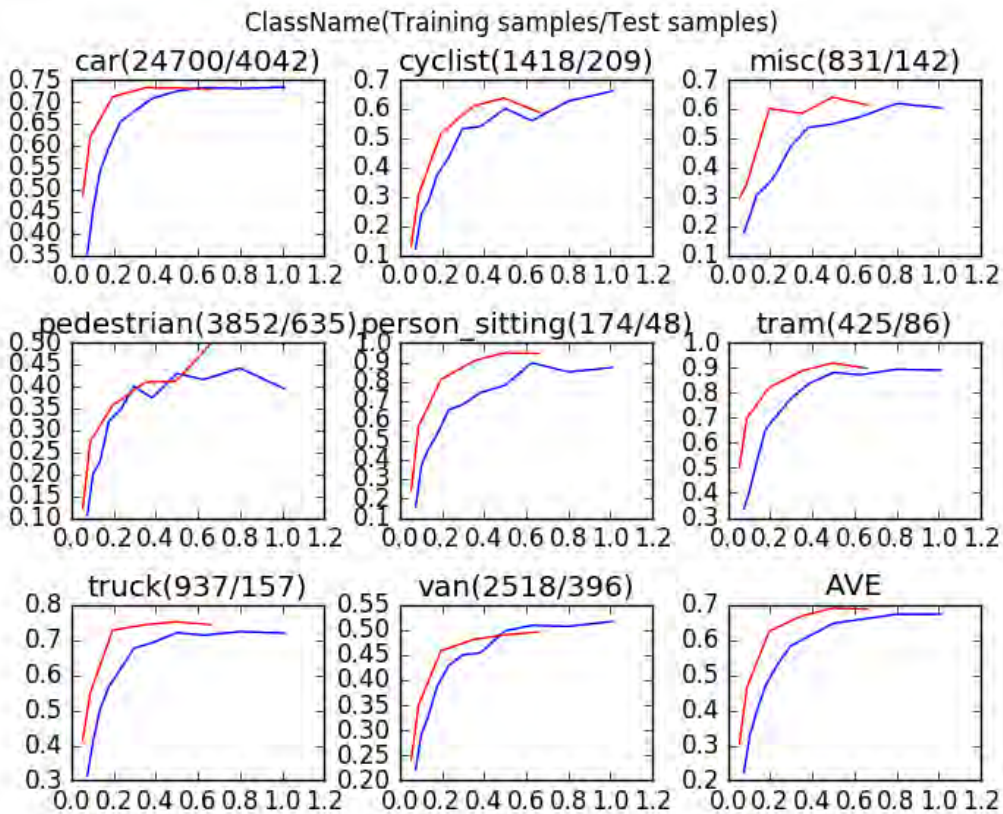


Figure 3.13: Rate-Performance curve on kitti set. The rate is evaluated with bits per pixel (bpp). And the detection performance is evaluated with average precision (AP). We show performance on 8 separate main objects in the set and the average of them. The blue curve represents BPG and the red curve is our method.

of ROI map is set as follows. For the ROI region, ω is set to be 3; otherwise, ω is 1. Considering that the surrounding parts of the ROI region may also make an influence on the performance of a specific task, we expend the ROI region by a little. In detail, we expend the width and the height of the ROI bounding box by $1/8$. Besides, we also model the importance of the ROI region in the importance map space and consider it in the rate loss. Since the importance maps are much smaller than the input images, we first down-sample the ROI map ω to the same size of importance map with a nearest neighbor down-sampling algorithm. In application, the ROI map is down-sampled by 8 times. Denoted by $\hat{\omega}$, the down-sampled ROI map is used to build a task-driven ratio loss. Different from the distortion loss, a smaller weight should be given to encourage allocating more bits to task-related regions. For convenience, we directly imply the inverse of $\hat{\omega}$ in building the task-driven rate loss.

$$\mathcal{L}_{WR}(\mathbf{x}) = \max\left\{0, \left(\frac{n}{L}\sum_{i,j}\frac{\mathbf{p}'_{i,j}}{\hat{\omega}_{i,j}} - r.nhw\right)\right\} \tag{3.29}$$

3.5.2 Experiments for Task Driven Image Compression

With the modified task-related distortion and rate loss, the content weighted image compression framework can be directly optimized for task-driven image compression. We test the task-driven image compression framework on two tasks, *i.e.*, object detection, and face recognition. For training the model for a specific task, we should first collect enough high quality and definition images and corresponding ROI bounding boxes. A poor quality image will bring compression artifacts for training. For the task of object detection, we adopt the kitti dataset. It offers 7, 481 lossless compressed image with corresponding bounding boxes. Due to the unavailable test set, we cut the image into two sets. The first 6,400 images are used for training and the last 1081 images are adopted for testing. To evaluate the detection performance. We choose a classic model, *i.e.*, faster RCNN, and train it on kitti. Then, the faster RCNN is used to test the compressed images on the test set and the popular metric for detection, *i.e.*, AP (Average Precision), is used as the performance metric. Due to unavail-

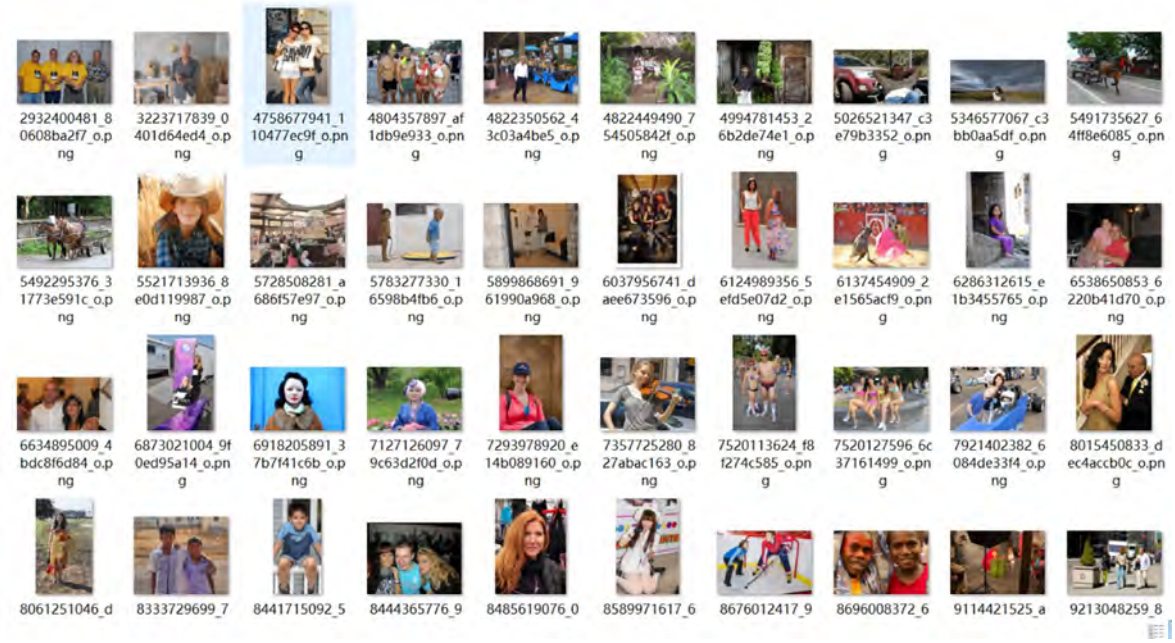


Figure 3.14: High quality face image dataset collected from the Flickr.

able codes, we fail to compare with other deep learning-based methods and only compare our task-driven model with the state-of-art image compression.

Figure 3.13 gives the performance of our method and BPG on kitti set on 8 main objects. On most of the objects, our model has significantly better object detection performance than the counterpart. On average, our model can save 46.7% bits with AP ranges from 0.4 to 0.65.

For the face recognition task, there is not such a high-quality face image dataset for training. We collect some high quality and definition images from the Flickr and downsample the image by 3 times to remove compression artifacts brought by JPEG compression. Then, we detect the faces and generate the ROI bounding boxes of human faces in the images with the toolkit dlib. Then, we look through all the data and the corresponding bounding boxes carefully. The images with a poor visual quality or incorrect bounding boxes are removed. We collected 2,646 high-quality images with faces on different scales and different backgrounds. All the faces have corresponding ROI bounding boxes. Figure 3.14 gives the example of the collected dataset.

For training the task-driven compression framework, we sample 600,000 image patches

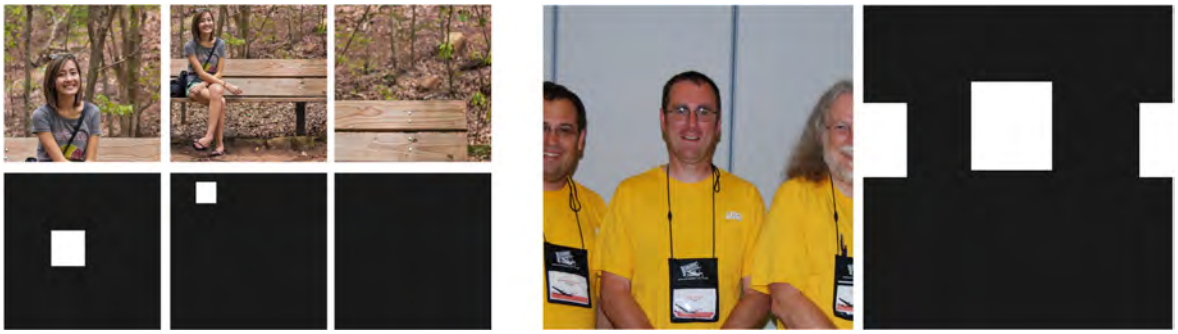


Figure 3.15: Samples for generating the training set.

with a size of 656 and corresponding bounding boxes. As shown in Figure 3.15, three conditions are considered in generating the patches. (i) Patches with a single face in different scales; (ii) Patches without faces; (iii) Patches with multiple faces.

For testing the models for face recognition or verification, we should collect another test set with the identity of the faces. Different from the training set, we first collected the names of 55 celebrities from the LFW dataset and then collected corresponding high-resolution images from Flickr for each person. Finally, we collect 742 images for testing. To evaluate the identification performance, the face tool-kit dlib is used to extract the features from the face images. The extracted feature is further compared with the reference feature of each person. If the distance between two features is less than a given threshold, the two faces are believed to belong to the same person. We compare the proposed model with BPG and give the results of face recognition accuracy with a different threshold. Figure 3.16 gives the results. With all the given thresholds, our model shows to have better performance than BPG.

Figure 3.17 shows the visual results of our model and BPG on three face images. Clear artifacts such as blurring can be observed in BPG results. With the guidance of the importance map, our method is more able to generate visually better face images.

3.6 Conclusion

In this chapter, we improved the learning-based content-weighted image compression framework with a better network structure for transforms, learned channel-wise multi-valued quan-

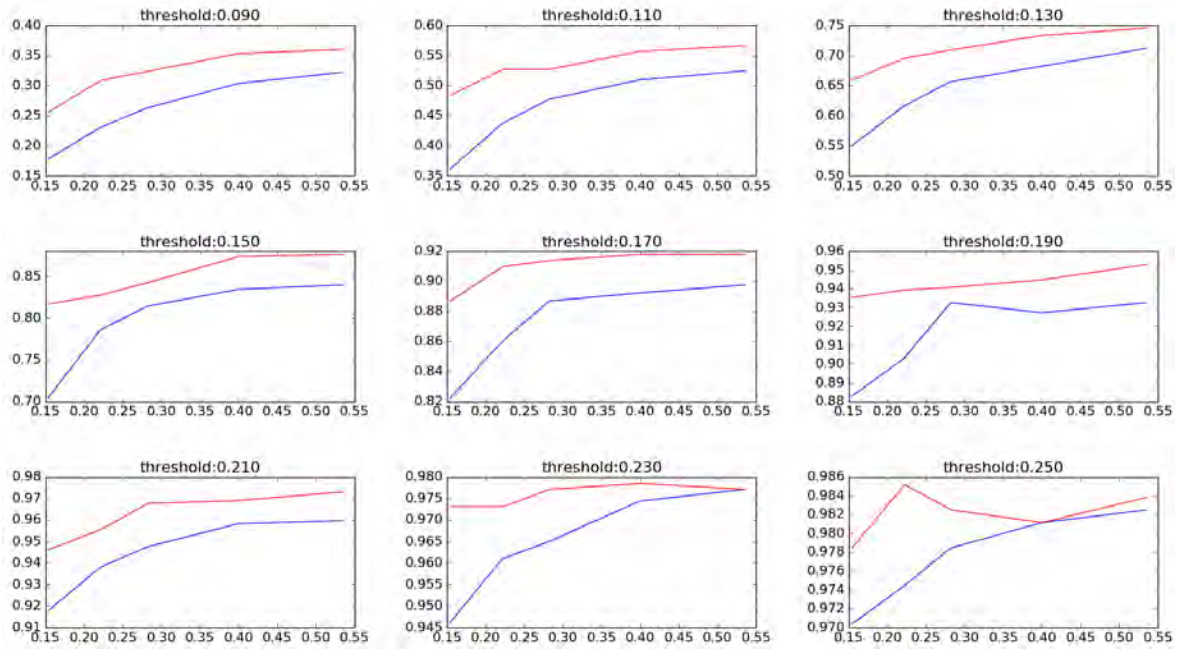


Figure 3.16: Rate-Performance curve on the collected dataset. The rate is evaluated with bits per pixel (bpp). And the detection performance is evaluated with Face recognition accuracy. We show performance on 9 separate thresholds for the face recognition. The blue curve represents BPG and the red curve is our method.

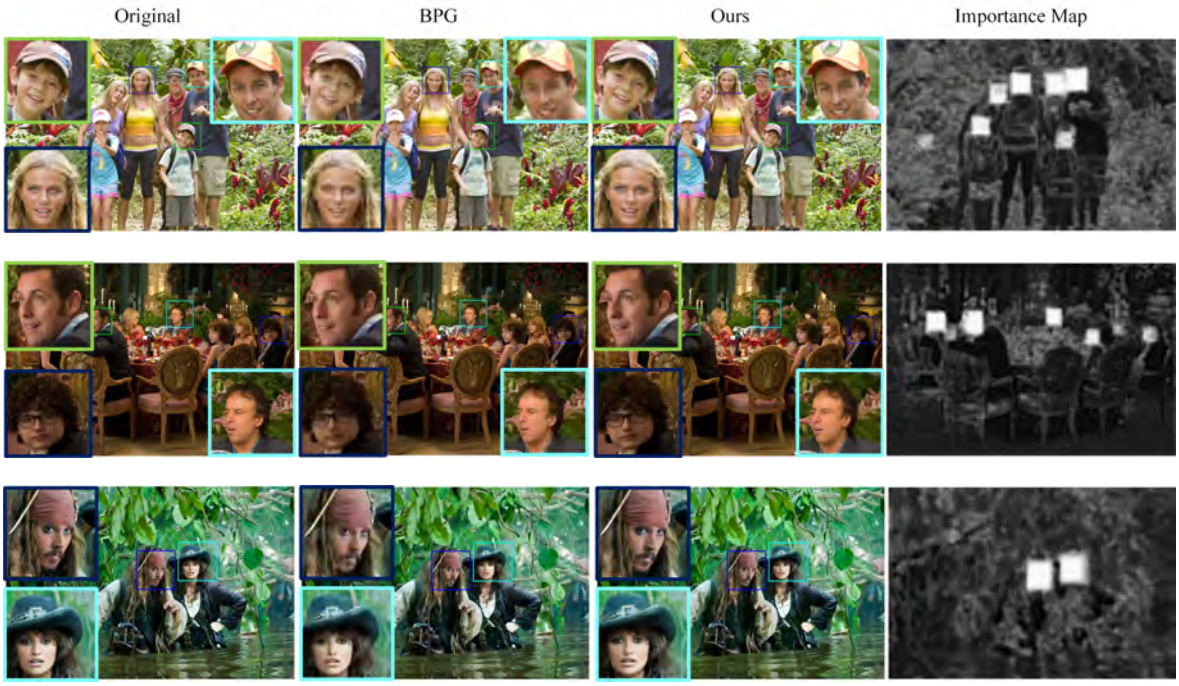


Figure 3.17: Visual quality of our model and BPG on three face images.

tization, and better context modeling. A better strategy is designed to optimize the importance map to the rate-distortion loss in end-to-end training of the whole framework. For modeling the context, TCAE is introduced to enlarge the context while maintaining the efficiency of the entropy prediction, and inclined TCAE is further presented to accelerate the decoding process. Experimental results show that our CWIC performs favorably in comparison to the state-of-the-art deep image compression methods and traditional image compression standards, and is effective in recovering salient structures and rich details, especially at lower bpp. We further extend the CWIC to task-driven image compression task and get extremely better performance not only on the performance of specified tasks but also on the visual quality.

CHAPTER 4

EFFICIENT AND EFFECTIVE CONTEXT-BASED CONVOLUTIONAL ENTROPY MODELING FOR IMAGE COMPRESSION

Precise estimation of the probabilistic structure of natural images, *i.e.*, entropy modeling, plays an essential role for both lossless and lossy image compression. Despite the remarkable success of recent end-to-end optimized image compression, the latent code representation is usually assumed to be fully statistically factorized to simplify entropy modeling. However, such an assumption generally does not hold and is unbeneficial to compression performance. In this work, we present context-based convolutional networks (CCNs) that exploit statistical redundancies in the codes for effective entropy modeling. In particular, a 3D zigzag coding order, as well as a 3D code dividing technique, are respectively introduced to define a proper context and to achieve parallel entropy decoding for efficiency, both of which boil down to place translation-invariant binary masks on convolution filters of CCNs. We demonstrate the power of CCNs for entropy modeling in both lossless and lossy image compression. For the former, we directly apply a CCN to the binarized plane representation of an image for estimating the Bernoulli distribution of each code. For the latter, the categorical distribution of each code is represented by a discretized mixture of Gaussian distributions, whose parameters are estimated by three CCNs. And the CCN-based entropy model can be jointly learned with analysis and synthesis transforms for optimized rate-distortion performance. Experiments on the Kodak and Tecnick datasets show that the proposed lossless and lossy image compression methods based on CCNs generally achieve better compression performance than existing methods with manageable computational complexity.

4.1 Introduction

Data compression has played a significant role in engineering for centuries [78]. Compression can be either lossless or lossy. Lossless compression allows perfect data reconstruction from compressed bitstreams to assign shorter codewords to more "probable" codes. Typical examples include Huffman coding [31], arithmetic coding [79], and range coding [45]. Lossy compression discards "unimportant" information and the definition of importance is application-dependent. For example, if the data (such as images and videos) are meant to be consumed by the human visual system, importance should be measured in accordance with human perception, discarding perceptually redundant data, while keeping those that are most visually noticeable. In lossy compression, one must face the rate-distortion trade-off, where the rate is computed by the entropy of the discrete codes [62] and the distortion is measured by a signal fidelity metric. A prevailing scheme in the context of lossy image compression is transform coding, which consists of three operations - transform, quantization, and entropy coding. Transforms are designed to be better-suited for exploiting aspects of human perception and map an image to a latent code space with several statistical advantages. Early transforms [5] are linear, invertible, and fixed for all bit rates; errors arise only from quantization. Recent transforms take the form of deep neural networks (DNNs) [11], aiming for more comprehensible and nonlinear representations. DNN-based transforms are mostly non-invertible, which, however, may encourage discarding perceptually less important image features during transformation. This gives us an opportunity to learn different transforms at different bit rates for better rate-distortion performance. Entropy coding is responsible for losslessly compressing the quantized codes into bitstreams for storage and transmission.

In either lossless or lossy image compression, a discrete probability distribution of the latent codes shared by the encoder and the decoder (*i.e.*, the entropy model) is essential in determining the compression performance. According to Shannon's source coding theorem [62], given a vector of code intensities $\boldsymbol{y} = \{y_0, \ldots, y_M\}$, the optimal code length of $\boldsymbol{y}$ should be $\lceil -\log_2 P(\boldsymbol{y}) \rceil$, where binary symbols are assumed to construct the codebook. Without further constraints, the general problem of estimating $P(\boldsymbol{y})$ in high-dimensional spaces is intractable, a problem commonly known as the curse of dimensionality. For this

reason, most entropy coding schemes assume $\boldsymbol{y}$ is fully statistically factorized with the same marginal distribution, leading to a code length of $\lceil -\sum_{i=0}^{M} \log_2 P(y_i) \rceil$. Alternatively, the chain rule in probability theory offers a more accurate approximation

$$P(\boldsymbol{y}) \approx \prod_{i=0}^{M} P(y_i | \mathrm{PTX}(y_i, \boldsymbol{y})), \tag{4.1}$$

where $\mathrm{PTX}(y_i, \boldsymbol{y}) \subset \{y_0, \ldots, y_{i-1}\}$ represents the partial context of y_i coded before it in $\boldsymbol{y}$. A representative example is the context-based adaptive binary arithmetic coding (CABAC) [44] in H.264/AVC, which considers the two nearest codes as the context and obtains noticeable improvements over previous image/video compression standards. As the size of $\mathrm{PTX}(y_i, \boldsymbol{y})$ becomes large, it is difficult to estimate this conditional probability by constructing histograms. For modelling the estimation of $P(y_i | \mathrm{PTX}(y_i, \boldsymbol{y}))$ with larger partial contexts, Recent methods such as PixelRNN [51] and PixelCNN [52] take advantage of DNNs in modeling long-range relations among pixels, but inevitably are computationally intensive.

In this work, we present context-based convolutional networks (CCNs) for effective and efficient parallel entropy modeling. Given $\boldsymbol{y}$, we specify a 3D zigzag coding order so that the most relevant codes of y_i can be included in its context. Parallel computation during entropy encoding is straightforward as the context of each code is known and readily available. However, this is not always the case during entropy decoding because the context of y_i is not ready for probability estimation until all codes in $\mathrm{PTX}(y_i, \boldsymbol{y})$ have been decoded sequentially, which is prohibitively slow. To address this issue, we introduce a 3D code dividing technique, which partitions $\boldsymbol{y}$ into multiple groups in compliance with the proposed coding order. The codes within each group are assumed to be conditionally independent given their respective contexts, and therefore can be decoded in parallel. In the context of CCNs, this amounts to applying properly designed translation-invariant masks to convolutional filters.

To validate the proposed CCNs, we combine them with arithmetic coding [79] for entropy coding. For lossless image compression, we convert the input grayscale image into eight binary planes and train a CCN to predict the Bernoulli distribution of y_i, optimizing for the entropy loss in information theory [18]. For lossy image compression, we parameterize the categorical distribution of y_i with a discretized mixture of Gaussian (MoG) distributions,

whose parameters (*i.e.*, mixture weights, means, and variances) are estimated by three CCNs, depending on its context. The CCN-based entropy model is jointly optimized with analysis and synthesis transforms (*i.e.*, mappings between raw pixel space and latent code space) over a database of training images for the tradeoff of rate-distortion performance. Experiments on the Kodak and Tecnick datasets show that our methods for lossless and lossy image compression perform favorably against image compression standards and DNN-based methods, especially at low bit rates.

4.2 CCNs for Entropy modelling

In this section, we present in detail the construction of CCNs for entropy modelling. We start with a fully convolutional network, consisting of T layers of convolutions followed by point-wise nonlinear activation functions. In order to perform efficient context-based entropy modelling, three assumptions are made on the network architecture:

- For a code block $\boldsymbol{y} \in \mathbb{R}^{M \times H \times W}$, where M, H, and W denote channel, height, and width, respectively, the output of the t-th layer convolution $\boldsymbol{v}^{(t)} \in \mathbb{R}^{M \times H \times W \times N_t}$, where N_t denotes the number of feature blocks to represent $\boldsymbol{y}$. By doing so, we are able to associate the feature point $v_{i,j}^{(t)}(p,q)$ in i-th channel and j-th feature block at spatial location (p,q) with $y_i(p,q)$ uniquely.

- Let $\mathrm{CTX}(y_i(p,q), \boldsymbol{y})$ be the set of codes encoded before $y_i(p,q)$ (full context), and $\mathrm{SS}(v_{i,j}^{(t)}(p,q))$ be the set of codes in the receptive field of $v_{i,j}^{(t)}(p,q)$ that contributes to its computation (support set), respectively. Then, $\mathrm{SS}(v_{i,j}^{(t)}(p,q)) \subset \mathrm{CTX}(y_i(p,q), \boldsymbol{y})$.

- For $\boldsymbol{y}' \subset \boldsymbol{y}$ and $y_i(p,q) \in \boldsymbol{y}'$, $\mathrm{CTX}(y_i(p,q), \boldsymbol{y}') \subset \mathrm{CTX}(y_i(p,q), \boldsymbol{y})$.

Assumption I establishes a one-to-many correspondence between the input code block $\boldsymbol{y}$ and the output feature representation $\boldsymbol{v}^{(T)}$. Assumption II ensures that the computation of $v_i^{(t)}(p,q)$ depends only on a subset of $\mathrm{CTX}(y_i(p,q), \boldsymbol{y})$. Together, the two assumptions guarantee the legitimacy of context-based entropy modeling in fully convolutional networks, which can be achieved by placing binary masks to convolution filters.

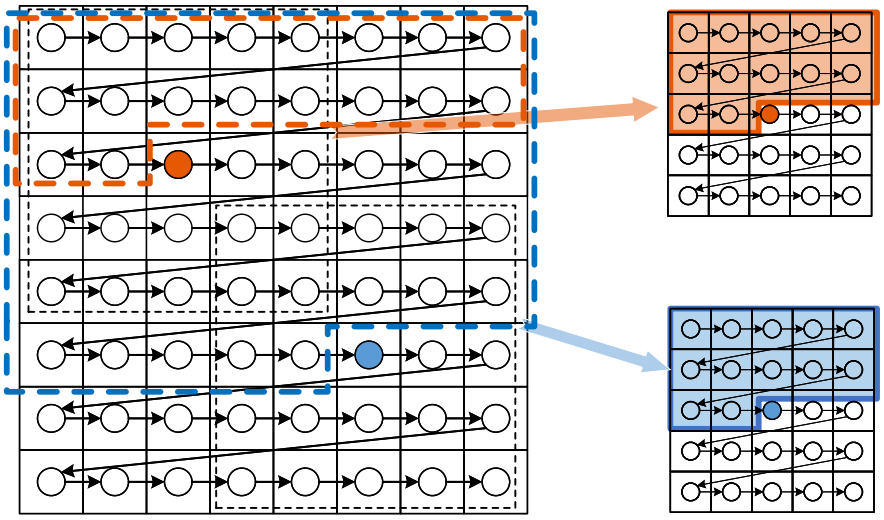


Figure 4.1: Illustration of 2D mask convolution in the input layer of the proposed CCN for entropy modeling. A raster coding order (left to right, top to bottom) and a convolution kernel size of 5×5 are assumed here. The orange and blue dashed regions indicate the full context of the orange and blue codes, respectively. In the right panel, we highlight the support sets of the two codes in corresponding colors, which share the same mask.

Then, $P(y_i(p,q)|\mathrm{CTX}(y_i(p,q),\boldsymbol{y}))$ can be computed from $\{(v_{ij}^{(T)}(p,q)\}$, which specify the parameters of the conditional distribution. Assumption III allows the masks to be translation-invariant, which can be achieved by properly designed coding orders. Specifically, we start with the case of a 2D code block, where $\boldsymbol{y} \in \mathbb{R}^{H \times W}$, and define mask convolution at the t-th layer as

$$v_i^{(t)}(p,q) = \sum_{j=1}^{N_t} \left(u_j^{(t)} * \left(m^{(t)} \odot w_{i,j}^{(t)} \right) \right)(p,q) + b_i^{(t)}, \tag{4.2}$$

where $*$ and $\odot$ denote 2D convolution and Hadamard product, respectively. $\boldsymbol{w}_{ij}^{(t)}$ is a 2D convolution filter and $\boldsymbol{m}^{(t)}$ is the corresponding 2D binary mask. According to Assumption I, the input $\boldsymbol{u}_i^{(t)}$ and the output $\boldsymbol{v}_i^{(t)}$ are of the same size as $\boldsymbol{y}$. The input code block $\boldsymbol{y}$ corresponds to $\boldsymbol{u}_0^{(0)}$.

For the input layer of a fully convolutional network, the codes to produce $v_i^{(0)}(p,q)$ is $\Omega_{p,q} = \{y(p+u,q+v)\}_{(u,v)\in\Psi}$, where Ψ is the set of local indexes centered at $(0,0)$. We choose

$$\mathrm{SS}(v_i^{(0)}(p,q)) = \mathrm{CTX}(y(p,q),\Omega_{p,q}) \subset \mathrm{CTX}(y(p,q),\boldsymbol{y}), \tag{4.3}$$

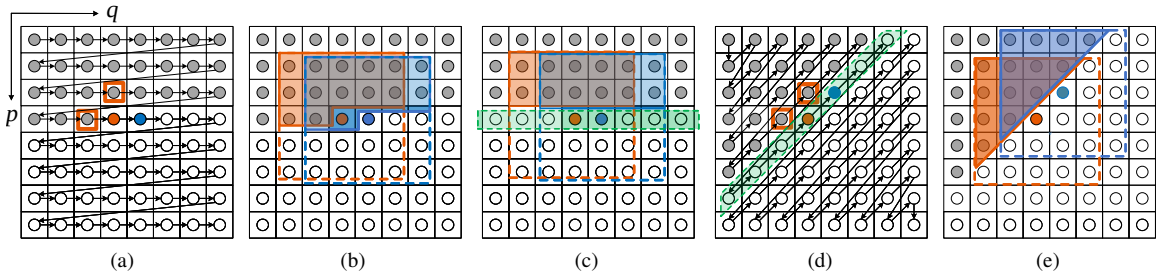


Figure 4.2: Illustration of code dividing techniques in conjunction with different coding orders for a 2D code block. The orange and blue dots represent two nearby codes. The gray dots denote codes that have already been encoded, while the white circles represent codes yet to be encoded. (a) Raster coding order adopted in many compression methods. (b) Support sets of the orange and blue codes, respectively. It is clear that the orange code is in the support set of the blue one, and therefore should be decoded first. (c) Code dividing scheme for the raster coding order. By removing the dependencies among codes in each row, the orange and blue codes can be decoded in parallel. However, the orange code is excluded from the support set of the blue one, which may hinder entropy estimation accuracy. (d) Zigzag coding order and its corresponding code dividing scheme. The two codes in the orange squares that are important for the orange code in entropy prediction are retained in its partial context. (e) Support sets of the orange and blue codes in compliance with the zigzag coding order.

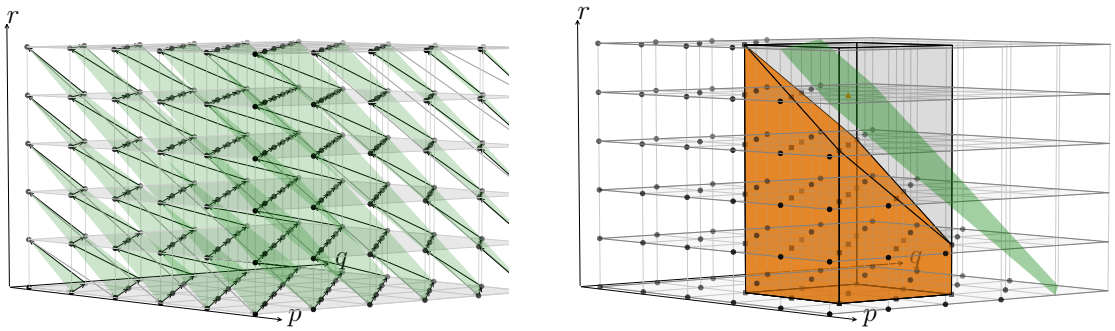
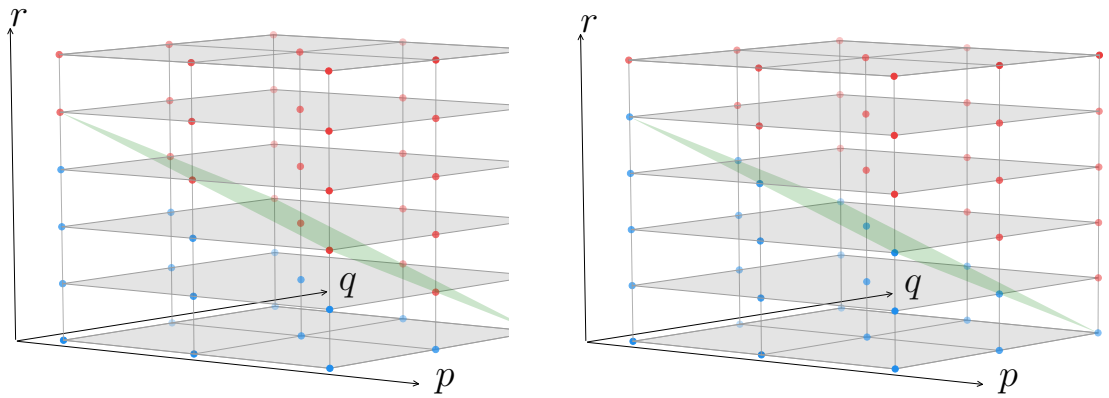


Figure 4.3: Illustration of the proposed 3D zigzag coding order and 3D code dividing technique. (a) Each group in the shape of a diagonal plane is highlighted in green. Specifically, $\mathrm{GP}_k(\boldsymbol{y}) = \{y_r(p,q)|r+p+q=k\}$ are encoded first, than $\mathrm{GP}_{k+1}(\boldsymbol{y})$. Within $\mathrm{GP}_k(\boldsymbol{y})$, we first process codes along the line $p+q=k$ by gradually decreasing p. We then process codes along the line $p+q=k-1$ with the same order. The procedure continues until we sweep codes along the last line $p+q=\max(k-r,0)$ in $\mathrm{GP}_k(\boldsymbol{y})$. (b) Support sets of the orange codes with a spatial filter size of 3×3.



(a) (b)

Figure 4.4: Illustration of masked codes with $M = 6$, $r = 2$, and a filter size of 3×3. Blue dots represent codes activated by the mask and red dots indicate the opposite. The only difference lies in the green diagonal plane. (a) Input layer. (b) Hidden layer.

which can be achieved by setting

$$m^{(0)}(u,v) = \begin{cases} 1, & \text{if } \Omega_{p,q}(u,v) \in \mathrm{CTX}(y(p,q), \boldsymbol{y}) \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

Fig. 4.1 illustrates the concepts of full context $\mathrm{CTX}(y(p,q), \boldsymbol{y})$, support set $\mathrm{SS}(v^{(0)}(p,q))$, and translation-invariant mask $\mathbf{m}^{(0)}$, respectively. At the t-th layer, if a code $y(p+u, q+v) \in \mathrm{CTX}(y(p,q), \boldsymbol{y})$, we have

$$\begin{aligned} \mathrm{SS}(u_j^{(t)}(p+u, q+v)) \subset & \mathrm{CTX}(y(p+u, q+v), \boldsymbol{y}) \\ \subset & \mathrm{CTX}(y(p,q), \boldsymbol{y}), \end{aligned} \tag{4.5}$$

where the first line follows by induction and the second line follows from the definition of context. That is, as long as $y(p+u, q+v)$ is in the context of $y(p,q)$, we are able to compute $v_i^{(t)}(p,q)$ from $u_j^{(t)}(p+u, q+v)$ without violating Assumption II. Unlike the input layer, $u_j^{(t)}(p,q)$ with $t > 0$ is also generated from $\mathrm{CTX}(y(p,q), \boldsymbol{y})$, and can be used to compute $v_i^{(t)}(p,q)$. Therefore, the mask at the t-th layer can be defined as

$$\mathbf{m}^{(t)}(u,v) = \begin{cases} \mathbf{m}^{(0)}(u,v), & \text{if } (u,v) \neq (0,0) \\ 1, & \text{otherwise.} \end{cases} \tag{4.6}$$

With the proposed CCN with translation-invariant masks in Eqn. (4.4) and Eqn. (4.6), $\boldsymbol{y}$ can be efficiently encoded by exploiting fully convolutional network for shared computation. However, it remains very difficult for shared and parallel computation in the entropy

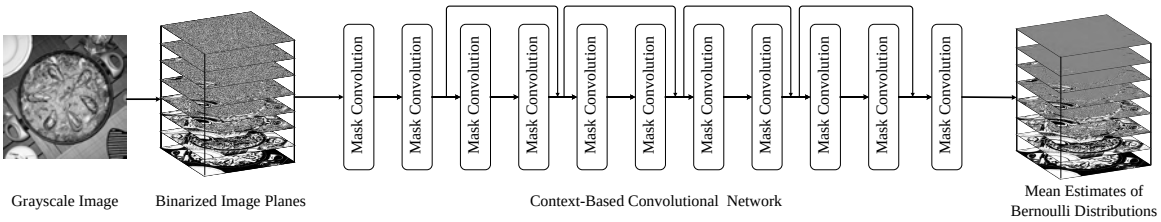


Figure 4.5: The proposed CCN-based entropy model for lossless image compression. The grayscale image $\boldsymbol{x}$ is first converted to bit-plane representation $\boldsymbol{y}$, which is fed to the network to predict the mean estimates of Bernoulli distributions $P(y_r(p,q)|\mathrm{SS}(v_r(p,q)))$. The size of convolution filters and the number of feature blocks in intermediate layers are set to $S \times S$ and N, respectively. Each convolution layer is followed by a parametric ReLU nonlinearity, except for the last layer, where a sigmoid function is applied. From the mean estimates, we find that for most significant bit-planes, our model makes more confident predictions closely approaching local image structures. For least significant bit-planes, our model is less confident, producing mean estimates close to 0.5.

decoding phase. As shown in Fig. 4.2 (a) and (b), the two nearby codes in the same row (highlighted in orange and blue, respectively) cannot be decoded simultaneously because the orange code is in the support set (or context) of the blue code given the raster coding order. To speed up entropy decoding with parallel computation, we may further remove dependencies among codes at the risk of model accuracy. Specifically, we partition $\boldsymbol{y}$ into K groups, namely, $\mathrm{GP}_0(\boldsymbol{y}), \ldots, \mathrm{GP}_K(\boldsymbol{y})$, and remove dependencies among the codes within the same group, resulting a partial context $\mathrm{PTX}(y(p,q), \boldsymbol{y}) = \{\mathrm{GP}_0(\boldsymbol{y}), \ldots, \mathrm{GP}_{k-1}(\boldsymbol{y})\}$ for $y(p,q) \in \mathrm{GP}_k(\boldsymbol{y})$. As a result, all codes in the k-th group share the same partial context, and can be decoded in parallel. Note that code dividing schemes are largely constrained by the pre-specified coding order. For example, if we use a raster coding order, it is straightforward to divide $\boldsymbol{y}$ by row. In this case, $y(p, q-1)$ (p and q index vertical and horizontal directions, respectively), which is extremely important in predicting the probability of $y(p,q)$ according to CABAC [44], has been excluded from its partial context. To make a good trade-off between modelling efficiency and accuracy, we adopt a zigzag coding order as shown in Fig. 4.2 (d), where $\mathrm{GP}_k(\boldsymbol{y}) = \{y(p,q)|p+q=k\}$ and $\mathrm{PTX}(y(p,q), \boldsymbol{y}) = \{y(p',q')|p'+q' < k\}$. As such, we retain the most relevant codes in the partial context for better entropy modelling (see Fig. 4.9 for quantitative results). Accordingly, the mask at the t-th layer becomes

$$m^{(t)}(u,v) = \begin{cases} m^{(0)}(u,v), & \text{if } u+v \neq 0 \\ 1, & \text{otherwise.} \end{cases} \tag{4.7}$$

Now, we extend our discussion to a 3D code block, where $\boldsymbol{y} \in \mathbb{R}^{M\times H\times W}$. Fig. 4.3 (a) shows the proposed 3D zigzag coding order and 3D code dividing technique (zoom in for improved visibility). Specifically, $\boldsymbol{y}$ is divided into $K = M+H+W-3$ groups in the shape of diagonal planes, where the k-th one is specified by $\mathrm{GP}_k(\boldsymbol{y}) = \{y_r(p,q)|r+p+q=k\}$. The partial context of $y_r(p,q) \in \mathrm{GP}_k(\boldsymbol{y})$ is defined as $\mathrm{PTX}(y_r(p,q),\boldsymbol{y}) = \{y_{r'}(p',q')|r'+p'+q'<k\}$. We then write mask convolution in the 3D case as

$$v_{i,r}^{(t)}(p,q) = \sum_{j=1}^{N_t}\sum_{s=1}^{M}\left(u_{j,s}^{(t)} * \left(m_{r,s}^{(t)} \odot w_{i,j,r,s}^{(t)}\right)\right)(p,q) + b_i^{(t)}, \tag{4.8}$$

where $\{i,j\}$ and $\{r,s\}$ are indexes for the feature block and channel dimensions, respectively. For the 2D case, each layer shares the same mask ($M=1$). When extending to 3D code blocks, each channel in a layer shares a mask, and there are a total of M 3D masks. For the input layer, the codes to produce $v_{i,r}^{(0)}(p,q)$ is $\Omega_{p,q} = \{y_s(p+u,q+v)\}_{(u,v)\in\Psi,0\le s<M}$, based on which we define the mask as

$$m_{r,s}^{(0)}(u,v) = \begin{cases} 1, & \text{if } \Omega_{p,q}(u,v) \in \mathrm{PTX}(y_r(p,q),\boldsymbol{y}) \\ 0, & \text{otherwise.} \end{cases} \tag{4.9}$$

For the t-th layer, we modify the mask to include the current diagonal plane

$$m_{r,s}^{(t)}(u,v) = \begin{cases} m_{r,s}^{(0)}(u,v), & \text{if } s+u+v \neq r \\ 1, & \text{otherwise,} \end{cases} \tag{4.10}$$

as shown in Fig. 4.4, where we highlight the difference in the green diagonal plane.

4.3 CCN-Based Entropy Models for Lossless Image Compression

In this section, we combine our CCN-based entropy model with the arithmetic coding algorithm for lossless image compression.

As a starting point, we binarize the grayscale image $\boldsymbol{x} \in \mathbb{R}^{H\times W}$ to obtain a 3D code block

$$y_r(p,q) = \left\lfloor \frac{x(p,q)}{2^{7-r}} \right\rfloor \bmod 2, \quad r = 0,1,\ldots,7, \tag{4.11}$$

where we index the most significant bit-plane with $r=0$.

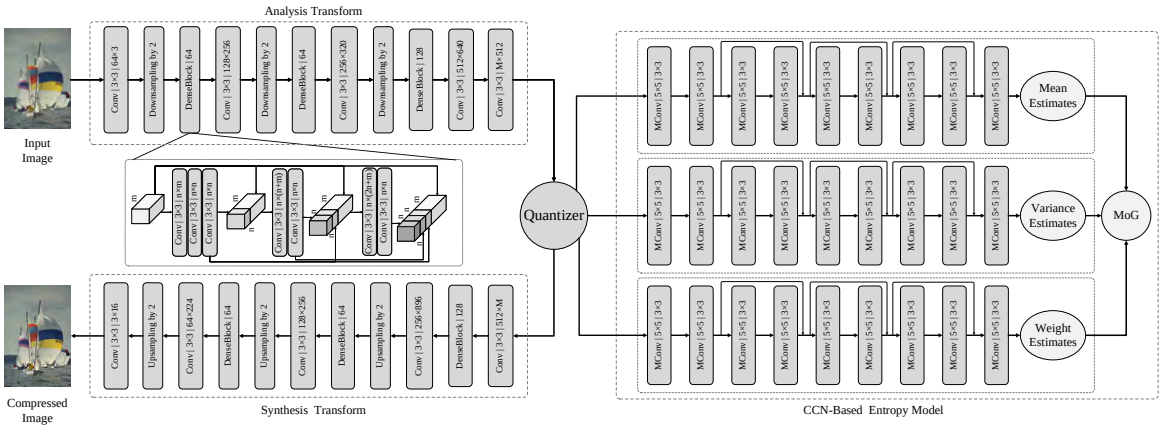


Figure 4.6: The architecture of the proposed lossy image compression method, which consists of an analysis transform g_a, a non-uniform and trainable quantizer g_d, a CCN-based entropy model, and a synthesis transform g_s. Conv: regular convolution with filter support ($S \times S$) and number of channels (output×input). Down-upsampling: implemented jointly with the adjacent convolution (also referred to as stride convolution). DenseBlock: m matches the input channel number of the preceding convolution. n is the channel number in DenseBlock set empirically. MConv: mask convolution used in our CCNs with filter support ($S \times S$) and number of feature blocks (output×input). Note that the number of channels is fixed in MConv, and is determined by that of $\bar{\boldsymbol{y}}$.

Our CCN takes $\boldsymbol{y}$ as input and produces a feature block $\boldsymbol{v}$ (the superscript (T) is omitted for notation convenience) of the same size to compute the mean estimates of Bernoulli distributions $P(y_r(p,q)|\mathrm{SS}(v_r(p,q))$. Fig. 4.5 shows the network architecture, which consists of eleven mask convolution layers with parametric ReLU nonlinearities in between. The last convolution responses undergo a sigmoid nonlinearity to constrain the dynamic range within $[0,1]$. We make four residual connections as suggested in [27] to accelerate training. We will experiment with two hyper-parameters in CCN: the size of convolution filters for all layers S and the number of feature blocks in hidden layers N.

To optimize the network parameters, which are collectively denoted by $\boldsymbol{\theta}$, we adopt the expected code length as the empirical loss

$$\ell(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{y}}\Big[-\sum_{r,p,q}\big(\mathbb{1}(y_r(p,q)=1)\log_2(v_r(p,q)) + \mathbb{1}(y_r(p,q)=0)\log_2(1-v_r(p,q))\big)\Big], \tag{4.12}$$

where $\mathbb{1}(\cdot)$ is an indicator function and the expectation may be approximated by averaging over a mini-batch of training images. Finally, we implement our own arithmetic coding with

the learned CCN-based entropy model to compress $\boldsymbol{y}$ to bitstreams, and report performance using actual bit rates. This facilitates comparison against widely used image compression standards.

4.4 CCN-Based Entropy Models for Lossy Image Compression

In lossy image compression, our objective is to minimize a weighted sum of rate and distortion, $\ell_r + \lambda \ell_d$, where λ governs the trade-off between the two terms. As illustrated in Fig. 4.6, our compression method consists of four components: an analysis transform g_a, a quantizer g_d, a CCN-based entropy model, and a synthesis transform g_s. The analysis transform g_a takes a color image $\boldsymbol{x}$ as input and produces the latent code representation $\boldsymbol{y}$. g_a consists of three convolutions, each of which is followed by down-sampling with a factor of two. A dense block [30] comprised of seven convolutions is employed after each down-sampling. After the last dense block, we add another convolution layer with M filters to produce $\boldsymbol{y}$. Empirically, the parameter M sets the upper bound of the bit rate that a general DNN-based compression method can achieve. The parameters of g_a constitute the parameter vector $\boldsymbol{\phi}$ to be optimized.

The synthesis transform g_s is a mirror of the analysis transform. Particularly, the depth-to-space reshaping [63, 69] is adopted to up-sample the feature maps. The last convolution layer with three filters is responsible for producing the compressed image in RGB space. The parameters of g_s constitute the parameter vector $\boldsymbol{\psi}$ to be optimized.

For the quantizer g_d, we parameterize its quantization centers for the r-th channel by $\{\omega_{r,0}, \ldots, \omega_{r,L-1}\}$, where L is the number of quantization centers and $\omega_{r,0} \leq \ldots \leq \omega_{r,L-1}$. The monotonicity of $\boldsymbol{\omega}$ can be enforced by a simple re-parameterization based on cumulative functions. Given a fixed set of $\boldsymbol{\omega}$, we perform quantization by mapping $y_r(p,q)$ to its nearest center that minimizes the quantization error

$$\bar{y}_r(p,q) = g_d(y_r(p,q)) = \arg\min_{\{\omega_{r,l}\}} \|y_r(p,q) - \omega_{r,l}\|_2^2. \tag{4.13}$$

g_d has zero gradients almost everywhere, which hinders training via back-propagation. Taking inspirations from binarized neural networks [17, 55, 86], we make use of an identify

mapping $\hat{g}_d(y_r(p,q)) = y_r(p,q)$ as a more continuous proxy to the step quantization function. During training, we use g_d and $\hat{g}_d$ in the forward and backward passes, respectively.

The quantization centers $\boldsymbol{\omega}$ should be optimized by minimizing the mean squared error (MSE),

$$\ell_q(\boldsymbol{\omega}) = \frac{1}{MHW} \sum_{r,p,q} \|y_r(p,q) - \bar{y}_r(p,q))\|_2^2, \tag{4.14}$$

which is essentially a k-means clustering problem, and can be solved efficiently by the Lloyd's algorithm [41]. Specifically, we initialize $\boldsymbol{\omega}$ using uniform quantization, which appears to work well in all experiments. To make parameter learning of the entire model smoother, we adjust $\boldsymbol{\omega}$ using stochastic gradient descent instead of a closed-form update.

Without prior knowledge of the categorical distributions of the quantized codes $\bar{\boldsymbol{y}}$, we choose to work with discretized MoG distributions, whose parameters are predicted by the proposed CCNs. We write the differentiable MoG distribution with C components as

$$\bar{y}_r(p,q) \sim \sum_{i=0}^{C-1} \pi_i \mathcal{N}(\bar{y}_r(p,q); \mu_i, \sigma_i^2), \tag{4.15}$$

where π_i, μ_i and δ_i^2 are the mixture weight, mean, and variance of the i-th component, respectively. Then,

$$P(\bar{y}_r(p,q)) = \int_{\Delta} \sum_{i=0}^{C-1} \pi_i \mathcal{N}(\xi; \mu_i, \sigma_i^2) d\xi. \tag{4.16}$$

where Δ is the quantization bin that $\bar{y}_r(p,q)$ lies in.

Next, we describe the proposed entropy model in lossy image compression, which is comprised of three CCNs with the same structure, as shown in Fig. 4.6. Each CCN consists of nine mask convolutions with three residual connections to produce C feature blocks, matching the number of components in MoG. They separately output mixture weights, means and variances to build the discretized MoG distributions. The network parameters of our CCNs constitute the parameter vector $\boldsymbol{\theta}$ to be optimized.

Finally, we are able to write the empirical rate-distortion objective for the parameters

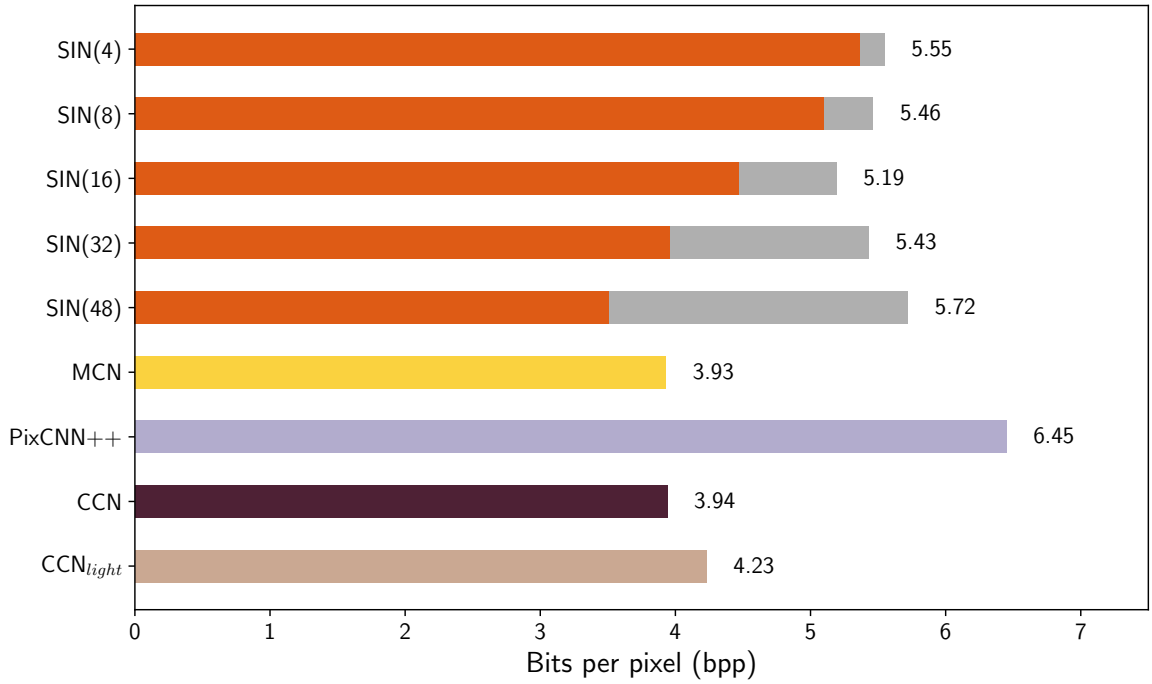


Figure 4.7: Bit rates (in terms of bpp) of different DNN-based entropy models for lossless image compression on the Kodak dataset. SIN(M) refers to a side information network that allocates M output channels to represent side information. The orange and gray bars represent the bit rates from the image and the side information, respectively.

$\{\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\theta}\}$ as

$$\ell(\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}} \Bigg[- \sum_i \log_2 P_{\bar{y}_i}\Big(g_d\Big(g_a(\boldsymbol{x}; \boldsymbol{\phi})\Big); \boldsymbol{\theta}\Big) + \lambda \ell_d\Big(g_s\Big(g_d\big(g_a(\boldsymbol{x}; \boldsymbol{\phi})\big); \boldsymbol{\psi}\Big), \boldsymbol{x}\Big)\Bigg]. \tag{4.17}$$

ℓ_d is the distortion term, which is more preferable to be assessed in perceptual space. In this paper, we optimize and evaluate our lossy image compression methods using standard MSE and a perceptual metric MS-SSIM [77]. Similar to lossless image compression, we combine the optimized entropy model with arithmetic coding and measure the rate using actual bit rates.

4.5 Experiments

In this section, we test the proposed CCN-based entropy models in the lossless and lossy image by comparing it to state-of-the-art image coding standards and recent deep image

compression algorithms. We first collect $10,000$ high-quality and high-definition images from Flickr, and down-sample them to further reduce possibly visible artifacts. We crop $1,280,000$ grayscale patches of size 128×128 and $640,000$ color patches of size $3 \times 256 \times 256$ as the training sets for lossless and lossy image compression, respectively. We test our models on two independent datasets - Kodak and Tecnick [9], which are widely used to benchmark image compression performance. The Caffe implementations along with the pre-trained models are made available at https://github.com/limuhit/SCAE.

4.5.1 Lossless Image Compression

We train our CCN-based entropy model using the Adam stochastic optimization package [33] by minimizing the objective in Eqn. (4.12). We start with a learning rate of 10^{-4} and subsequently lower it by a factor of 10 when the loss plateaus, until 10^{-6}. The (actual) bit rate in terms of bits per pixel (bpp) is used to quantify the compression performance, which is defined as the ratio between the total amount of bits used to code the image and the number of pixels in the image. A smaller bpp indicates better performance. For example, an uncompressed grayscale image has eight bpp.

We first compare the proposed CCNs with mask convolutional networks (MCNs) [36, 46], PixelCNN++ [61], and side information networks (SINs) [12] for entropy modelling. As a special case of CCNs, MCNs specify the raster coding order without using any code dividing technique (see Fig. 4.2). We implement our version of MCN that inherits the network architecture from the CCN with $N = 16$ (number of feature blocks) and $S = 5$ (filter support). PixelCNN++ [61] is originally designed for image inpainting as a generative model. Here we adapt it for entropy modeling. Starting from a down-sampled grayscale image with a factor of four, we use a DNN of similar model complexity compared with our CCN to predict the categorical distributions from the down-sampled image. SINs summarize the side information of $\bar{\boldsymbol{y}}$ with a separate DNN, which is helpful in probability estimation. We adopt a DNN-based autoencoder of similar model complexity as our CCN (including three stride convolutions and two residual connections) to generate the side information, which is further quantized and compressed with arithmetic coding for performance evaluation. We test

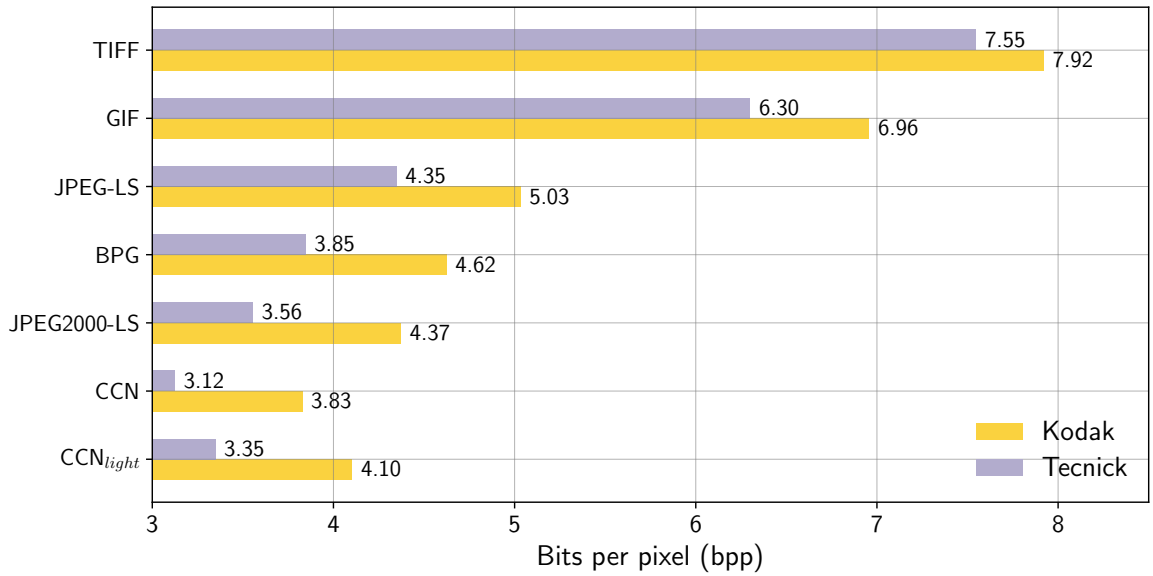


Figure 4.8: Bit rates of CCN in comparison with lossless image compression standards on the Kodak and Tecnick datasets.

Table 4.1: Running time in seconds of different DNN-based entropy models on the Kodak dataset with image size of 752×496

	SIN	MCN	PixelCNN++	CCN	CCN$_{light}$
Encoding	0.155	0.323	0.121	0.323	0.074
Decoding	0.155	3079.68	0.121	35.28	0.984

five variants of SINs with different amount of side information by changing the number of output channels M. All competing models are trained on the same dataset described at the beginning of Section 4.5. We also introduce a light-weight version of our method, which we name CCN$_{light}$, by making the network architecture lighter (with $N = 3$ and $S = 3$) and by dividing the input image into non-overlapping patches for parallel processing.

Fig. 4.7 shows the bit rates of the competing methods on the Kodak dataset. The proposed CCN matches the best performing model MCN, which suggests that with the proposed zigzag coding order and code dividing technique, CCN does not exclude the most important codes from the partial context of the current code being processed. The bit rates of SINs come from two parts - the image itself and the side information. It is clear from the figure that increasing the amount of side information leads to bit savings of the image, at the cost of additional bits introduced to code the side information. In general, it is difficult to determine the amount of side information for optimal compression performance. Pixel-

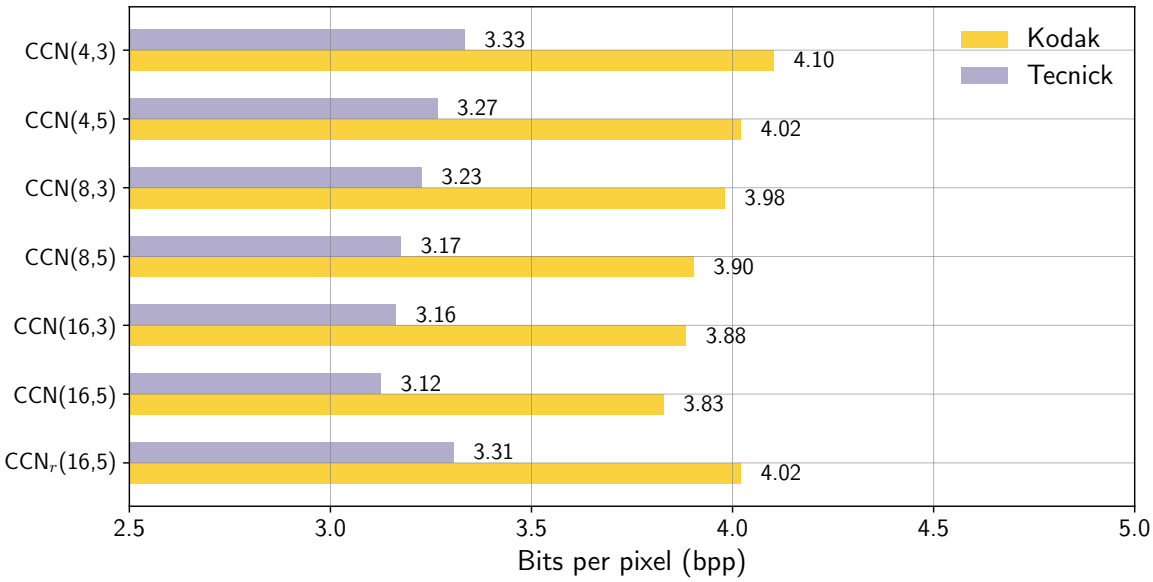


Figure 4.9: Ablation study of CCN on the Kodak and Tecnick datasets. CCN(N,S) denotes the CCN with N feature blocks and $S \times S$ filter size. CCN$_r$ represents the CCN with the raster coding order and the corresponding code dividing technique (see Fig. 4.2).

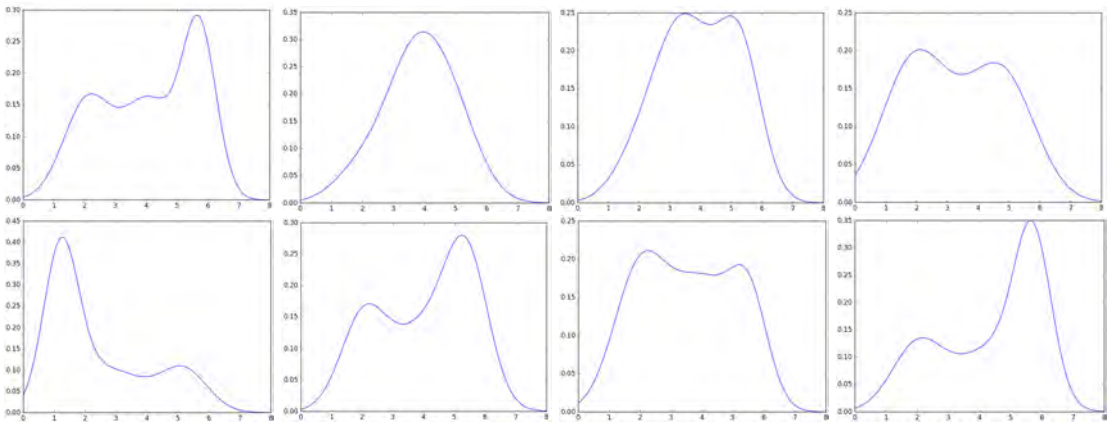


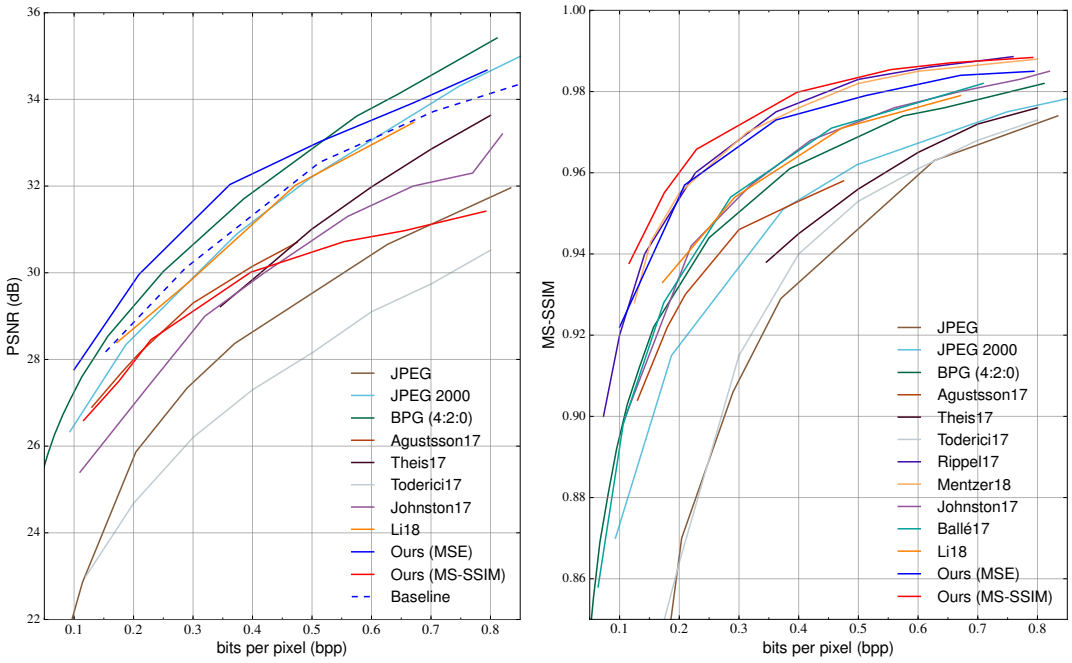
Figure 4.10: Visualization of the learned continuous MoG distributions of sample codes before discretization. It is clear that most of them are multimodal and therefore cannot be well fit using a single Gaussian.

CNN++ can also be regarded as a special case of SINs, whose side information is a small image without further processing, leading to the worst performance.

We also compare CCN with the widely used lossless image compression standards, including TIFF, GIF, PNG, JPEG-LS, JPEG2000-LS, and BPG. All test results are generated by MATLAB2017. From Fig. 4.8, we find that CCN (along with its light-weight version CCN$_{light}$) overwhelms all competing methods on the Kodak/Tecnick dataset, achieving more than $5.9\%/6.2\%$ bit savings compared to the best lossless image compression standard, JPEG2000-LS.

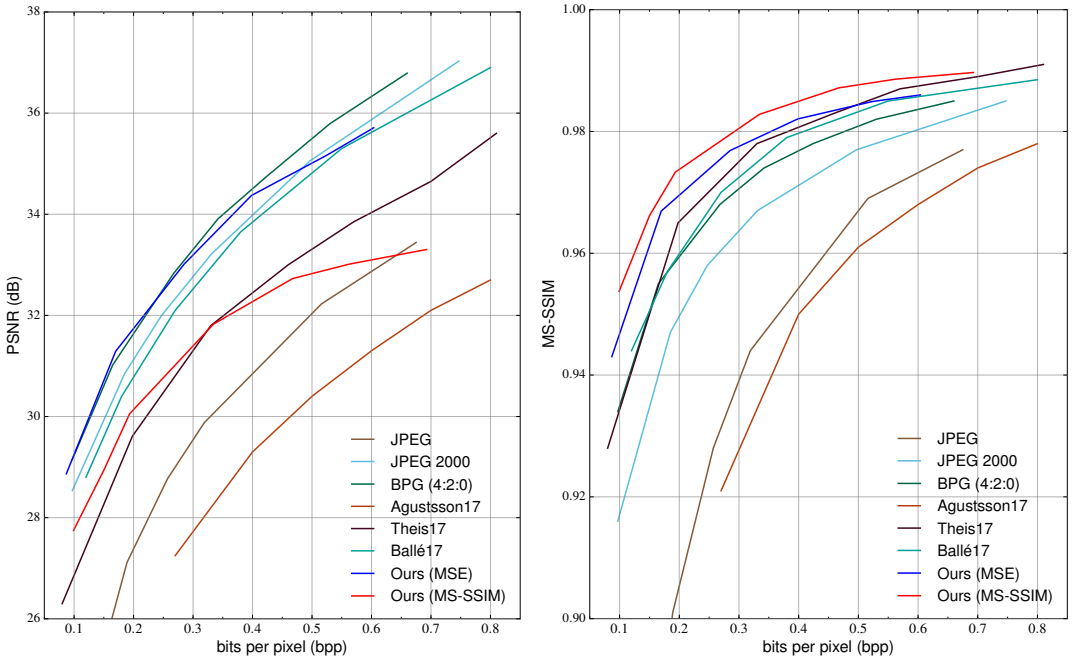
The running time of the four types of DNN-based entropy models is tested on an NVIDIA TITAN Xp machine, whose results on the Kodak dataset are listed in Table 4.1. For encoding, CCN$_{light}$ enjoys the fastest speed, followed by PixelCNN++ and SIN (the best performing variant). Despite similar encoding time, they have substantially different decoding complexities. PixelCNN++ has the fastest decoder, followed by SIN. Due to the sequential decoding nature, MCN is the slowest, taking nearly one hour to decode a grayscale image of size 752×496. Our CCN achieves a significant improvement upon MCN with the proposed code dividing technique, while maintaining nearly the same bit rates. Moreover, CCN$_{light}$ speeds up CCN more than 30 times, striking a good balance model between efficiency and accuracy.

We conduct thorough ablation experiments to analyze the impact of individual components to final compression performance. Fig. 4.9 shows the bit rates of CCNs with three different numbers of feature blocks ($N \in \{4, 8, 16\}$) and two filter sizes ($S \in \{3, 5\}$). When the filter size and the network depth are fixed, adding more feature blocks effectively increases the model capability and thus boosts the compression performance. Similarly, using a larger filter size with fixed network depth and feature block number increases the partial context, leading to better entropy modeling. Moreover, we replace the proposed zigzag coding order in CCN with the raster coding order, whose model is denoted by CCN$_r$. From Fig. 4.9, we observe that the performance of CCN$_r$ drops significantly, only comparable to the CCN with four feature blocks, which verifies the advantages of the proposed coding



(a) (b)

Figure 4.11: Rate-distortion curves of different compression methods on the Kodak dataset. (a) PSNR. (b) MS-SSIM. Baseline denotes our method with separately optimized transforms and entropy model for MSE.



(a) (b)

Figure 4.12: Rate-distortion curves of different compression methods on the Tecnick dataset. (a) PSNR. (b) MS-SSIM.

order.

Table 4.2: Running time in second of our CCN-based entropy model at six bit rates on the Kodak dataset

Average bpp	0.100	0.209	0.362	0.512	0.671	0.794
Encoding	0.013	0.025	0.044	0.066	0.085	0.103
Decoding	0.116	0.227	0.457	0.735	1.150	1.232

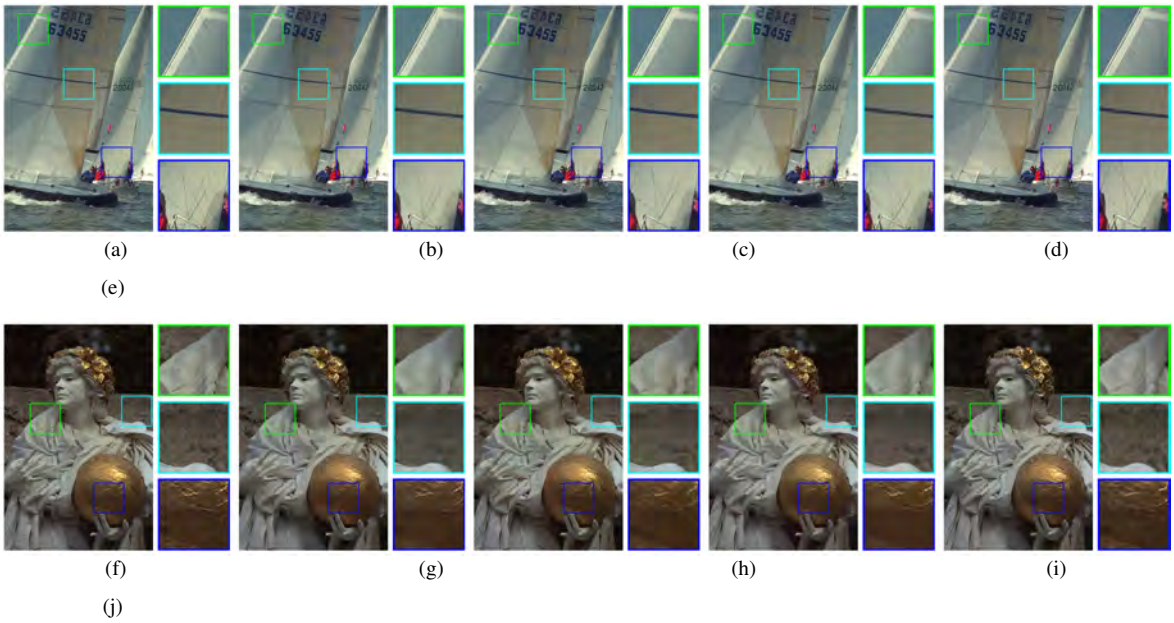


Figure 4.13: Compressed images by different compression methods on the Kodak dataset. The quantitative measures are in the format of "bpp / PSNR / MS-SSIM". (a) Uncompressed "Sailboat" image. (b) Ballé17 [11]. 0.209 / 31.81 / 0.962. (c) Chapter 2. 0.244 / 31.97 / 0.966. (d) BPG. 0.220 / 33.19 / 0.963. (e) Ours optimized for MS-SSIM. 0.209 / 31.01 / 0.978. (f) Uncompressed "Statue" image. (g) Ballé17. 0.143 / 29.48 / 0.942. (h) Chapter 2. 0.115 / 29.35 / 0.938. (i) BPG. 0.119 / 29.77 / 0.935. (j) Ours optimized for MS-SSIM. 0.116 / 28.05 / 0.954.

4.5.2 Lossy Image Compression

In lossy image compression, the analysis transform g_a, the non-uniform quantizer g_d, the CCN-based entropy model, and the synthesis transform g_s are jointly optimized for rate-distortion performance. In the early stages of training, the probability $P(\bar{\boldsymbol{y}})$ may change rapidly, which makes it difficult to keep track of, causes instability in learning the entropy model. We find that this issue can be alleviated by a simple warmup strategy. Specifically, g_a and g_s are trained using the distortion term ℓ_d only for the first epoch. We then fix g_a and train the CCN-based entropy model until it reasonably fits the current distribution of the codes. After that, we end-to-end optimize the entire method for the rest epochs. We use

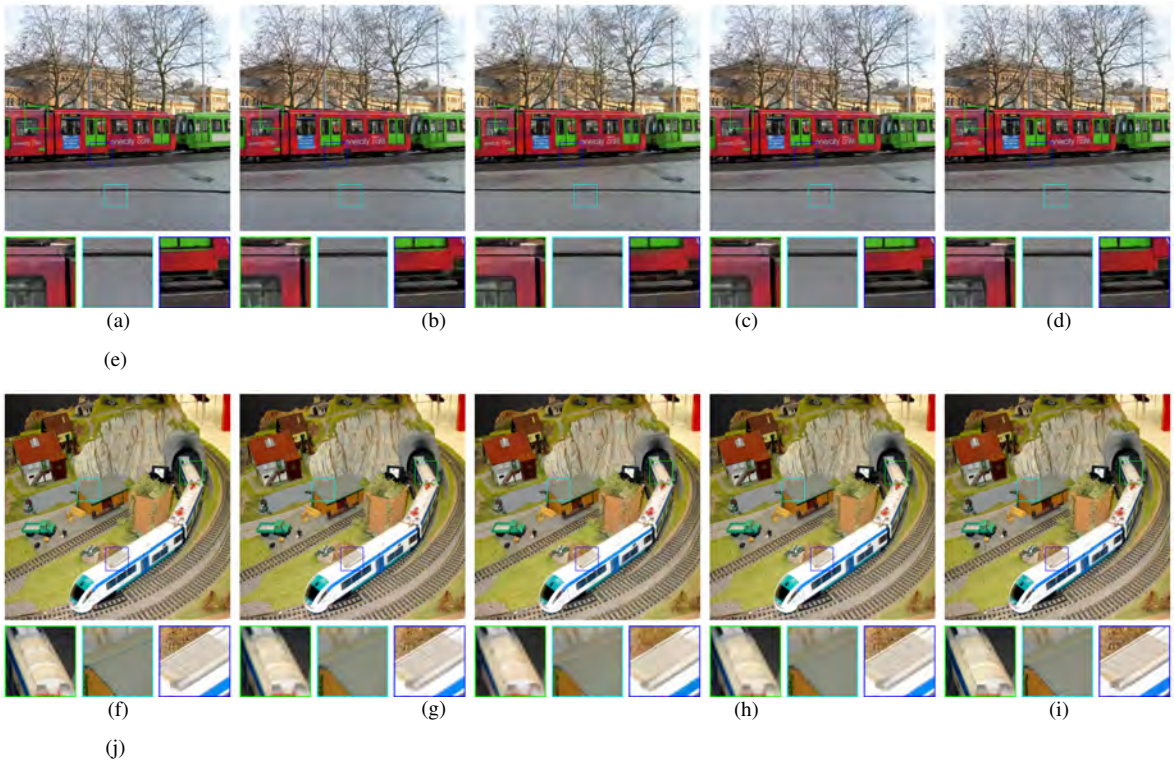


Figure 4.14: Compressed images by different compression methods on the Tecnick dataset. The quantitative measures are in the format of "bpp / PSNR / MS-SSIM". (a) Uncompressed "Bus" image. (b) JPEG2K. 0.199 / 24.41 / 0.914. (c) Chapter 2. 0.224 / 23.41 / 0.908. (d) BPG. 0.208 / 25.36 / 0.928. (e) Ours(MS-SSIM). 0.198 / 23.71 / 0.951. (f) Uncompressed "Toy train" image. (g) JPEG2K. 0.201 / 28.29 / 0.917. (h) Chapter2. 0.189 / 26.83 / 0.899. (i) BPG. 0.210 / 29.25 / 0.933. (j) Ours(MS-SSIM). 0.198 / 28.08 / 0.949.

Adam with a learning rate of 10^{-5} and gradually lower it by a factor of 10, until 10^{-7}. The number of quantization centers is $L = 8$ and the number of Gaussian components in MoG is $C = 3$. Fig. 4.10 shows the learned continuous distributions of some codes, which are typically complex and multimodal. This optimization is performed separately for each λ and each distortion measure. We optimize twelve models for six bitrates and two distortion metrics (MSE and MS-SSIM). MSE is converted to the peak signal-to-noise ratio (PSNR) for quantitative analysis.

We compare our methods with existing image coding standards and recent DNN-based compression models. These include JPEG [73], JPEG2000 [64], BPG [13], Agustsson17 [2], Theis17 [67], Toderici17 [69], Rippel17 [58], Mentzer18 [46], Johnston17 [32], Ballé17 [11], and the results of chapter 2. Both JPEG (with 4:2:0 chroma subsampling) and JPEG2000 are based on the optimized implementations in MATLAB2017. For BPG, we adopt the latest version from its official website with the default setting. When it comes to DNN-based models for lossy image compression, the implementations are generally not available. Therefore, we report the results from their respective papers.

Fig. 4.11 shows the rate-distortion curves on the Kodak dataset. We find that our method optimized for MSE outperforms all competing methods at low bit rates (< 0.5 bpp), except for BPG. When optimized for MS-SSIM, our method performs on par with Rippel17 and is much better than the rest. Fig. 4.12 shows the rate-distortion curves on the Tecnick dataset, where we observe similar trends for both PSNR and MS-SSIM. An interesting observation is that when we continue increasing the bit rate, PSNR/MS-SSIM starts to plateau, which may be due to the limited model capability. Without any constraint on the rate ($\lambda = \infty$) and quantization ($L = \infty$), our method optimized for MSE only reaches 38.2dB on the Kodak dataset, which may be considered as an empirical upper-bound for our network structure. Preliminary results indicate that increasing the depth and width of the network leads to performance improvements at high bit rates.

We visually compare the compressed images by our method against Ballé17, results from chapter 2, JPEG2K, and BPG. Fig. 4.13 and Fig. 4.14 show sample compressed results

on the Kodak and Tecnick datasets, respectively. JPEG2K and BPG exhibit artifacts (such as blocking, ringing, blurring, and aliasing) that are common to all handcrafted transform coding methods, reflecting the underlying linear basis functions. Ballé17 is effective at suppressing ringing artifacts at the cost of over-smoothing fine structures. Chapter 2 allocates more bits to preserve large-scale strong edges, while tends to eliminate small-scale localized features (*e.g.*, edges, contours, and textures). In contrast, our method generates compressed images with more faithful details and less visible distortions. Also, we investigate the effectiveness of the joint optimization of the transforms and the CCN-based entropy model. A baseline method is introduced, which first optimizes the transforms for MSE ($\lambda = \infty$), and trains the CCN-based entropy model with learned (and fixed) transforms. As shown in Fig. 4.11 (a), the baseline method underperforms the jointly optimized one by 1dB.

We report the running time of our method at six bit rates on the Kodak dataset using the same machine. It takes 0.024 second to generate $\bar{\boldsymbol{y}}$ and 0.032 second to reconstruct the image. The entropy coding time is listed in Table 4.2, which we see that more time is needed to encode and decode images at higher bit rates. With the help of the proposed code dividing technique, our method performs entropy decoding in around one second for images of size 752×496.

4.6 Conclusion

We have introduced CCNs for context-based entropy modeling. Parallel entropy encoding and decoding are achieved with the proposed coding order and code dividing technique, which can be efficiently implemented using mask convolutions. We test the CCN-based entropy model (combined with arithmetic coding) in both lossless and lossy image compression. For the lossless case, our method achieves the best compression performance, which we believe arises from the more accurate estimation of the Bernoulli distributions of the binary codes. For the lossy case, our method offers improvements both visually and in terms of rate-distortion performance over image compression standards and recent DNN-based models.

The application scope of the proposed CCN is far beyond building the entropy model in image compression. As a general probability model, CCN appears promising for several image processing applications. For example, we may use CCN to learn a probability model $P(\boldsymbol{x})$ for natural images, and use it as a prior in Bayesian inference to solve various vision tasks such as image restoration [54, 83], image quality assessment [43, 75], and image generation[15, 51].

CHAPTER 5

LEARNING CONTEXT-BASED NON-LOCAL ENTROPY MODELLING FOR IMAGE COMPRESSION

Inspired by chapter 4, we further explore the entropy modeling of image compression. Despite the remarkable success of recent end-to-end optimized image compression, the code representation is usually assumed to depend on some side information or local context. However, such an assumption fails to take the global similarity inner the context into account, and thus hinders the estimation of the entropy. In this chapter, we introduce the non-local similarity of the codes that exploit statistical redundancies in the context for effective entropy modeling. Besides a mask convolutional network which focuses on the local context, a non-local attention block is introduced to combine the local auto-regressive representation and global content similarity weighted estimation in modeling the probability of the codes. Besides, the width of the transforms defined as the minimum number of channels in the output of each layer is essential in training low distortion models. An U-net block is proposed to increase the width of the auto-encoder network with manageable network complexity. By supposing each code follows a mixture of Gaussian distribution based on its context, the analysis transform, synthesis transform, and the context-based non-local entropy model are jointly optimized in an end-to-end manner. Experiments on the Kodak and Tecnick datasets show that the proposed lossy image compression methods based on non-local entropy modeling and U-net blocks generally achieve better compression performance.

5.1 Introduction

Image compression is a fundamental problem in engineering which has been studied for centuries. With the population of artificial intelligence and social media, the requirement of sharing and storing high-definition media also increases explosively, which places an ad-

ditional burden on the storage and the bandwidth of the Internet. To relieve the pressure brought by a large amount of media data, more effective image compression methods are still needed.

Deep learning has shown great power in fitting complex problems and achieved unprecedented successes in various vision tasks like image restoration [20, 21, 81, 83], image quality assessment [43], and image generation [51, 52], which throws light on designing better lossy image compression methods with the deep learning toolkit. Since deep networks are natural transforms, nearly all of the recent learned deep image compression frameworks are transform coding frameworks and learned by optimizing a joint rate-distortion optimization problem. The transform coding framework consists of three components, *i.e.*, transform, quantization, and entropy coding. Transforms aim to map the image into a latent code space where the codes are easy to be compressed. For early transform coding frameworks such as JPEG [73] and JPEG2000 [64], the transforms are linear, invertible and fixed for all bit rates. Distortion only arises from the quantization. Differently, recent transforms are modeled with deep neural networks (DNNs), which are non-linear and complex but non-invertible. DNN-based transforms would indeed generate better codes, but extra distortion arises at the same time. The lossy image compression models are a trade-off between rate and distortion. For models in low bit rate region where the distortion are quite large, such extra distortion could be ignored. But for models that aim for small distortion, a better network structure is needed to reduce the extra distortion brought by DNN-based transforms.

To reduce the distortion introduced by the DNN-based transforms, the width of the transforms, *i.e.*, the minimum number of channels in the output of each layer, should be increased to keep as much information from the input image. For a deep transform with hundreds of layers, the growing of width will inevitably increase the computational complexity and GPU memory consumption. An U-net like block is introduced to process features in each scale and help reduce the time complexity and memory usage in the transforms. With the paired downsampling, upsampling operations and skip connection, the U-net like architecture is not only able to speed up the transforms but also could combine the information in different scales, facilitate the information propagation and ease the training of the transforms.

In lossy image compression, another important issue is to model the rate loss of the learned framework. Building an accurate discrete probability distribution function for each code is essential in determining the compression performance. According to Shannon's source coding theorem [62], given a sequence of codes $\boldsymbol{y} = \{y_0, \ldots, y_M\}$, the optimal code length of $\boldsymbol{y}$ should be $\lceil -\log_2 P(\boldsymbol{y}) \rceil$ with the codebook constructed by binary symbols. Without further constraints, estimating $P(\boldsymbol{y})$ in high-dimensional spaces is intractable and surfers the curse of dimensionality. To relieve this problem, most entropy coding schemes directly assume the codes in $\boldsymbol{y}$ are independent and follow the same marginal distribution, resulting in a code length of $\lceil -\sum_{i=0}^{M} \log_2 P(y_i) \rceil$. Alternatively, a more accurate approximation could be given with the chain rule

$$P(\boldsymbol{y}) \approx \prod_{i=0}^{M} P(y_i | \mathrm{CTX}(y_i, \boldsymbol{y})), \tag{5.1}$$

where $\mathrm{CTX}(y_i, \boldsymbol{y}) \subset \{y_0, \ldots, y_{i-1}\}$ represents the context of y_i, *i.e.*, all the codes coded before it in $\boldsymbol{y}$. Taking the representative traditional method, context-based adaptive binary arithmetic coding (CABAC) [44], as an example, it considers two nearest codes as the context for entropy prediction and gets noticeable performance improvement over previous image compression standards, which supports the effectiveness of the context in modeling the probabilistic distribution of the codes.

As the possible conditions grow explosively with the increase of the size of $\mathrm{CTX}(y_i, \boldsymbol{y})$, it is hard to estimate the conditional probability with the traditional histogram-based methods. Recent deep autoregressive models including RNN [47], LSTM [66] in natural language processing and PixelRNN [51], PixelCNN [52] for image generation, can model long-range dependency among sequential data or pixels and employ much larger context for entropy modeling but have to process the pixels/codes with a raster scanning order. The heavy computational burden makes them not practical in many real applications. Taking both of the effectiveness and efficiency into account, another general auto-regressive entropy model, context-based convolutional networks (CCNs) [37], introduces a specially defined scanning order and corresponding context. With a given code dividing scheme, the CCNs can perform parallel decoding.

However, all of the CNN based entropy models have a definite receptive filed and only take the local context inner the receptive filed into account. Besides, the CNN based entropy models only fit the probabilistic distribution of the target code in a receptive filed and the context with a large amount of training data. The content similarity of the codes for a single image is usually ignored. Supposing the target code to be regressed $y_r(p,q)$ is in the r-th channel and at the space of (p,q) in a 2D plane of a 3D code block $\boldsymbol{y}$. Inspired by the non-local means used in image denoising, we introduce non-local similarity into the context-based entropy modeling by regressing a code, $y_r(p,q)$, with a weighted sum over all the codes in the r-th channel of the context. And the weights are produced according to the content similarity between these codes and the target code. Due to the definition of the context, the target code, $y_r(p,q)$, is unknown and could not directly be adopted to evaluate the content similarity.We alternatively adopt a weighed L2 distance among the available code vectors with the same 2D position, *i.e.*, $(y_0(p,q),\ldots,y_{r-1}(p,q))$, as the proxy to evaluate the content similarity. A context-based non-local attention block is further introduced to combine the local auto-regressive representations and the global content weighted estimations.

Without too much hypothesis on the probabilistic distribution function of the codes, the context-based entropy of the codes is estimated by parametrizing the distribution of each code with a mixture of Gaussian (MoG) distribution, whose parameters are predicted with context-based non-local entropy model. We jointly optimize transforms, *i.e.*, analysis transform and synthesis transform, with U-net blocks and the context-based non-local entropy model to the trade-off of rate-distortion performance in an end-to-end manner. Experiments on the Kodak and Tecnick datasets show that the proposed method can outperform state-of-art lossy image compression standards.

5.2 Context Based Non-local Entropy modelling

Modeling the entropy of the code from its context is an auto-regression problem where the probability distribution function (PDF) of the code is regressed from the code context. In entropy coding, all the codes should be processed with a given scanning order.

$\mathrm{CTX}(y_r(p,q), \boldsymbol{y})$ denoted by the context of the code $y_r(p,q)$, which is defined as all the codes processed before $y_r(p,q)$ in entropy coding. Here, r indicates the channel of the code and (p,q) is its position in the 2D code plane. Fig. 5.1 (b) gives an example of the context of the red target code in the 3D code block with a raster scanning order.

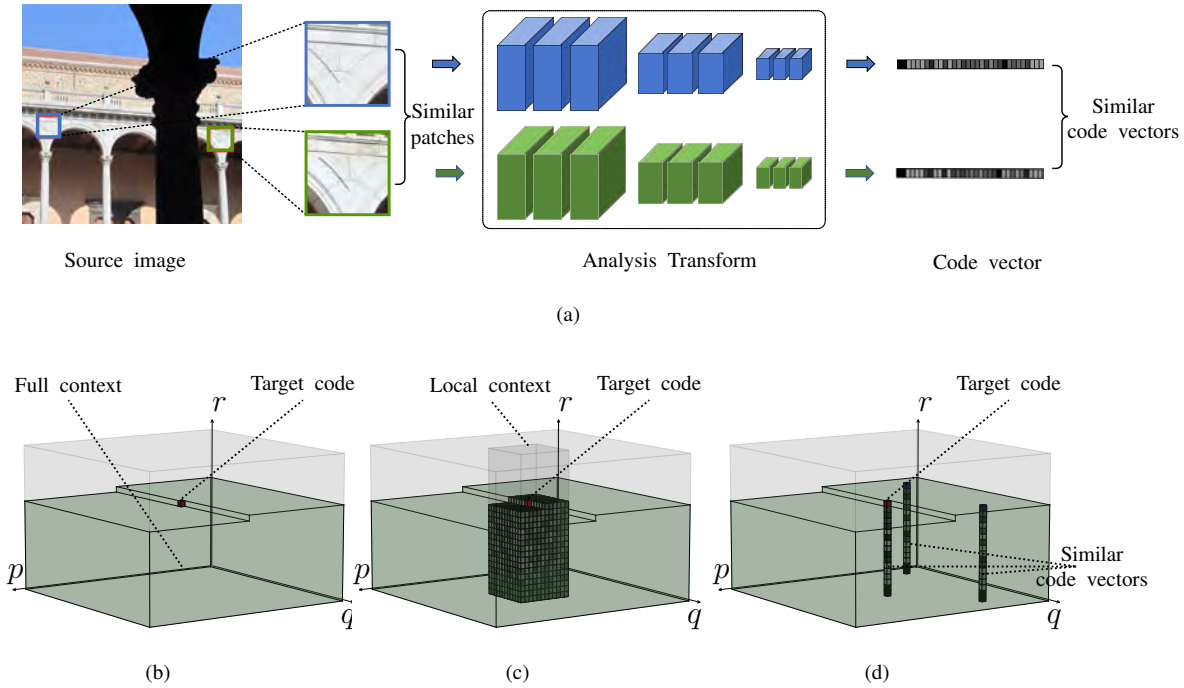


Figure 5.1: Illustration for context-based non-local entropy modeling. (a) indicates the non-local similarity among the codes generated by the analysis transform. (b) shows the context of a target code in 3D code block $\boldsymbol{y}$ with a raster scanning order. (c) gives the local context used in CNN-based entropy modeling. (d) illustrates the non-local similarity in the context.

Due to the efficiency of CNNs in processing 2D and 3D data, the previous context-based entropy modeling usually adopts mask CNNs to estimate the PDFs of the codes. As shown in Fig. 5.1 (c), limited by the structure, CNNs can only focus on part of the context in the receptive field, *i.e.*, local context, to predict the PDFs. The content similarity of the codes and global information are generally ignored in the CNN based entropy model. How to introduce global information and content similarity for context-based entropy modeling? Non-local is a good solution.

Non-local is a popular image prior which is adopted in many image restoration tasks for making use of global information. As indicated in some low-level image processing methods, like BM3D, there are many similar patches in the image. Thus, the codes generated

by an analysis transform with similar image patches in the receptive filed should also be similar. In Fig. 5.1 (a), we visualize two code vectors, *i.e.*, the vector of codes in the same 2D position, and the corresponding image patches for the generating the code vectors, which clearly support our hypothesis that the non-local similarity prior holds for the code vectors generated by the analysis transform. As shown in Fig. 5.1(d), if we could find similar code vectors as the target code vector, it is reasonable to predict the unknown target code (red block) with the corresponding known code (blue block).

In this chapter, we dig out the non-local similarity inner the context and introduce a context-based non-local block into CNN based entropy models to combine the local and global information for context-based entropy modeling. In the following, we will first introduce the CCNs from chapter 4 for its efficiency and effectiveness in modeling local context and then define the context-based non-local operation for exploiting global information in the context.

5.2.1 CCNs for Local Context Based Entropy modelling

Denoted by $\boldsymbol{y} \in \mathbb{R}^{M \times H \times W}$ a 3D code block generated by the analysis transform. M, H, and W separately are the channel, height, and width of the code block. The output of the t-th CCN layer is a 4D tensor $\boldsymbol{v}^{(t)} \in \mathbb{R}^{M \times H \times W \times N_t}$ with N_t feature blocks to represent $\boldsymbol{y}$. Each feature $v_{i,r}^{(t)}(p,q)$ in r-th channel and i-th feature block at spatial location (p,q) is uniquely connected with $y_r(p,q)$ and only convey information from the $\mathrm{CTX}(y_r(p,q), \boldsymbol{y})$.

Due to the overlap of context, the codes inner the 3D code block should be processed in the serial order in decoding, which greatly reduces the efficiency of the CNN based entropy model. In CCNs, a 3D zigzag scanning order together with a novel code dividing scheme is introduced to increase the parallel ability and result in a special partial context, *i.e.*, $\mathrm{PTX}(y_r(p,q), \boldsymbol{y})$. In the partial context, codes are divided into $K = M + H + W - 3$ non-overlap groups with $\mathrm{GP}_k(\boldsymbol{y}) = \{y_r(p,q) | r + p + q = k\}$ denotes the k-th group. Then, the partial context of $y_r(p,q) \in \mathrm{GP}_k(\boldsymbol{y})$ is defined as $\mathrm{PTX}(y_r(p,q), \boldsymbol{y}) = \{y_{r'}(p',q') | r' + p' + q' < k\}$. Codes inner the same group are supposed to be independent, share the same

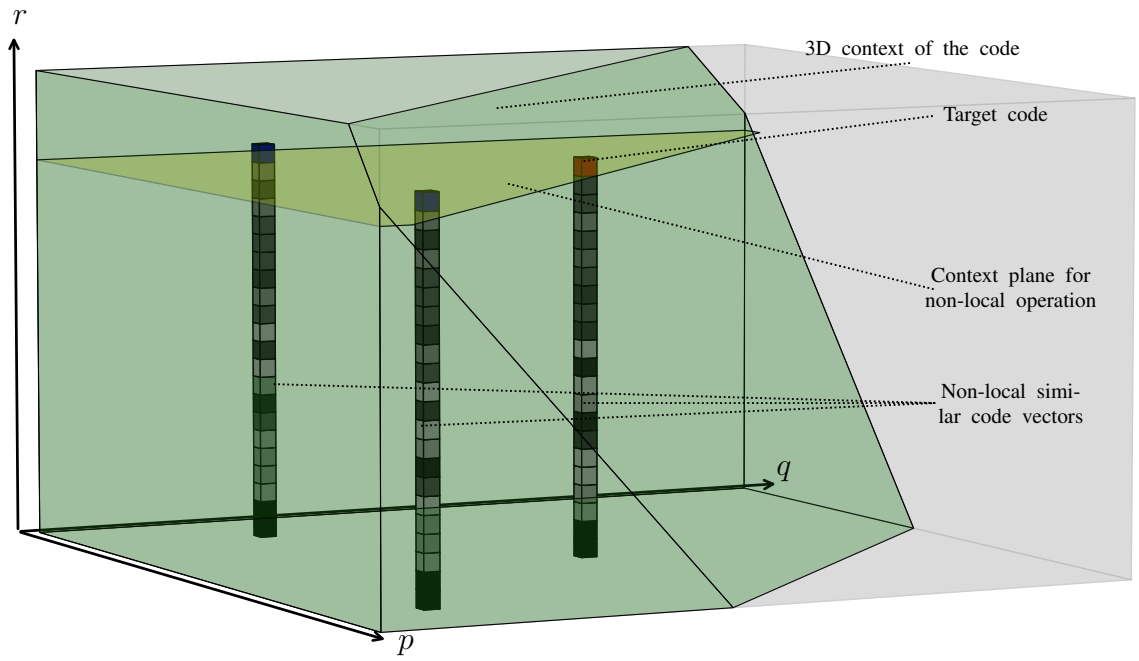


Figure 5.2: Illustration of the non-local similarity of the codes inner the special partial context used in CCNs for entropy modelling. The red block is the target code to be predicted. And the green region is the partial context for the CCNs. Global similar code vectors as the target code vector is located by the proxy similarity metric g_d. And corresponding similar codes (blue blocks) are adopted to predict the target code. The yellow plane indicates the code plane in the context used in the non-local operation.

context and thus could be parallel processed in decoding. Without a clear drop in the entropy prediction performance, the special partial context could dramatically accelerate the decoding efficiency.

The CCNs are built with mask convolution layers and element-wise activation functions. The mask convolution layer is defined as

$$v_{i,r}^{(t)}(p,q) = \sum_{j=1}^{N_t} \sum_{s=1}^{M} \left(u_{j,s}^{(t)} * \left(m_{r,s}^{(t)} \odot w_{i,j,r,s}^{(t)} \right) \right)(p,q) + b_i^{(t)}, \tag{5.2}$$

where $\{i,j\}$ and $\{r,s\}$ are indexes for the feature block and channel dimensions, respectively. $\boldsymbol{u}^{(t)}$ and $\boldsymbol{v}^{(t)}$ are the input and output of the t-th convolution layer, respectively. A nonlinear element-wise activation function is adopted to connect $\boldsymbol{v}^{(t-1)}$ and $\boldsymbol{u}^{(t)}$. For the input layer, the codes to produce $v_{i,r}^{(0)}(p,q)$ is $\Omega_{p,q} = \{y_s(p+u,q+v)\}_{(u,v)\in\Psi,0\leq s<M}$. The

mask for the input layer is defined as

$$m_{r,s}^{(0)}(u,v) = \begin{cases} 1, & \text{if } \Omega_{p,q}(u,v) \in \text{PTX}(y_r(p,q), \boldsymbol{y}) \\ 0, & \text{otherwise.} \end{cases} \tag{5.3}$$

For the t-th hidden layer, the mask is modified to include the codes in the same group

$$m_{r,s}^{(t)}(u,v) = \begin{cases} m_{r,s}^{(0)}(u,v), & \text{if } s+u+v \neq r \\ 1, & \text{otherwise.} \end{cases} \tag{5.4}$$

5.2.2 Context Based Non-local Operation

As the different code planes inner the 3D block are generated by different convolutional filters, only the codes inner the same plane share the same transformation. The non-local operation for the context-based autoregressive task is a weighted averaging on a 2D code plane that the target code belongs to. And the weights are set according to the similarity between the target code and the others. With the definition of the context, only the codes included in the context can be adopted in the non-local operation. A location adaptive mask $m^l(p,q,u,v)$ is introduced to exclude codes outside of the context. Set the target code to be $y_r(p,q)$ in r-th channel at the 2D location (p,q). If another code in the same plane $y_r(u,v) \in \text{PTX}(y_r(p,q), \boldsymbol{y})$, *i.e.*, $v+v < p+q$, the mask $m^l(p,q,u,v)$ is set to be 1; otherwise, it is 0. The yellow plane in Fig. 5.2 indicates the plane of codes could be used to predict the target code in non-local operation where the masks are 1s. Another thing is that the target code to be predicted is unknown in the context. It is unable to directly build a metric to evaluate the content similarity between the target codes and the others. As the codes inner the same 2D position are generated from the same image patch in the receptive field. If two codes $y_j(p,q)$ and $y_j(u,v)$ are similar, it is a high probability event the $y_r(p,q)$ and $y_r(u,v)$ are still similar. We alternatively adopt a similarity metric $g_d(y_r(p,q), y_r(u,v))$ based on the available codes in the context at the same 2D location as (u,v) and (p,q) to evaluate the similarity.

$$g_d(y_r(p,q), y_r(u,v)) = \sum_{j=0}^{r-1} w_{r,j}^d \|y_j(p,q) - y_j(u,v)\|^2, \tag{5.5}$$

where $w_{r,j}$ is a weight to balance the contribution of the codes in different code planes in modelling the similarity metric. $w_{r,j}^d$ is set to be $1/(r+1)$ for even contribution and is

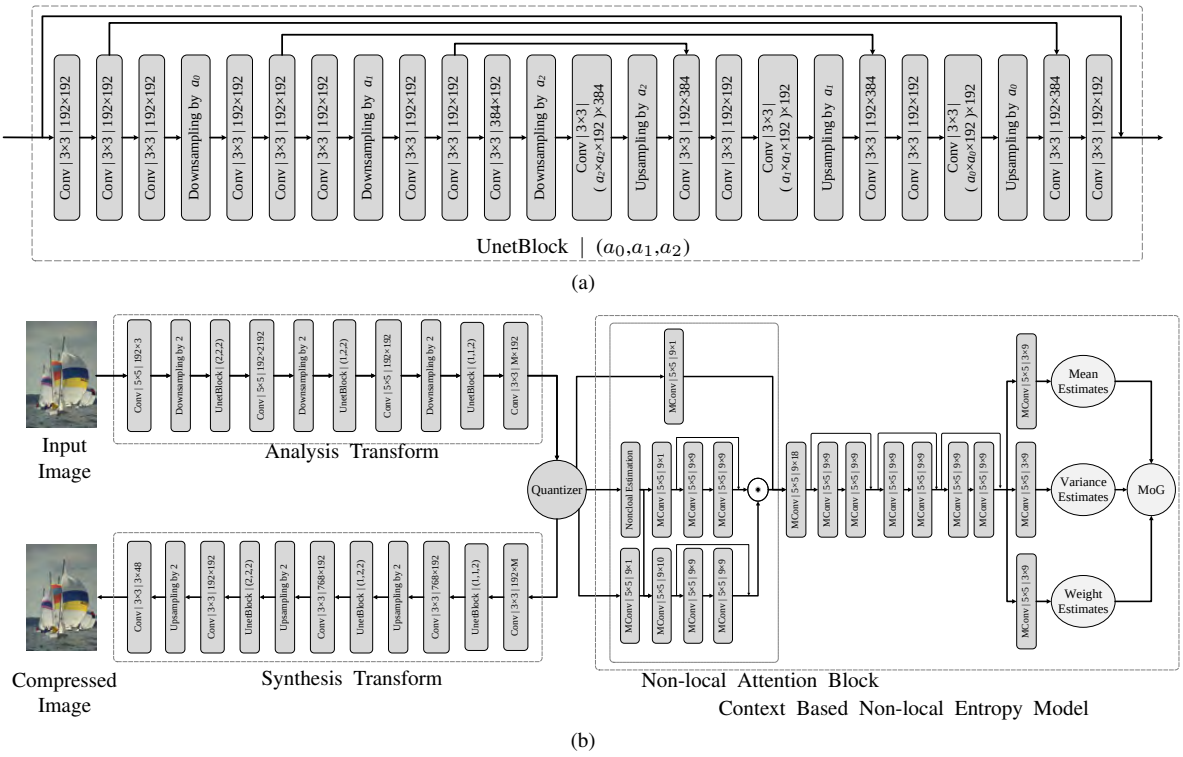


Figure 5.3: The architecture of the proposed lossy image compression method, including an analysis transform g_a, an adaptive trainable quantizer g_q, a context-based non-local entropy model, and a synthesis transform g_s. Conv: regular convolution with filter support (kernel_size $\times$ kernel_size) and number of channels (output$\times$input). UnetBlock: a_0, a_1 and a_2 are the downsampling multipliers in the U-net structure. MConv: mask convolution used in our CCNs with filter support (kernel_size$\times$ kernel_size) and number of feature blocks (output$\times$input). Note that the number of channels is fixed in MConv, and is the same as the input of the entropy model, *i.e.*, $\boldsymbol{y}$.

dynamically optimized in the training of the whole framework. g_d is a proxy similarity metric. With more available codes included in the computation, it will be more accurate in approximate the real similarity metric. The context based non-local operation is defined as,

$$g_{nlc}(y_r(p,q)) = \sum_{u,v} w_r^s(p,q,u,v) y_r(u,v). \tag{5.6}$$

where the content adaptive weight $w_r^s(p,q,u,v)$ is defined as,

$$w_r^s(p,q,u,v) = \frac{m^l(p,q,u,v) e^{-g_d(y_r(p,q),y_r(u,v))}}{\sum_{u',v'} m^l(p,q,u',v') e^{-g_d(y_r(p,q),y_r(u',v'))}} \tag{5.7}$$

Different from traditional non-local operation, the target code itself is absent in computation. There may not exists such content similar codes to estimate the target code. In such case, the non-local estimation will be inaccurate. In order to indicate the confidence that the

target code is estimated by content similar codes, a confidence indicator is introduced as the weighed similarity among the target code and the others.

$$g_c(y_r(p,q)) = \sum_{u,v} w_r^s(p,q,u,v) g_d(y_r(p,q), y_r(u,v)). \tag{5.8}$$

The confidence indicators together and local context are further combined to generate the attention weight for the non-local attention block proposed in this chapter. And an element-wise multiplication operation is adopted for the attention weights and non-local estimation results to produce the attention results, which are then merged with the local representations with a concat operation. Fig. 5.3(b) gives the structure of the non-local attention block .

5.3 Context Based Non-local Entropy Modelling for Lossy Image Compression

To build a DNN-based lossy image compression framework, three key issues to be considered are the structure of the transforms, quantization operation, and the joint rate-distortion objective function. Figure 5.3 gives a whole framework including the synthesis transform, analysis transform, and the context-based non-local entropy model. In this chapter, we will describe the transforms, quantization, and objective function in detail. Finally, a post entropy model is introduced for simplifying the entropy coding process.

5.3.1 Network Structure for Transforms

CNN-based deep transforms are generally not non-invertible, and extra distortion arises in the process of transforms. To get better performance, the extra distortion should be minimized especially for low distortion models. Empirically speaking, staking more different feature maps at each layer of the transform, *i.e.*, increasing the width of the network, will help increase the invertible ability of the transform and reduce extra distortion. Notwithstanding, such a strategy will inevitably increase the time complexity of the transforms and memory consumption. A computationally efficient and wide enough network structure is needed. U-net [39, 59] is a light structure proposed for medical image segmentation. With down-sampling and skip connections, U-net is fast and can adopt multi-scale representation. And the skip connection could facilitate the information propagation and the training of the net-

work [84]. As shown in Fig. 5.3(a), we adopt the U-net structure as a block, *i.e.*, UnetBlock, as a basic block to build the transforms. The UnetBlock could keep the width and reduce the time complexity and memory consumption of the transforms. Each UnetBlock contains 3 downsampling and 3 upsampling convolution layers with the downsampling/upsampling multipliers are determined by the 3 separate parameters, *i.e.*, a_0, a_1 and a_2. And skip connections are introduced to combine the features at each scale. The downsampling convolution layer is conducted by stride convolution, while the upsampling convolution layer is a normal convolution to increase the channels of the feature maps followed by the depth-to-space reshaping [63, 69]. Each convolution layer is followed by a PReLU nonlinearity.

The analysis transform g_a takes the color image $\boldsymbol{x}$ as input and produces the latent code representation $\boldsymbol{z}$, which is further quantized to the discrete code block $\boldsymbol{y}$. g_a is composed of 3 downsampling layer with each followed by a UnetBlock. And an additional convolutional layer with sigmoid nonlinearity is adopted after the last UnetBlock to generate $\boldsymbol{z} \in (0, 1)$ with M channels. And M determines the upper bound of the bit rate of a DNN based compression framework with fixed quantization levels.The parameters of g_a constitute the parameter vector $\boldsymbol{\phi}$ to be optimized.

The synthesis transform g_s is a mirror of the analysis transform. And the upsampling operation is the same as used in UnetBlock. The last convolution layer makes use of three filters to produce the RGB decompressed image. The parameters of g_s constitute the parameter vector $\boldsymbol{\psi}$ to be optimized.

5.3.2 Adaptive Quantization Function

The adaptive quantization function g_q with L quantization levels maps the output of analysis transform $\boldsymbol{z}$ to L discrete quantization centres in each channel and generates the quantized code block $\boldsymbol{y}$. We parametrize the quantization interval for the r-th channel by $\{\sigma_{r,0}, \ldots, \sigma_{r,L-1}\}$. Then, the quantization centres for r-th channel is represented as,

$$\omega_{r,i} = \sum_{j=0}^{i} e^{\sigma_{r,j}}, \text{for } i = 0, \ldots, L-1. \tag{5.9}$$

The output feature $z_r(p,q)$ is assigned to the $\bar{y}_r(p,q)$-th quantization center in r-th channel by minimizing the quantization error

$$\bar{y}_r(p,q) = \arg\min_l \|z_r(p,q) - \omega_{r,l}\|_2^2. \tag{5.10}$$

Then, $y_r(p,q) = g_q(z_r(p,q)) = \omega_{r,\bar{y}_r(p,q)}$ is the quantized code and $\bar{y}_r(p,q)$ is the corresponding integer representation.

g_q has zero gradients almost everywhere, which hinders training via back-propagation. Inspired by the binarized neural networks [17, 55, 86] and the DNN based compression framework [67], an identify mapping $\hat{g}_q(z_r(p,q)) = z_r(p,q)$ is adopted as a continuous proxy for the quantization function in back-propagation.

The quantization intervals $\boldsymbol{\sigma}$ are optimized and modified according to the distribution of $\boldsymbol{z}$ by minimizing the mean squared error (MSE),

$$\mathcal{L}_q(\boldsymbol{\sigma}) = \frac{1}{MHW}\sum_{r,p,q}\|y_r(p,q) - z_r(p,q))\|_2^2, \tag{5.11}$$

Specifically, we initialize $\boldsymbol{\sigma}$ using uniform quantization, which appears to work well in all experiments.

5.3.3 Modelling the Objective Function

The whole framework is optimized with a joint rate-distortion objective function. The distortion loss is directly modeled on the decompressed image $\hat{\boldsymbol{x}}$ and the original image $\boldsymbol{x}$. In this chapter two separate metrics, *i.e.*, standard mean square error and the perceptual metric MS-SSIM [77], are adopted as the distortion loss. The MSE distortion loss $\mathcal{L}_d^{\text{MSE}}$ and MS-SSIM distortion loss $\mathcal{L}_d^{\text{MS-SSIM}}$ are defined as follows

$$\mathcal{L}_d^{\text{MSE}}(\hat{\boldsymbol{x}}, \boldsymbol{x}) = \frac{1}{3H_I W_I}\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_2^2, \tag{5.12}$$

and

$$\mathcal{L}_d^{\text{MS-SSIM}}(\hat{\boldsymbol{x}}, \boldsymbol{x}) = 100\big(1 - \text{MS-SSIM}(\hat{\boldsymbol{x}}, \boldsymbol{x}\big). \tag{5.13}$$

Here, H_I and W_I are separately the height and width of the image $\boldsymbol{x}$. We denote our method with $\mathcal{L}_d^{\text{MSE}}$ as Ours(MSE) and with $\mathcal{L}_d^{\text{MS-SSIM}}$ as Ours(MS-SSIM).

Without prior knowledge of the PDFs of the quantized codes $\boldsymbol{y}$, we approximate them with discretized MoG distributions, whose parameters are predicted by the proposed context-based non-local entropy model. The non-local entropy model consists of a non-local attention block, several CCN residue blocks and three final CCN layers to separately produce the mean estimates, variance estimates and weight estimates for the MoG distributions. The network parameters of our context-based non-local entropy model constitute the parameter vector $\boldsymbol{\theta}$ to be optimized.

Fig. 5.3(b) shows the network structure of the proposed context-based non-local entropy model. We write the differentiable MoG distribution with C components as

$$y_r(p,q) \sim \sum_{i=0}^{C-1} \pi_i \mathcal{N}(y_r(p,q); \mu_i, \sigma_i^2), \tag{5.14}$$

where π_i, μ_i and δ_i^2 are the mixture weight, mean, and variance of the i-th component, respectively. Then, we get the discrete probability by calculating the integral on a quantization bin

$$P(y_r(p,q);\theta) = \int_{\Delta} \sum_{i=0}^{C-1} \pi_i \mathcal{N}(\xi; \mu_i, \sigma_i^2) d\xi. \tag{5.15}$$

where $\Delta = [(\omega_{r,\bar{y}_r(p,q)-1} + y_r(p,q))/2, (\omega_{r,\bar{y}_r(p,q)+1} + y_r(p,q))/2]$ is the quantization bin that $y_r(p,q)$ lies in. $0 \leq \bar{y}_r(p,q) < L$ indexes the quantization centers $\boldsymbol{\omega}$. For $\bar{y}_r(p,q) = 0$ and $\bar{y}_r(p,q) = L-1$, we separately set the nonexistent centers as $\omega_{r,\bar{y}_r(p,q)-1} = 0$ and $\omega_{r,\bar{y}_r(p,q)+1} = 1$ according to $y_r(p,q) \in (0,1)$.

Finally, we are able to write the empirical rate-distortion objective for the parameters $\{\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\theta}\}$ as

$$\begin{aligned}\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}} \Bigg[& - \sum_i \log_2 P\Big(g_q\Big(g_a(\boldsymbol{x};\boldsymbol{\phi})\Big);\boldsymbol{\theta}\Big) \\ & + \lambda \mathcal{L}_d\Big(g_s\Big(g_q\big(g_a(\boldsymbol{x};\boldsymbol{\phi})\big);\boldsymbol{\psi}\Big), \boldsymbol{x}\Big)\Bigg].\end{aligned} \tag{5.16}$$

$\mathcal{L}_d$ is the distortion term, which is more preferable to be assessed in a perceptual space. In this chapter, we optimize and evaluate our lossy image compression methods using standard MSE and a perceptual metric MS-SSIM [77].

5.3.4 Post Processing for Entropy Coding

For entropy coding, we choose arithmetic coding to compress the codes of the image into a bitstream for saving. For arithmetic coding, the integer code and a corresponding probability table for each possible discrete value of the code are needed. With the context-based non-local entropy model, we should first estimate the parameter for MoG distributions and then cut the PDF into L intervals and get the discrete possibility table with an integral on each interval. Considering a large amount of code to be processed, it will bring further computational burden and slow down the entropy coding process. We simplify this process by directly training a post entropy model which takes the integer representation of the codes, *i.e.*, the index of the quantization centers $\bar{\boldsymbol{y}}$, as input and directly output the discrete probability table for each integer representation. For the post entropy model, it adopts a similar structure as the context-based non-local entropy model, and the only difference is the last CCN layer. Instead of taking three separate CCN layer to produce the mean, variance and weight used in MoGs, it adopt only one CCN layer followed by a softmax nonlinearity to produce the possibility table $\boldsymbol{u}$ with $u_{i,r}(p,q)$ represents the possibility that $\bar{y}_r(p,q) = i$. The post entropy network parameters are optimized by minimizing the expected code length

$$\mathcal{L}_{post}(\boldsymbol{\theta}) = -\mathbb{E}_{\bar{\boldsymbol{y}}}\Big[\sum_{r,p,q}\sum_{i}\mathbb{1}(\bar{y}_r(p,q)=i)\log_2(u_{i,r}(p,q))\Big], \tag{5.17}$$

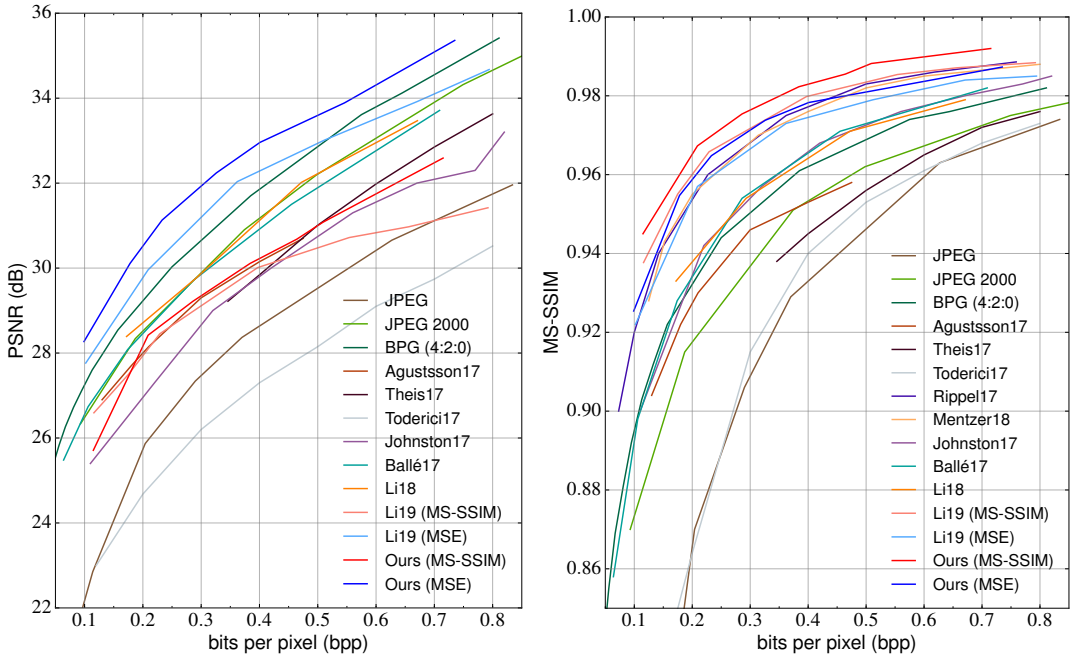
where $\mathbb{1}(\cdot)$ is an indicator function and the expectation may be approximated by averaging over a mini-batch of integer representations of the code blocks. Finally, we implement our own arithmetic coding with the context-based non-local entropy model to compress $\bar{\boldsymbol{y}}$ to bitstreams, and report performance using actual bit rates.

5.4 Experiments

In this section, we test the proposed context-based non-local entropy models in the lossy image compression by comparing it to state-of-the-art image coding standards and recent deep image compression algorithms. $10,000$ high-quality and high-definition images are collected from Flickr, and down-sampling operations are conducted on them to further reduce possibly compression artifacts. We crop $640,000$ color patches of size $3 \times 256 \times 256$ as the training sets for lossy image compression. For the post entropy model, we first extract the integer code block, *i.e.*, $\bar{\boldsymbol{y}}$, and then crop code patches of size $M \times 60 \times 60$ for training. We test our models on two independent datasets - Kodak and Tecnick [9], which are widely used to benchmark image compression performance. The pre-trained models for testing are made available at https://github.com/limuhit/Nonlocal-CCN.

5.4.1 Experimental Setup

We jointly optimize the analysis transform g_a, the non-uniform quantizer g_q, the context-based non-local entropy model, and the synthesis transform g_s by minimizing the rate-distortion objective function in an end-to-end manner. And the same warmup strategy used in [37] is used for initialization. Then, an Adam solver with a learning rate of 10^{-5} is adopted to optimize the whole model. Smaller learning rates, *i.e.*, 10^{-6} and 10^{-7}, are adopted until the objective function does not decrease for 5 successive epochs. The post entropy model is also optimized with Adam solver in the same way. The number of quantization centers is $L = 8$ and the number of Gaussian components in MoG is $C = 3$. This optimization is performed separately for each λ and each distortion measure. We train 14 model models for seven bit rates and two distortion metrics (MSE and MS-SSIM). For testing, the compression rate is evaluated by bits per pixel (bpp), which is the total amount of bits used to compress the image divided by the whole number of pixels in the image. Two quantitative metrics, *i.e.*, Multi-Scale Structural Similarity (MS-SSIM) and the Peak Signal-to-Noise Ratio (PSNR), are used to evaluate the image distortion.



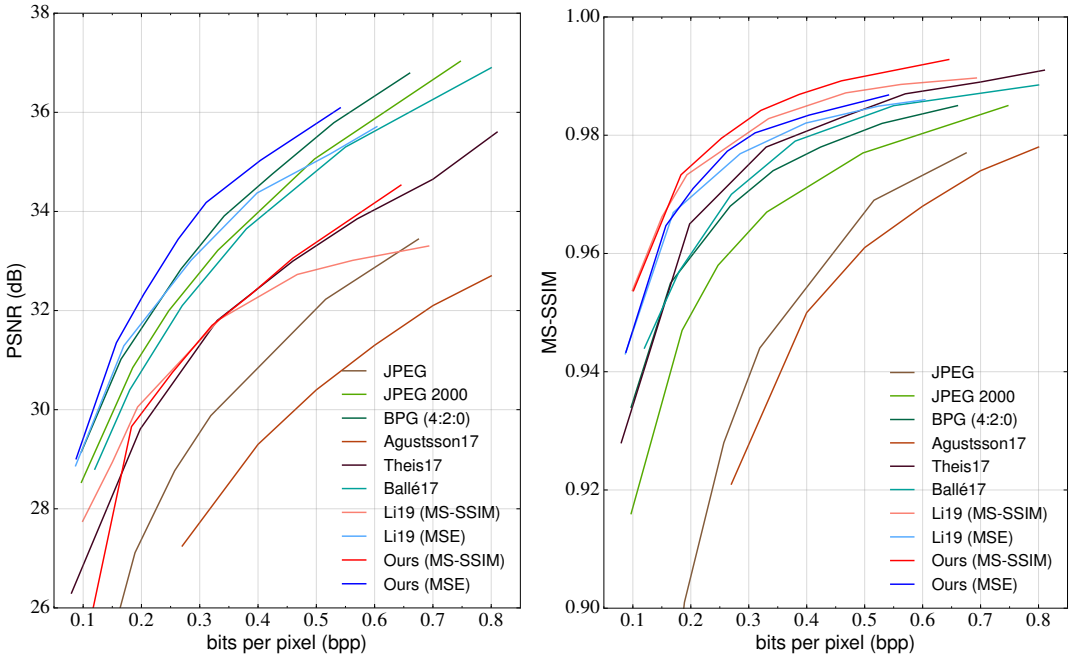
(a) (b)

Figure 5.4: Rate-distortion curves of different compression methods on the Kodak dataset. (a) PSNR. (b) MS-SSIM.

5.4.2 Quantitative Evaluation

Using MS-SSIM and PSNR as distortion metrics, we compare our methods with existing image coding standards and recent DNN-based compression models in term of rate-distortion curves. Image coding standards include JPEG [73], JPEG2000 [64], BPG [13], and the DNN-based compression models include Agustsson17 [2], Theis17 [67], Toderici17 [69], Rippel17 [58], Mentzer18 [46], Johnston17 [32], Ballé17 [11], our results in chapter 2 (Li18) and chapter 4 (Li19). Both JPEG (with 4:2:0 chroma subsampling) and JPEG2000 are based on the optimized implementations in MATLAB2017. For BPG, we adopt the latest version from its official website with the default setting. When it comes to DNN-based lossy image compression models, the implementations are generally not available. Therefore, we carefully digitalize the rate-distortion curves and report the results from their respective chapters.

Fig. 5.4 shows the rate-distortion curves on the Kodak dataset. In terms of PSNR, the results of Rippel17 [58] and Mentzer18 [46] are missing due to that they do not report the PSNR curve in the chapter. We find that both of our methods optimized for MSE and MS-SSIM outperform all competing methods. Especially, compared with chapter 4 with local



(a) (b)

Figure 5.5: Rate-distortion curves of different compression methods on the Tecnick dataset. (a) PSNR. (b) MS-SSIM.

context-based entropy modeling, our models with context-based non-local entropy modeling are much better, which support the effectiveness of adopting context-based non-local estimation in entropy modeling. And in the high bit rate region, our models outperform Li19 by a lot, which also indicates the effectiveness of the proposed UnetBlock in reducing extra distortion brought by transforms. Fig. 5.5 shows the rate-distortion curves on the Tecnick dataset, where similar trends as Koadak dataset for both PSNR and MS-SSIM can be observed. We fail to compare with Rippel17 [58], Theis17 [67], Agustsson17 [2] and Mentzer18 [46] on Tecnick dataset due to the unavailable results in the chapters.

5.4.3 Visual Quality Evaluation

We visually compare the decompressed images by our method against results from chapter 4, JPEG2K, and BPG. Fig. 5.6 and Fig. 5.7 show sample decompressed images and the uncompressed images on the Kodak and Tecnick datasets, respectively. For images at low bit rate, the methods optimized with MS-SSIM are visually much better due to that MS-SSIM takes structural similarity in different scale into account and are more consistent with the

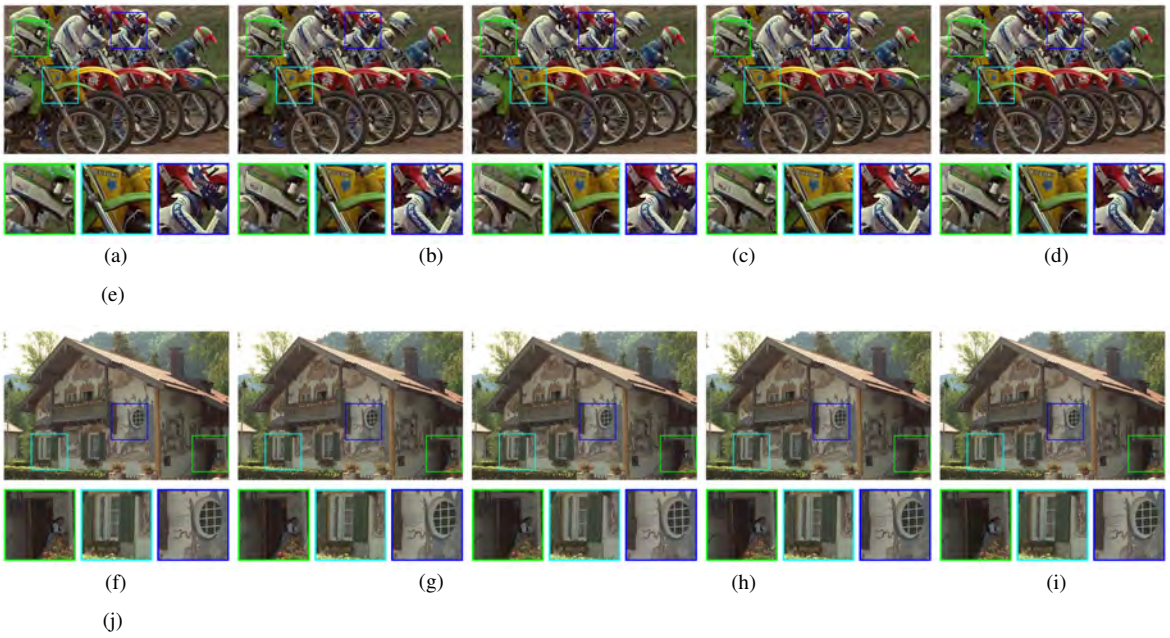


Figure 5.6: Compressed images by different compression methods on the Kodak dataset. The quantitative measures are in the format of "bpp / PSNR / MS-SSIM". (a) Uncompressed "Motorcycle" image. (b) JPEG2K. 0.673 / 27.75 / 0.962. (c) BPG. 0.712 / 29.65 / 0.973. (d) Chapter 4 optimized for MSE. 0.694 / 28.81 / 0.986. (e) Ours optimized for MSE. 0.670 / 30.53 / 0.984. (f) Uncompressed "House" image. (g) JPEG2K. 0.879 / 31.52 / 0.972. (h) BPG. 0.877 / 32.64 / 0.979. (i) Chapter 4 optimized for MSE. 0.871 / 29.99 / 0.988. (j) Ours optimized for MSE. 0.865 / 33.04 / 0.989.

human visual system. When it comes to the images at a high bit rate, the methods optimized with MSE are better at keeping small scale details. Thus, we compare Ours(MS-SSIM) with Li19(MS-SSIM) in low bit rate as shown in Fig. 5.7 and compare Ours(MSE) with Li19(MSE) at high bit rate in Fig 5.6. In Fig. 5.7, JPEG2K and BPG exhibit artifacts (such as blocking, ringing, blurring, and aliasing) that are common to all handcrafted transform coding methods. Li19(MS-SSIM) is effective at suppressing most of the artifacts bus still surfers from blurring in some parts of the image. In contrast, our method optimized for MS-SSIM is more able to generate decompressed images with more faithful details and less visible distortions. In Fig. 5.4, the whole visual quality is nearly the same. But when zooming into details, similar to BPG, our methods optimized for MSE shows to have better small scale edges and textures. Li19(MSE) blurring the small edges and losing useful information, such as small text.

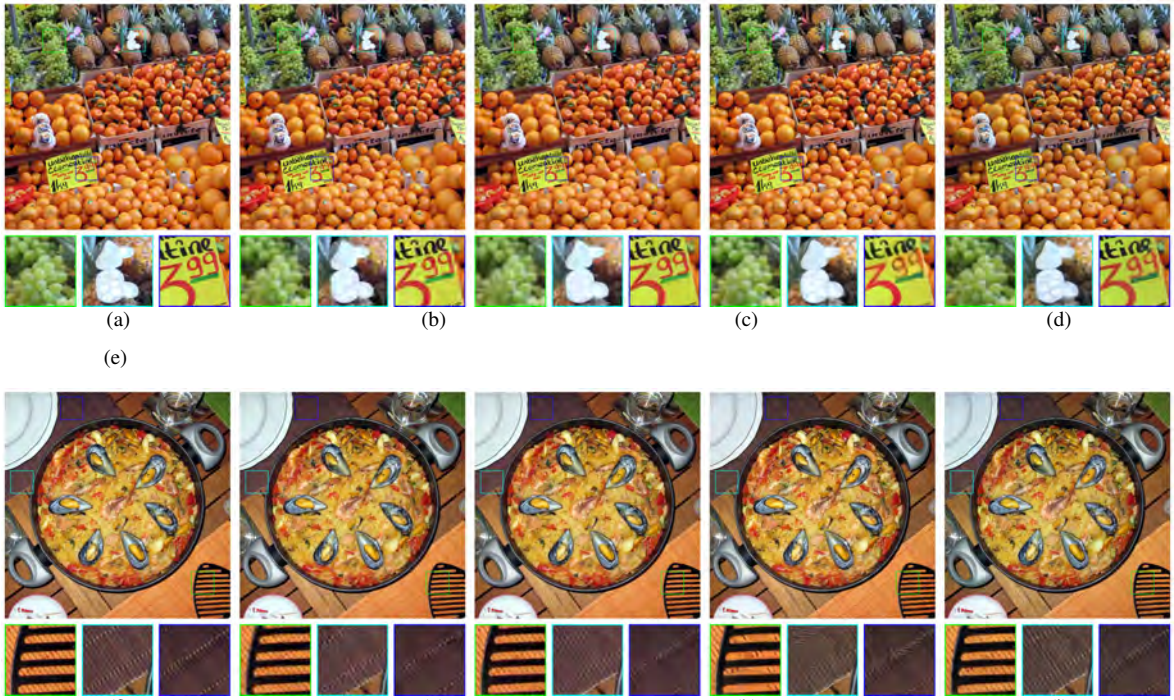


Figure 5.7: Compressed images by different compression methods on the Tecnick dataset. The quantitative measures are in the format of "bpp / PSNR / MS-SSIM". (a) Uncompressed "Fruit" image. (b) JPEG2K. 0.137 / 24.95 / 0.890. (c) BPG. 0.136 / 25.58 / 0.901. (d) Chapter 4 optimized for MS-SSIM. 0.139 / 25.63 / 0.938. (e) Ours optimized for MS-SSIM. 0.134 / 24.46 / 0.939. (f) Uncompressed "Seafood" image. (g) JPEG2K. 0.137 / 25.52 / 0.882. (h) BPG. 0.135 / 26.42 / 0.905. (i) Chapter 4 optimized for MS-SSIM. 0.138 / 25.62 / 0.925. (j) Ours optimized for MS-SSIM. 0.133 / 25.03 / 0.932.

Table 5.1: Running time in seconds, GPU memory usage in GBs and distortion evaluated with PSNR of three network structures.

Network Structure	Running Time (seconds)	GPU Memory (GB)	Distortion (dB)
DenseBlock [37]	0.025	0.86	35.54
ResidueBlock	0.136	2.65	36.34
UnetBlock	0.048	1.52	36.29

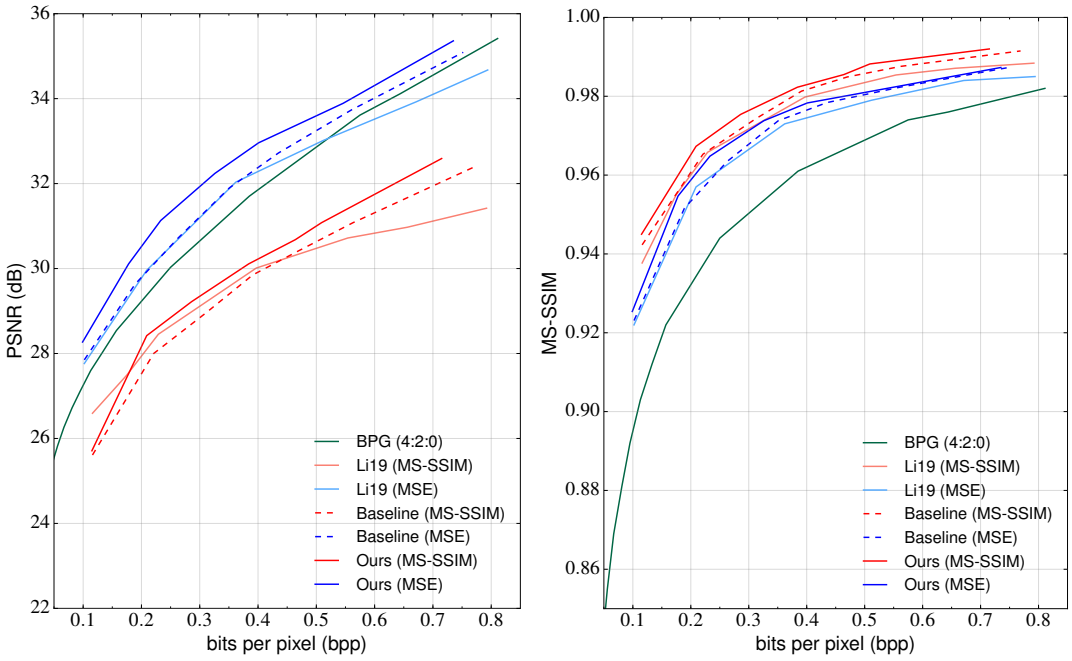
5.4.4 Ablation Experiments

We conduct thorough ablation experiments to analyze the impact of individual components, *i.e.*, the UnetBlock, and context-based non-local entropy modeling, to final compression performance. For fair comparisons, we use the same set of parameters and training set for all the competing models.

UnetBlock for analysis and synthesis transforms

The width of the transforms, *i.e.*, the minimum number of channels in the output of each layer, is supposed to have a significant influence on the performance of low distortion image compression models. A narrow transform will inevitably lose necessary information for reconstructing the decoding image and introduce extra distortion. To support this hypothesis, we introduce a baseline model that adopts the entropy model from Chapter 4 and transforms with UnetBlocks. The only difference between Chapter 4 and the baseline model is the network structure of the transforms. Our baseline model adopts the analysis and synthesis transforms with a width of 192. But for transforms in Li19, the width is 64. As the DenseBlocks of Li19 concat all the output of each previous sub-blocks as the input for a new sub-block, the computational consumption increase a lot with the growth of the width of the input. In practice, limited by the hardware, *i.e.*, GPU memory, it is unable to adopt a wide input, *i.e.*, 192, with a lot of feature maps for the DenseBlocks in Li19.

Fig. 5.8 shows the performance of the baseline models. We compare the baseline models with Chapter 4 separately optimized with two distortion metrics, *i.e.*MS-SSIM and MSE. In low bit rate region (< 0.4bpp), the baseline model optimized with MSE, *i.e.*, Baseline(MSE), and Li19(MSE) have nearly the same performance in both of the MS-SSIM and



(a) (b)

Figure 5.8: Rate-distortion curves of different variants of the proposed methods on the Kodak dataset. (a) PSNR. (b) MS-SSIM. Baseline denotes our method with unet blocks and normal entropy mdoel.

Table 5.2: Entropy coding for integer codes optimized by MSE and MS-SSIM. The results are evaluted by bits per code. $\bar{\boldsymbol{y}}^{a,i}$ ($\bar{\boldsymbol{y}}^{b,i}$) represents the interger code of the i-th model optimized for the MSE(MS-SSIM).

Entropy coding for integer codes optimized by MSE							
Code set	$\bar{\boldsymbol{y}}^{a,0}$	$\bar{\boldsymbol{y}}^{a,1}$	$\bar{\boldsymbol{y}}^{a,2}$	$\bar{\boldsymbol{y}}^{a,3}$	$\bar{\boldsymbol{y}}^{a,4}$	$\bar{\boldsymbol{y}}^{a,5}$	$\bar{\boldsymbol{y}}^{a,6}$
CCN [37]	1.65	1.56	1.45	1.45	1.43	1.46	1.58
Non-local	1.58	1.42	1.24	1.30	1.28	1.34	1.47
Entropy coding for integer codes optimized by MS-SSIM							
Code set	$\bar{\boldsymbol{y}}^{b,0}$	$\bar{\boldsymbol{y}}^{b,1}$	$\bar{\boldsymbol{y}}^{b,2}$	$\bar{\boldsymbol{y}}^{b,3}$	$\bar{\boldsymbol{y}}^{b,4}$	$\bar{\boldsymbol{y}}^{b,5}$	$\bar{\boldsymbol{y}}^{b,6}$
CCN [37]	1.86	1.78	1.66	1.60	1.64	1.39	1.55
Non-local	1.84	1.67	1.53	1.54	1.48	1.26	1.43

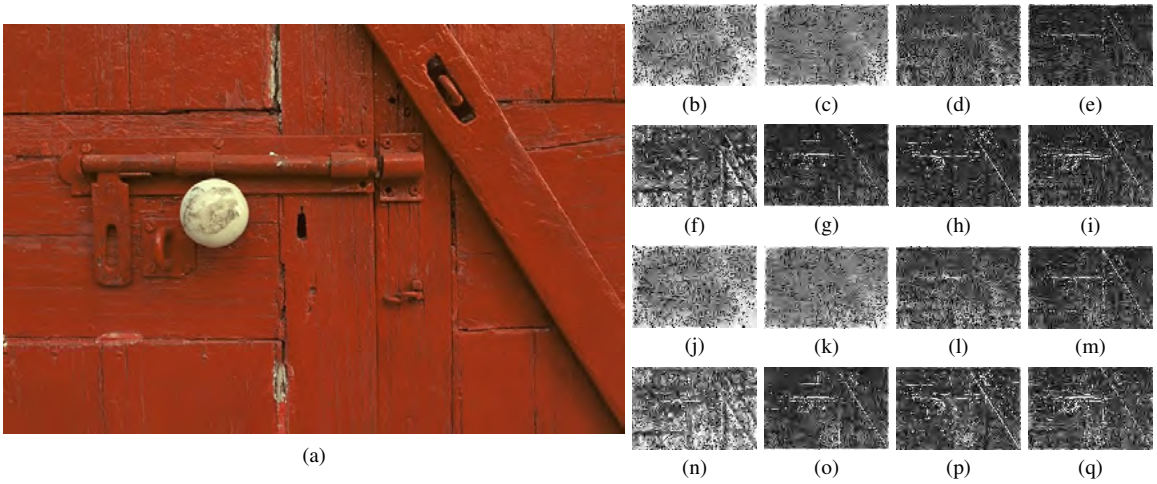


Figure 5.9: Visualization of the estimated probability for the context-based non-local entropy modelling and the CCN-based entropy modelling. Each gray figure is a visualization of one code plane. The bright/dark color represents large/small estimated discrete probability (small/large entropy) of the corresponding codes. The quantitative measures are in the format of channel number of the code plane / bits per code. For simplification, "NLC" represents the context-based non-local entropy modelling and "LC" represents the CCN-based entropy modelling. (a) Uncompressed "Door" image. (b) LC. 1 / 1.05. (c) LC. 7 / 1.16. (d) LC. 16 / 1.85. (e) LC. 27 / 2.40. (f) LC. 28 / 1.67. (g) LC. 29 / 2.37. (h) LC. 30 / 2.37. (i) LC. 31 / 2.30. (j) NLC. 1 / 1.05. (k) NLC. 7 / 1.14. (l) NLC. 16 / 1.75. (m) NLC. 27 / 2.11. (n) NLC. 28 / 1.21. (o) NLC. 29 / 2.03. (p) NLC. 30 / 1.97. (q) NLC. 31 / 1.94.

MSE. But when the bit rate grows, the baseline model optimized with MSE has a significant improvement, which supports the hypothesis about the width of the transforms. Similar trends can be observed for our model optimized with MS-SSIM.

To support the effectiveness and efficiency of the proposed UnetBlock, we compare the UnetBlock based network structure, including analysis transform, adaptive quantization and synthesis transform, to a variant structure with only residue blocks in the transforms. The variant network adopts several residue blocks to replace each UnetBlock. And the width and the total number of layers in the variant structure is nearly the same as the UnetBlock based network structure. Without the limitation of the entropy model ($\lambda = \infty$), we directly optimize the UnetBlock based network, the residue block based network and the DenseBlock based network [37] to the MSE distortion loss. Table 5.1 gives the results of the three structures on running time, GPU memory usage and distortion performance. As the ResidueBlock based network can not directly process the whole size image from the Kodak dataset, the running time in seconds, the GPU memory evaluated by GB and the distortion

performance evaluated by PSNR are given on 256×256 patches sampled from the Kodak dataset. As shown in the table, the UnetBlock base network performs on par with ResidueBlock based network on distortion performance and overwhelm the narrow network, *i.e.*, DenseBlock based network. When it comes to efficiency, the UnetBlock based network is much faster and needs less GPU memory in processing the images than the ResidueBlock based network. Thus, UnetBlock is a good trade-off between efficiency and effectiveness.

Context based non-local entropy modelling

The context-base non-local entropy modeling exploits both of the non-local similarity among the context and the local representations for the auto-regressive entropy modeling. We compare it with the entropy model only adopts local representations use in Chapter 4. The same baseline models as described above are used for comparison. Compared with our models used in this chapter, the baseline models adopt the CCN-based entropy model focusing on local representations. As shown in Fig. 5.8, both of our model optimized for MSE and our model optimized for MS-SSIM outperform the counterpart baseline models by a large margin, which strongly support the effectiveness of the proposed non-local attention block in context modeling.

To move the effectiveness of different initialization parameters and the training process. We test the performance of the context-based non-local entropy modeling and the CCN-based entropy modeling on the same set of integer codes generated by our models. Seven code sets, *i.e.*, $\bar{\boldsymbol{y}}^{a,0}, \ldots, \bar{\boldsymbol{y}}^{a,6}$, are separately generated by Ours(MSE) at seven bit rate. And another seven code sets denoted by $\bar{\boldsymbol{y}}^{b,0}, \ldots, \bar{\boldsymbol{y}}^{b,6}$ are produced by Ours(MS-SSIM). The performance of the two entropy models is evaluated as the number of bits used to coding one code (bits per code). Table 5.2 shows the performance of the two entropy model, on the 14 code set, the context-based non-local entropy model is significantly better than the CCN-based entropy model, which further supports the contribution of the introduced non-local attention block in entropy modeling.

Besides, we further visualize the estimated probability for each code plane by mapping $P(y_r(p,q))$ to an integer in the range of $[0, 255]$ and shows each code plane as a gray

image in Fig. 5.9. Fig. 5.9 (a) gives the uncompressed image; (b)-(i) are the visualized probability of the CCN-based entropy model on 8 code planes; (j)-(q) are the corresponding results of the context-based non-local entropy model. In bottom code planes, as shown in Fig. 5.9 (b), (c), (j) and (k), CCN-based entropy model has similar performance as the proposed non-local entropy model. This can be illustrated by the proxy similarity function. The similarity of codes inner one plane is estimated by all the bottom code planes. With few bottom code planes available for estimation, the similarity is not accurate and thus lead to poor non-local estimation. The non-local attention block tends to focus on the local feature representations instead of the non-local estimation. As a result, the performance of the two competing entropy models on the bottom code maps is similar. However, the context-based non-local entropy modelling shows to have overwhelming performance in the top code plane as shown in Fig. 5.9 (f), (g), (h), (i), (n), (o), (p) and (q). With the increasing number of planes in the bottom, the proxy similarity begins more accurate and the non-local estimations star to contribute to the performance.

5.5 Conclusion

In this chapter, we model the non-local similarity in the context and introduce a non-local attention block to combine the context-based non-local estimation and local auto-regressive representations for context-based entropy modeling. And an effective and efficient network structure, *i.e.*, UnetBlock, is introduced to build the analysis transform and synthesis transform to help reduce the extra distortion brought by the non-invertible transforms. We test the context-based non-local entropy model and the UnetBlock for lossy image compression. Our model gets significant improvements over image compression standards and recent DNN-based models. s

CHAPTER 6
CONCLUSION AND FUTURE WORK

6.1 Summary

As a very basic problem of computer science, image compression methods are studied for decades. With the recent success of deep learning and the perfect match between the convolutional networks and the transform coding framework, we proposed several deep network-based image compression in this thesis for better image compression performance.

In chapter 2, we proposed a content weighed image compression framework to allocate more bits to code the content important regions in the images and few bits to code the less important regions with a learned importance map optimized to rate-distortion loss. The sum of the importance map is adopted as a proxy of the rate loss. Finally, we extract a small block around the code as its context and adopt a simple CNN for post entropy coding.

In chapter 3, we extend the job in chapter 2 with better deep network-based transforms, better quantization functions, better optimization strategies for the importance map. Furthermore, a trimmed convolutional network is introduced for parallel entropy prediction without extract context for each code. Finally, we apply the content weighted image compression framework for task-driven image compression by introducing region of interest of specific tasks into the learning of importance map.

In chapter 4, we focus on entropy modeling of the deep image compression framework and proposed a general entropy modeling framework with mask convolution operations. And a special code diving scheme together with its corresponding context is introduced for the efficiency of entropy modeling without a clear drop in the performance. The entropy modeling is further applied for lossless image compression and guiding the learning of lossy image compression methods.

In chapter 5, we introduce the non-local similarity inner the codes generated by deep networks for context-based entropy modeling. A non-local attention block that combines the local auto-regressive representations and the global context based non-local estimation. As the network-based transforms are inevitable and have internal distortion, a better network structure with smaller distortion is also introduced for better performance. With the improvements, we achieve state-of-art image compression performance.

6.2 Future Work

In future work, we plan to investigate the following three problems:

- **Efficient deep image compression framework.** Currently, the deep image compression framework is computational inefficiency which can not be applied on personal devices without high-performance graphic cards. In future work, we would design some light deep image compression framework with high efficiency.

- **Task driven image compression.** We have implied the content weighed image compression framework for task-driven image compression. However, under the current framework, we should decode the images first and then conduct a specific task on the decoded image, which is not efficient. Also, we only adopt the ROI region in chapter 3 instead of task performance. A joint task-driven framework for the performance of specific tasks instead of the ROI region would be better. And we will try to explore to do the tasks on the codes of the images instead of the decoded image.

- **Transparent image compression.** The deep network-based transforms are inevitable, which makes the deep network-based transform coding scheme can not be applied for lossless or nearly lossless image compression. In the future, we would try to explore this problem and design better networks for the transparent image compression task.

Bibliography

[1] M. D. Adams and R. Ward. Wavelet transforms in the JPEG-2000 standard. In *2001 IEEE Pacific Rim Conf. Commun., Comput. Signal Process.*, volume 1, pages 160–163 vol.1, Aug 2001.

[2] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Neural Inf. Process. Syst*, pages 1141–1151, 2017.

[3] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *IEEE Conf. Comput. Vis. Pattern Recog. Workshops*, volume 3, 2017.

[4] Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool. Extreme learned image compression with GANs. In *IEEE Conf. Comput. Vis. Pattern Recog. Workshops*, pages 2587–2590, 2018.

[5] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Trans. Comput.*, C-23(1):90–93, 1974.

[6] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE Trans. Comput.*, 100(1):90–93, 1974.

[7] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, (12):2037–2041, 2006.

[8] Marc Antonini, Michel Barlaud, Pierre Mathieu, and Ingrid Daubechies. Image coding using wavelet transform. *IEEE Trans. Image Process.*, 1(2):205–220, 1992.

[9] Nicola Asuni and Andrea Giachetti. TESTIMAGES: A large data archive for display and algorithm testing. *Journal of Graphics Tools*, 17(4):113–125, 2013.

[10] Nicola Asuni and Andrea Giachetti. Testimages: a large-scale archive for testing visual devices and basic image processing algorithms. In *STAG - Smart Tools & Apps Graph. Conf.*, 2014.

[11] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. In *Int. Conf. Learning Representations*, 2017.

[12] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *Int. Conf. Learning Representations*, 2018.

[13] Fabrice Bellard. BPG image format. https://bellard.org/bpg/, 2019.

[14] Vicki Bruce and Andy Young. Understanding face recognition. *British journal of psychology*, 77(3):305–327, 1986.

[15] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Neural Inf. Process. Syst*, pages 6571–6583, 2018.

[16] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conf. Empirical Methods Natural Language Process. (EMNLP 2014)*, 2014.

[17] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv:1602.02830*, 2016.

[18] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, New York, NY, USA, 2006.

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 248–255, 2009.

[20] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *Eur. Conf. Comput. Vis.*, pages 184–199. 2014.

[21] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(2):295–307, 2016.

[22] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645, 2009.

[23] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Int. Conf. Comput. Vis.*, pages 580–587, 2014.

[24] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Neural Inf. Process. Syst*, pages 2672–2680, 2014.

[25] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE Int Conf. Acoustics, Speech Signal Process.*, pages 6645–6649. IEEE, 2013.

[26] Robert M. Gray and David L. Neuhoff. Quantization. *IEEE Trans. Inform. Theory*, 44(6):2325–2383, 1998.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 770–778, 2016.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 770–778, 2016.

[29] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Process. Mag.*, 29, 2012.

[30] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, volume 1, pages 4700–4708, 2017.

[31] David A Huffman. A method for the construction of minimum-redundancy codes. *Proc. IRE*, 40(9):1098–1101, 1952.

[32] 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 *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4385–4393, 2018.

[33] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Int. Conf. Learning Representations*, 2015.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Neural Inf. Process. Syst*, pages 1097–1105, 2012.

[35] Jani Lainema, Frank Bossen, Woo-Jin Han, Junghye Min, and Kemal Ugur. Intra coding of the hevc standard. *IEEE Trans. Circuits Syst. Video Technol.*, 22(12):1792–1801, 2012.

[36] Mu Li, Shuhang Gu, David Zhang, and Wangmeng Zuo. Efficient trimmed convolutional arithmetic encoding for lossless image compression. *arXiv:1801.04662*, 2018.

[37] Mu Li, Kede Ma, Jane You, David Zhang, and Wangmeng Zuo. Efficient and effective context-based convolutional entropy modeling for image compression. *arXiv preprint arXiv:1906.10057*, 2019.

[38] Mu Li, Wangmeng Zuo, Shuhang Gu, Debin Zhao, and David Zhang. Learning convolutional networks for content-weighted image compression. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3214–3223, 2018.

[39] Xiaomeng Li, Hao Chen, Xiaojuan Qi, Qi Dou, Chi-Wing Fu, and Pheng-Ann Heng. H-DenseUNet: hybrid densely connected UNet for liver and tumor segmentation from CT volumes. *IEEE Trans. Medical Imaging*, 37(12):2663–2674, 2018.

[40] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *IEEE Conf. Comput. Vis. Pattern Recog. Workshops*, 2017.

[41] Stuart Lloyd. Least squares quantization in PCM. *IEEE Trans. Inform. Theory,*, 28(2):129–137, 1982.

[42] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention visual question answering. In *Neural Inf. Process. Syst*, pages 289–297, 2016.

[43] K. Ma, W. Liu, K. Zhang, Z. Duanmu, Z. Wang, and W. Zuo. End-to-end blind image quality assessment using deep neural networks. *IEEE Trans. Image Process.*, 27(3):1202–1213, 2018.

[44] Detlev Marpe, Heiko Schwarz, and Thomas Wiegand. Context-based adaptive binary arithmetic coding in the H. 264/AVC video compression standard. *IEEE Trans. Circuits Syst. Video Technol.*, 13(7):620–636, 2003.

[45] G.N.N. Martin. Range encoding: An algorithm for removing redundancy from a digitised message. In *Video & Data Recording Conf.*, pages 24–27, 1979.

[46] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4394–4402, 2018.

[47] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Int. Speech Commun. Assoc.*, 2010.

[48] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Neural Inf. Process. Syst*, pages 10794–10803. 2018.

[49] Gordon E Moore et al. Cramming more components onto integrated circuits, 1965.

[50] Boris Motik, Sergio Antonio Berná Niñerola, Pablo Castellanos Garcia, and Carlos González-Cadenas. Natural language question answering system and method based on deep semantics, 12 2011. US Patent App. 13/171,391.

[51] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Int. Conf. Machine Learning*, pages 1747–1756, 2016.

[52] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with PixelCNN decoders. In *Neural Inf. Process. Syst*, pages 4797–4805, 2016.

[53] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *British Machine Vis. Conf.*, 2015.

[54] Javier Portilla, Vasily Strela, Martin J Wainwright, and Eero P Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Trans. Image Process.*, 12(11), 2003.

[55] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Eur. Conf. Comput. Vis.*, pages 525–542, 2016.

[56] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 779–788, 2016.

[57] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Neural Inf. Process. Syst*, pages 91–99, 2015.

[58] Oren Rippel and Lubomir Bourdev. Real-time adaptive image compression. In *Int. Conf. Machine Learning*, pages 2922–2930, 2017.

[59] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Int. Conf. Medical Image Computing Computer-assisted Intervention*, pages 234–241. Springer, 2015.

[60] Amir Said. Introduction to arithmetic coding-theory and practice. *Hewlett Packard Laboratories Rep.*, pages 1057–7149, 2004.

[61] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. PixelCNN++: A PixelCNN implementation with discretized logistic mixture likelihood and other modifications. In *Int. Conf. Learning Representations*, 2017.

[62] Claude Elwood Shannon. A mathematical theory of communication. *Bell System Tech. J.*, 27(3):379–423, 1948.

[63] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[64] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi. The JPEG 2000 still image compression standard. *IEEE Signal Process. Mag.*, 18(5):36–58, 2001.

[65] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, Thomas Wiegand, et al. Overview of the high efficiency video coding(HEVC) standard. *IEEE Trans. Circuits Syst. Video Technol.*, 22(12):1649–1668, 2012.

[66] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. LSTM neural networks for language modeling. In *Int. Speech Commun. Assoc.*, 2012.

[67] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. In *Int. Conf. Learning Representations*, 2017.

[68] 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. In *Int. Conf. Learning Representations*, 2016.

[69] 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 *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5435–5443, 2017.

[70] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 586–591. IEEE, 1991.

[71] Paul Viola, Michael Jones, et al. Rapid object detection using a boosted cascade of simple features. In *IEEE Conf. Comput. Vis. Pattern Recog.*, volume 1, page 3, 2001.

[72] Graham Wade. *Signal coding and processing*. Cambridge university press, 1994.

[73] Gregory K Wallace. The JPEG still picture compression standard. *IEEE Trans. Consumer Electron.*, 38(1):xviii–xxxiv, 1992.

[74] Naiyan Wang and Dit-Yan Yeung. Learning a deep compact image representation for visual tracking. In *Neural Inf. Process. Syst*, pages 809–817, 2013.

[75] Zhou Wang and Al Bovik. *Modern Image Quality Assessment*. Morgan & Claypool Publishers, 1st edition, 2006.

[76] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):600–612, 2004.

[77] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *37th Asilomar Conf. Signals, Syst. and Comput.*, pages 1398–1402, 2003.

[78] Wikipedia. Morse code. https://en.wikipedia.org/w/index.php?title=Morse_code&oldid=895323877, 2019.

[79] Ian H Witten, Radford M Neal, and John G Cleary. Arithmetic coding for data compression. *Commun. ACM*, 30(6):520–540, 1987.

[80] John Wright, Allen Y Yang, Arvind Ganesh, S Shankar Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2):210–227, 2008.

[81] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *Neural Inf. Process. Syst*, pages 341–349, 2012.

[82] XK Yang, WS Ling, ZK Lu, Ee Ping Ong, and SS Yao. Just noticeable distortion model and its applications in video coding. *Signal Process. Image Commun.*, 20(7):662–680, 2005.

[83] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Trans. Image Process.*, 26(7):3142–3155, 2017.

[84] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual U-Net. *IEEE Geosci. Remote Sensing Lett.*, 15(5):749–753, 2018.

[85] Wenyi Zhao, Rama Chellappa, P Jonathon Phillips, and Azriel Rosenfeld. Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 35(4):399–458, 2003.

[86] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv:1606.06160*, 2016.