UNIVERSITÉ PARIS XIII - SORBONNE PARIS NORD (USPN)

École Doctorale Galilée

Compression d'images par apprentissage profond de bout en bout

End-to-End Deep Learning Image Compression

THÈSE DE DOCTORAT

présentée par **Bouzid AREZKI**

pour l'obtention du grade de

DOCTEUR EN SCIENCES POUR L'INGENIEUR

soutenue le 6 décembre 2024 devant le jury d'examen composé de :

Marco CAGNAZZO, Professeur - Université de Padova	-	Rapporteur
François-Xavier COUDOUX, Professeur - Université Poly. Haut de France	-	Examinateur
Olivier DEFORGES, Professeur - INSA, Rennes	-	Rapporteur
Pierre DUHAMEL, DR CNRS émérite - L2S CentraleSupélec	-	Examinateur
Lucile SASSATELLI, Professeure - Université Côte d'Azur	-	Présidente
Fangchen FENG, Maître de Conférences - Université Sorbonne Paris Nord	-	Co-encadrant
Anissa MOKRAOUI, Professeure - Université Sorbonne Paris Nord	-	Directrice

Acknowledgement

First and foremost, I would like to thank Almighty God for giving me the strength, wisdom, and perseverance to complete this thesis. Without His guidance, none of this would have been possible.

I would like to express my deepest gratitude to my supervisors, Anissa MOKRAOUI and Fangchen FENG, for their invaluable guidance, unwavering support, and insightful feedback throughout the entire research process. Your patience, encouragement, and expertise have been instrumental in shaping this work.

I would also like to thank my committee members, Pierre DUHAMEL, François-Xavier COUDOUX, Lucile SASSATELLI, Olivier DESFORGE, and Marco CAGNAZZO, for their constructive criticism and thoughtful suggestions, which significantly enriched my research.

I am especially grateful to my family for their endless love, encouragement, and sacrifices. To my spouse, parents, your faith in me provided the motivation I needed to persevere. Thank you for your patience and understanding during the challenging moments.

A heartfelt thanks goes to my colleagues and fellow researchers in the L2TI laboratory, whose collaboration, camaraderie, and discussions helped make this journey more enjoyable and intellectually stimulating. To everyone who has supported me on this journey, I am deeply grateful.

Résumé

L'explosion des images haute résolution, couplée aux contraintes de stockage et de bande passante, souligne l'importance de développer des techniques de compression d'images. Bien que les méthodes traditionnelles soient généralement efficaces, elles peinent souvent à préserver la qualité d'image à des taux de compression élevés tout en minimisant la complexité de calculs. Afin de surmonter ces défis, cette thèse étudie l'intégration des modèles d'apprentissage profond, tels que les réseaux de neurones convolutifs (CNN), les transformeurs et les modèles d'espace d'état (SSM). Nous proposons trois approches novatrices. La première approche propose le transformeur Swin sans encoder la position (Non-Positional Encoding) (SwinNPE), qui simplifie la complexité du modèle original tout en capturant efficacement le contexte local. La deuxième approche présente une architecture de compression d'images basée sur un modèle d'espace d'état (SSMIC). Cette architecture trouve un bon équilibre entre efficacité de compression, complexité de calcul et latence, la rendant idéale pour des applications en temps réel notamment sur des dispositifs aux ressources limitées. Enfin, la troisième approche propose un codec universel, qui s'appuie sur une architecture d'auto-encodeur variationnel (VAE), spécifiquement entraîné pour un seul compromis débit-distorsion. Grâce à une stratégie de facteur d'échelle appliquée à ce VAE déjà entrainé, il devient possible de comprimer des images à différents débits sans nécessiter le réentraînement de ce VAE. Les résultats expérimentaux montrent bien que nos modèles de compression offrent des performances compétitives en termes de complexité de calculs et d'efficacité, rivalisant ainsi les meilleures méthodes de l'état de l'art.

Mots-clés : Compression d'image, Apprentissage profond, Auto-encodeur variational, Modèle de représentation d'état, Transformeur, Optimisation débit-distorsion, Débit variable.

Abstract

The explosion of high-resolution images, coupled with storage and bandwidth constraints, highlights the importance of developing image compression techniques. While traditional methods are generally effective, they often struggle to preserve image quality at high compression rates while minimizing computational complexity. To overcome these challenges, this thesis studies the integration of deep learning models, such as convolutional neural networks (CNNs), transformers, and state-space models (SSMs). We propose three innovative approaches. The first approach introduces the Swin transformer without positional encoding (Non-Positional Encoding) (SwinNPE), which simplifies the complexity of the original model while effectively capturing local context. The second approach presents an image compression architecture based on a state-space model (SSMIC). This architecture strikes a good balance between compression efficiency, computational complexity, and latency, making it ideal for real-time applications, particularly on resource-constrained devices. Finally, the third approach proposes a universal codec, which relies on a variational auto-encoder (VAE) architecture specifically trained for a single rate-distortion trade-off. By applying a scaling factor strategy to this pre-trained VAE, it becomes possible to compress images at different rates without the need for retraining this VAE. Experimental results clearly demonstrate that our compression models offer competitive performance in terms of computational complexity and efficiency, rivaling the best state-of-the-art methods.

Keywords: Image compression, Deep learning, Variational auto-encoder, Space state representation Model, Transformer, Rate-distortion optimization, Variable-bitrate.

Contents

List of Figures

List of Tables

Acronyms

AD Arithmetic Decoding

AE Arithmetic Encoding

AG-VAE Asymmetric Gained Variational Auto-encoder

AR AutoRegressive

BD-rate Bjontegaard Delta-Rate

BPG Better Portable Graphics

BPP Bit Per Pixel

CCM Checkerboard Context Model

CGAM Channel-Gained Adaptive Module

CNN Convolutional Neural Network

CNNs Convolutional Neural Networks

CPU Central Processing Unit

CTMs Continuous-Time Models

CW-MSA Convolutional Window-based Multi-head Self-Attention

DCT Discrete Cosine Transform

DWT Discrete Wavelet Transform

ELIC Efficient Learned Image Compression

EVC Efficient single-model Variable-bit-rate Codec

FCNN Fully Connected Neural Network

FCNNs Fully Connected Neural Networks

FLOPs FLoating-point Operations Per second

GDN Generalized Divisive Normalization

GHz GigaHertz

GMACs Giga Multiply-ACcumulate operations per second

GPU Graphics Processing Unit

GRU Gated Recurrent Unit

HEVC High Efficiency Video Coding

IGDN Inverse Generalized Divisive Normalization

LSTM Long Short-Term Memory

MACs Multiply-ACcumulate operations per second

MAEs Modulated Auto-Encoders

MHSA Multi-Head Self Attention

MLIC Multi-reference entropy model for Learned Image Compression

MLP Multi-Layer Perceptron

MLPs Multi-Layer Perceptrons

MS-SSIM Multi Scale Structural Similarity Index Measure

MSE Mean Squared Error

multi-RD multi-rate-distortion

NLAM Non-Local Attention Module

NLN Non-Local Network

NLP Natural Language Processing

NN Neural Network

NNs Neural Networks

ODEs Ordinary Differential Equations

p.p. Percentage Point

PE Positional Encoding

PSNR Peak Signal to Noise Ratio

R-D Rate-Distortion

RGB Red Green and Blue

RLAUs Residual connected Lightweight Attention Units

RMSNorm Root Mean Square Normalisation

RNAB Residual (Non-)local Attention Block

RNN Recurrent Neural Network

RNNs Recurrent Neural Networks

SABR Spatially Adaptive Bit Rate

SATD Sum of Absolute Transformed Difference

SlimCAEs Slimmable Compressive Auto-Encoders

SS2D 2D Selective Scan

SSIM Structural Similarity Indexing Measure

SSM State Space Model

SSMIC State Space Model-based Image Compression

SSMs State Space Models

subAEs sub-Auto-Encodes

SW-MSA Shifted Window Multi-head Self-Attention

VAE Variational Auto-Encoder

ViT Vision Transformer

VSS Visual State Space

W-MSA Window-based Multi-head Self-Attention

ZOH Zero-Order Hold

1. Introduction

Contents

1.1. Thesis Context

In the fast-evolving digital age, the need for efficient image compression is more crucial than ever. The proliferation of high-resolution images across the internet, combined with the constraints of storage and bandwidth, has driven the need for advanced compression techniques. The primary goal of image compression is to reduce the amount of data required to represent an image while preserving a level of visual quality that meets the viewer's expectations.

Image compression methods are broadly classified into two types: lossless and lossy compression. Lossless techniques compress images without any data loss, allowing for the exact reconstruction of the original image. However, these methods typically result in lower compression ratios. Conversely, lossy compression techniques achieve higher compression by discarding some information, which leads to a reduction in the quality of the reconstructed image. Despite advances in research, traditional image compression methods often rely on manually engineered features, which can limit their effectiveness in preserving image quality at higher compression rates.

Recent advancements in deep learning, particularly in neural networks, have opened up new possibilities for image compression. Techniques such as Convolutional Neural Networks (CNNs), Transformers, and more recently, State Space Models (SSMs), have demonstrated significant potential in overcoming the limitations of traditional approaches. The ability of learnable methods to adaptively extract features from images, making them *"data-dependent"*, has revolutionized the field, offering a promising solution to these challenges. By training neural networks on large image datasets, these models can achieve more efficient compression and deliver higher-quality reconstruction at greater compression ratios

compared to traditional methods.

Despite the success of learnable image compression methods, they may not be well-suited for real-time applications where both speed and efficiency are crucial. The complexity of neural networks, particularly those with deep architectures or attention mechanisms, often results in significant computational overhead. This can lead to increased latency during the encoding and decoding processes, making these approaches impractical for scenarios requiring immediate image processing, such as video streaming, live broadcasting, or on-device image compression in mobile applications. Additionally, the high memory and processing power requirements of these models can hinder their deployment in resource-constrained environments, such as embedded systems or devices with limited hardware capabilities. Consequently, there is a growing need for more advanced techniques that can handle the increasing complexity of these methods.

This thesis investigates the intersection of deep learning and image compression, intending to develop innovative frameworks that achieve high compression efficiency while also reducing computational complexity and latency, making them suitable for real-world applications. By leveraging state-of-the-art techniques, such as convolutional and transformer models, and integrating SSMs, this research aims to push the boundaries of image compression, offering solutions that are both powerful and practical.

1.2. Objectives and Contributions

While much research on auto-encoder-based coding schemes has concentrated on improving compression performance, this thesis will shift the focus to achieving better compression efficiency with reduced computational costs. Specifically, we

aim to develop an auto-encoder-based framework that achieves higher efficiency by reducing computational complexity and time consumption while enhancing compression performance. Our goal is to create models that are both efficient and effective, with fewer parameters and lower computational requirements.

The key contributions of this thesis are outlined as follows:

- First, we address the image compression problem by introducing the Swin Non-Positional Encoding (SwinNPE) transformer. SwinNPE improves efficiency while reducing the number of model parameters. Specifically, we generalize the Swin cell [5] and propose the Swin convolutional block, which better handles local correlations between image patches. Additionally, this Swin convolutional block improves the capture of the local context between tokens. We argue that the convolutional operation inside the Swin block serves as a more efficient alternative to positional encoding, strengthening local dependencies and reducing overall complexity.

- Second, we propose a State Space Model-based Image Compression (SSMIC) architecture. This innovative architecture strikes a balance between compression efficiency, computational complexity, and latency, making it well-suited for practical multimedia processing applications. SSMIC integrates state space models from the Mamba model [17] into the image compression pipeline, improving contextual reasoning while optimizing computational and memory requirements. It is specifically designed for efficient real-time image compression on resource-constrained devices.

- Finally, we propose a universal end-to-end Variational Auto-Encoder (VAE) framework for lossy image compression. This model offers a flexible and efficient approach to variable-rate compression by employing a modulation

that adjusts the representation of image features, optimizing the trade-off between bitrate and image quality. This adaptability eliminates the need for designing multiple models (i.e. retraining across different compression rates), offering a practical solution for diverse multimedia applications where bitrate flexibility is crucial. Experimental results demonstrate that our VAE-based approach achieves competitive performance while significantly reducing computational complexity, making it suitable for real-time applications on resource-constrained devices.

1.3. Thesis Outline

This thesis is structured as follows:

Chapter 2 reviews the prominent neural network models commonly used in the context of image compression. It also examines existing deep learning-based image compression techniques, highlighting the gaps that this research aims to address.

Chapter 3 introduces the Convolutional Transformer-Based Image Compression architecture, providing an in-depth explanation of the proposed architecture with a comprehensive analysis of the experimental results.

Chapter 4 presents an efficient image compression method leveraging advanced state space models, detailing the architecture and demonstrating its efficiency in comparison to state-of-the-art methods.

Chapter 5 presents a universal neural network for lossy image compression. It details a strategy that utilizes a single variable-rate VAE framework trained for a specific rate-distortion trade-off, allowing images to be encoded and decoded at multiple bitrates without the need for retraining the VAE architecture.

Finally, Chapter 6 concludes the thesis, discussing its limitations and proposing directions for future research.

1.4. Publications

1.4.1. International Conference Papers

- Arezki Bouzid, Mokraoui Anissa and Feng Fangchen, "Efficient Image Compression Using Advanced State Space Models", *IEEE 26th International Workshop on Multimedia Signal Processing (MMSP)*, Purdue University, West Lafayette, Indiana, U.S.A, October 2–4, 2024.

- Arezki Bouzid, Feng Fangchen, and Mokraoui Anissa, "Universal End-to-End Neural Network for Lossy Image Compression", *European Signal Processing Conference (EUSIPCO)*, Lyon, France, August 2024.

- Arezki Bouzid, Feng Fangchen, and Mokraoui Anissa, "Convolutional Transformer-Based Image Compression", *IEEE International on Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, 154–159, Poznań, Poland, September 2023.
  Best Paper Presentation Award.

1.4.2. National Conference Papers

- Bouzid Arezki, Anissa Mokraoui and Fangchen Feng, "Compression efficace d'images basée sur un modèle de représentation d'état", *23ème édition de la conférence Compression et Représentation des Signaux Audiovisuels (CORESA)*, Rennes, France, novembre 2024.

- Bouzid Arezki, Fangchen Feng and Anissa Mokraoui, "Transformer-Based Image Compression Without Positional Encoding", *22ème édition de la conférence Compression et Représentation des Signaux Audiovisuels (CORESA)*, Lille, France, juin 2023.

2. Image Compression Using Deep Learning: State-of-the-Art

Contents

Summary of this Chapter

This chapter provides an in-depth overview of image compression techniques using deep learning, with a particular emphasis on auto-encoder-based approaches. It begins by outlining traditional compression methods, followed by a review of neural network models including Fully Connected Neural Network (FCNN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), attention mechanisms, and State Space Model (SSM). The focus is on auto-encoders, highlighting their applications in both single-rate and variable-rate compression, and demonstrating how these models enhance data representation and compression efficiency. Additionally, this chapter discusses the role of entropy coding in optimizing compression rates, while also briefly addressing other innovative neural network approaches, such as transform coding and intra-prediction schemes. Overall, it establishes auto-encoder-based methods as the forefront of cutting-edge image compression techniques leveraging deep learning.

2.1. Introduction

Traditional image compression techniques can be classified into two main categories: predictive coding schemes, as well as transform coding schemes, [18]. Notably, the latter category has attracted considerable attention over the recent decade. Transform coding schemes, in particular, enable the transformation of images into new domains that possess more relevant characteristics (e.g., signal decorrelation, frequency, multi-scale analysis, compact representation, etc) [19] making them more suitable for effective compression.

With the advent of deep learning, transform coding schemes have witnessed a paradigm shift. Neural networks have emerged as powerful tools for learning optimal representations of images, enabling adaptive and data-driven compression. These models can capture complex patterns and dependencies in visual data, significantly improving data reduction efficiency while maintaining high visual quality.

While deep learning has largely driven recent advancements in image compression, other approaches, such as hybrid methods that combine neural networks with traditional coding schemes, have also been proposed. These methods extend conventional techniques by integrating machine learning components to optimize both transform and prediction steps, offering promising alternatives in specific contexts.

This chapter, however, primarily focuses on auto-encoder-based techniques for image compression, which belong to the transform coding schemes. It provides a thorough exploration of state-of-the-art architectures, including, CNNs, transformers, and attention-based models. Furthermore, we also provide a brief overview of other approaches, such as neural network-enhanced transform coding

and predictive coding.

The remainder of this chapter is organized as follows. First, background on Neural Networks (NNs) is provided in Section 2.2. Then, auto-encoder-based techniques and other categories of NNs are discussed in Section 2.3 and 2.4 respectively. Finally, Section 2.5 concludes this chapter.

2.2. Background on Neural Network Models

Identifying nonlinear functions that accurately capture relationships within data (image, text, etc.) can be quite challenging. To tackle this issue, Neural Network (NN) were developed, as they can approximate any nonlinear function by learning from the data they encounter. In the following subsections, we will present the most effective neural network models, with a particular emphasis on those applied in image compression.

2.2.1. Fully Connected Neural Network (FCNN)

The Multi-Layer Perceptron (MLP) is one of the earliest neural network architectures, originating from the pioneering work of psychologist Frank Rosenblatt [20]. Inspired by natural neural networks found in the brain, Rosenblatt introduced the concept of a perceptron. An MLP consists of stacked layers containing numerous processing units, commonly referred to as neurons or perceptrons. Operating as a feed-forward neural network, it includes an input layer, at least one hidden layer, and an output layer. This structure ensures that each neuron in a layer is densely connected to every neuron in the preceding and subsequent layers. Each neuron receives the output from the preceding layer x_{l-1}, where l denotes the

layer number, and applies specific operations to it:

$$y_l^{(i)} = g_l(w_l^{(i)} x_{l-1} + b_l^{(i)}), \tag{2.1}$$

where i is the neuron number in layer l, $y_l^{(i)}$ represents the output of the neuron, g_l is the activation function, $w_l^{(i)}$ are the weights of the neuron, and $b_l^{(i)}$ is the bias term.

It is important to highlight that Fully Connected Neural Networks (FCNNs), including Multi-Layer Perceptrons (MLPs), have a wide range of applications, ranging from natural language processing and image recognition to more complex tasks like time series estimation and sentiment analysis. This versatility makes them both fascinating and highly valued in the field of neural networks [21, 22], including in deep learning-based image compression methods [23].

2.2.2. Convolutional Neural Network (CNN)

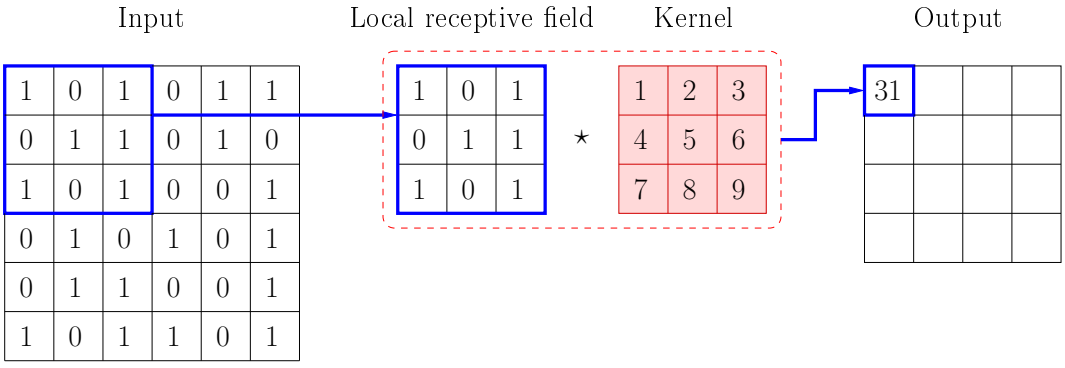


Figure 2.1: Convolution operation.

A CNN is a type of artificial neural network primarily designed for computer vision tasks, including image classification [24], object detection [25], image compression [8,26], image segmentation, and feature extraction in general [27]. Unlike

other networks that process input data in segments, a CNN analyzes the entire input at once. This capability allows CNNs to effectively extract features and hidden information, which is crucial for identifying spatial hierarchies in images, such as edges, textures, and patterns.

Typically, a CNN consists of an input layer, a series of convolutional layers, each followed by activation functions, and an output layer. Each convolutional layer contains one or more kernels K:

$$(I * K)(i, j) = \sum_{M-1}^{m=0} \sum_{N-1}^{n=0} I(i + m, i + n)K(m, n), \tag{2.2}$$

where M and N are the dimensions of the kernel K.

During the forward pass in a two-dimensional (2D) convolution, each kernel in K scans the input $I \in \mathbb{R}^{W \times H \times C}$ across its dimensions (width W, height H and depth C), performing the convolution operation to generate a 2D activation map $(I * K)$. The outputs from all kernels are then combined to produce the layer's output, as illustrated in Figure 2.1.

Depending on the specific application, a CNN typically includes additional output layers beyond the convolutional ones. For instance, batch normalization layers can be used to normalize the inputs of layers through re-centering and re-scaling, thereby improving training stability and training speed. Pooling layers, such as max pooling or average pooling, are also employed to reduce the dimensionality of feature maps, providing the network with a more comprehensive view of the input.

2.2.3. Recurrent Neural Network (RNN)

Jeffrey L. Elman [28] was the first to introduce the RNN structure for processing sequential and time-series data by proposing a loop-based RNN cell. In this context, activation functions initially introduced in [29] are fed back into the RNN cell. Since this pioneering work, RNNs have achieved significant success due to their ability to retain memory, allowing computations to consider historical information. The unfolded structure of a recurrent neural network cell is illustrated in Fig 2.2.

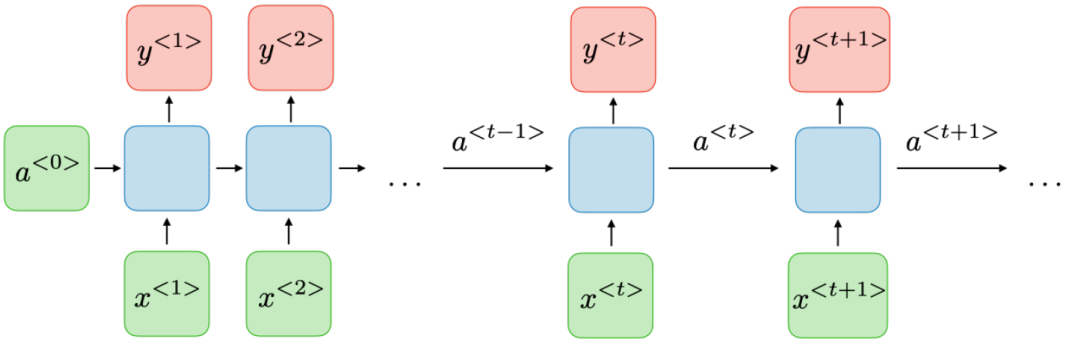


Figure 2.2: Workflow of a recurrent neural network cell.

We denote t as the time-step, $x^{<t>}$ as the input, $y^{<t>}$ as the output, and $a^{<t>}$ as the activation fed back into the network cell. The variables $a^{<t>}$ and $y^{<t>}$ can be expressed as follows:

$$a^{<t>} = g_1(w_{aa}a^{<t-1>} + w_{ax}x^{<t>} + b_a), \tag{2.3}$$

$$y^{<t>} = g_2(w_{ya}a^{<t>} + b_y). \tag{2.4}$$

In this context, g_1 and g_2 represent two activation functions. The parameters w_{aa} and w_{ax} are applied to $a^{<t-1>}$ and $x^{<t-1>}$ respectively to compute $a^{<t>}$, with b_a

serving as the bias. Similarly, $y^{<t>}$ is derived by applying the parameter w_{ya} to $a^{<t>}$ and adding the bias b_y. The output $y^{<t>}$ integrates information from the current input $x^{<t>}$ as well as from all previous inputs $\{x^{<1>}, \ldots, x^{<t>}\}$.

Recurrent Neural Networks (RNNs) face challenges like vanishing gradients and short-term compared to other neural network architectures. To address these issues and learn long-term dependencies, two specific variants, known as Long Short-Term Memory (LSTM) networks and Gated Recurrent Unit (GRU), have been developed. More details about these architectures can be found in [30] and [31]. Understanding these advancements is essential because RNNs are powerful tools for modeling sequential and temporal data. Understanding their structure and applications is essential for leveraging their capabilities in various deep-learning tasks, including Language modeling, machine translation, text generation, sentiment analysis, and image compression.

2.2.4. Attention Mechanism

Attention mechanisms have been successfully employed across a variety of tasks. In machine translation, for instance, by focusing on the most relevant parts of the input sentence, attention mechanisms have significantly improved the accuracy of machine translation models. They have also proven effective in Text Summarization [32], Image Classification [33], object detection [34] and image compression [7]. In recent years, these attention mechanisms have emerged as a crucial component in neural networks, greatly improving model performance across various domains, including natural language processing, computer vision, and image compression. The attention mechanism was first introduced by Bahdanau et al. in 2014 [35] in the context of machine translation. The fundamental idea of attention is to enable the model to focus on relevant parts of the input sequence

as it generates each part of the output sequence. This mechanism mimics the human cognitive process of selective attention, enabling the model to emphasize specific aspects of information while disregarding others. As a result, it enhances the model's capability to handle long-range dependencies and complex structures in the data.

Despite the variations in attention mechanism, such as the additive method proposed by Bahdanau et al. [35] or the dot-product method proposed by Vaswani et al. [1] as illustrated in Figure 2.3, or Self-Attention (a special case of dot-product), the fundamental process remains the same. A Score S is computed for each pair (x_i, y_j) from the input sequence $X = \{x_1, x_2, \ldots, x_n\}$ and the output sequence $Y = \{y_1, y_2, \ldots, y_m\}$ which represents the relevance of the input element to the current output element. A softmax function is then applied to generate a set of weights that reflect the importance of each input element relative to the current output element:

$$\alpha_{ij} = \text{Softmax}(S(x_i, y_j)). \tag{2.5}$$

Building on this, a context vector V_{cxt} is generated through a weighted sum of each element in the input x and its corresponding attention weight, capturing the relevant information necessary for generating the current output element. This context vector, combined with the model's current state, produces the final output element:

$$V_{\text{cxt}}(y_i) = \sum_{i=1}^{n} \alpha_{ij} x_i. \tag{2.6}$$

More specifically, as illustrated in Figure 2.3, *"Scaled Dot-Product Attention"* proposed by Vaswani et al. [1] computes the attention across a set of queries

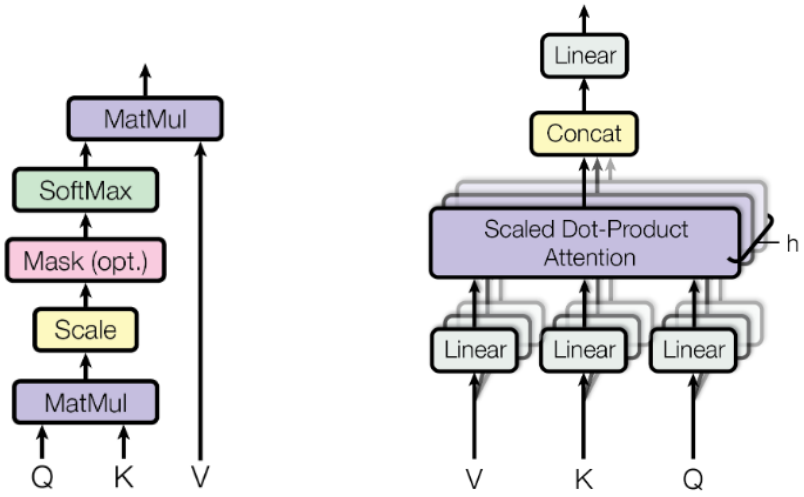


Figure 2.3: Scaled Dot-Product Attention (left). Multi-Head Attention consists of several attention layers running in parallel (right) (schemes taken from [1]).

simultaneously:

$$\text{Attention}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{d_k}})V, \tag{2.7}$$

where queries, keys, and values are grouped into matrices Q, K, and V which are linearly projected from the input, adding a scale factor $\frac{1}{\sqrt{d_k}}$.

Transformers

The Transformer model has fundamentally revolutionized the use of attention mechanisms in neural networks. Unlike RNNs and CNNs, which depend on sequential processing, the Transformer uses self-attention mechanisms to handle input and output sequences with remarkable efficiency. The latter consists of an encoder-decoder architecture, where both the encoder and decoder are composed of multiple layers of self-attention and feed-forward networks, as shown in Figure 2.4. The Key components of the Transformer include Multi-Head Attention and Positional Encoding, which will be described below.

Multi-Head Attention – This enhances the self-attention mechanism by allowing the model to focus on different segments of the input sequence simultaneously. Multiple attention heads h operate in parallel, each attending to different aspects of the input. Rather than applying a single attention function, as shown in Eq. 2.7 (with d_{model}-dimensional keys, values and queries), Vaswani et al. [1] project the queries, keys and values h times using different linear projections to d_k, d_k, and d_v dimensions respectively. Each linear projection of these sets of keys, values, and queries performs the attention function from Eq. 2.7 in parallel given d_v-dimensional output values concatenated and once again projected as depicted in Figure 2.3:

$$\begin{aligned} &\text{MultiHead}(Q, K, V) = \text{Concat}(h_1, \ldots, h_h)W^O \\ &h_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.8}$$

where the projections are performed through the parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W_i^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

Positional Encoding (PE) – Since the Transformer lacks inherent sequence information, positional encodings are added to the input embeddings as introduced in Vaswani et al. [1], to provide the model with information about the position of each token in the sequence. The Transformer architecture in [1] employs a specific method for positional encoding:

$$\begin{aligned} \text{PE}(\text{pos}, 2i) &= sin(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}) \\ \text{PE}(\text{pos}, 2i+1) &= cos(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}) \end{aligned} \tag{2.9}$$

where $\text{PE}(\text{pos}, 2i)$ and $\text{PE}(\text{pos}, 2i+1)$ represent the encoding for the $2i$-th and $2i+1$-th dimension of the position pos respectively, pos indicates the position

of the token in the sequence. This approach results in a matrix with the same dimension as the input embeddings, which can be added to the input embeddings before feeding them into the model, as illustrated in Figure 2.4. While the sine and cosine method is widely used, other approaches have also been proposed:

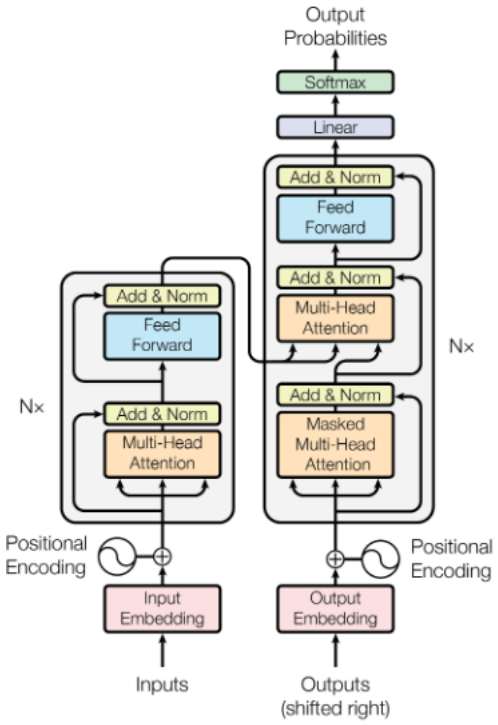


Figure 2.4: The Transformer - model architecture (scheme taken from [1]).

Learned Positional Embedding – This method allows the model to optimize the positional representations directly from the data, allowing positional embeddings to be learned during training rather than relying on fixed functions [1].

Relative Positional Encoding – The approach proposed by Shaw et al. [36] encodes the relative distances between tokens rather than their absolute positions. This relative positional encoding effectively captures local dependencies and offers more flexibility than absolute position representations, particularly for visual tasks.

2.2.5. State Space Model (SSM)

SSMs offer a powerful framework for modeling dynamic systems, where observations are generated from underlying states that evolve over time. These models characterize a system using a set of latent (hidden) variables known as states, which evolve over time according to a state transition process. The observed data is generated from these states through an observation process. An SSM consists of two main components: *State Transition* and *Observation* Model.

State Transition Model – The evolution of the hidden state from one-time step to the next can be defined as follows:

$$h_t = f(h_{t-1}, x_t) \tag{2.10}$$

where, h_t denotes the hidden state at time t, h_{t-1} represents the hidden state at the previous time step, f is the state transition function, and x_t signifies process noise or external inputs.

Observation Model – It defines how the observed data is generated from the hidden state:

$$y_t = g(h_t, v_t) \tag{2.11}$$

Here, y_t denotes the observed data at time t, g is the observation function, and v_t represents observation noise.
SSMs can be either linear, where the state transition function f and observation function g are linear functions, or non-linear in which case f and g are non-linear functions, suitable for systems exhibiting non-linear dynamics.

Recently Albert Gu, Karan Goel, and Christopher Ré [37] introduced "S4", an efficient Modeling for handling Long Sequences using Structured State Spaces.

This approach is based on the classical linear state space model, which maps a one-dimensional input signal $x(t) \in \mathbb{R}$ to a one-dimensional output signal $y(t) \in \mathbb{R}$ through a latent state $h(t) \in \mathbb{R}^N$ of dimension N, Typically, we can formulate the process by linear Ordinary Differential Equations (ODEs):

$$\begin{aligned} h'(t) &= \mathbf{A}h(t) + \mathbf{B}x(t), \\ y(t) &= \mathbf{C}h(t), \end{aligned} \tag{2.12}$$

where $\mathbf{A} \in \mathbb{R}^{N\times N}$, $\mathbf{B} \in \mathbb{R}^{N\times 1}$, $\mathbf{C} \in \mathbb{R}^{1\times N}$ are parameters representing neural networks in deep learning.

To handle the discrete input sequence $x = [x_0, x_1, \ldots, x_{L-1}] \in \mathbb{R}^L$, the parameters in equation (2.12) are discretized using a step size Δ, which indicates the resolution of the continuous input $x(t)$ [37]. In particular, the continuous parameters $\mathbf{A}$ and $\mathbf{B}$ are converted into their discrete counterparts $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ using the Zero-Order Hold (ZOH) technique, defined as follows:

$$\begin{aligned} \overline{\mathbf{A}} &= \exp(\Delta\mathbf{A}), \\ \overline{\mathbf{B}} &= (\Delta\mathbf{A})^{-1}(\exp(\Delta\mathbf{A}) - \mathbf{I}).\Delta\mathbf{B}. \end{aligned} \tag{2.13}$$

After discretizing $\mathbf{A}$ and $\mathbf{B}$ to obtain $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$, equation (2.12) can be reformulated as follows:

$$\begin{aligned} h_t &= \overline{\mathbf{A}}h_{t-1} + \overline{\mathbf{B}}x_t, \\ y_t &= \mathbf{C}h_t. \end{aligned} \tag{2.14}$$

SSMs can be efficiently computed using RNNs. This recursive process can be

reformulated and computed as a convolution:

$$\begin{aligned} \overline{\mathbf{K}} &= (\mathbf{C}\overline{\mathbf{B}}, \mathbf{C}\overline{\mathbf{A}\mathbf{B}}, \ldots, \mathbf{C}\overline{\mathbf{A}}^{L-1}\overline{\mathbf{B}}), \\ y &= x * \overline{\mathbf{K}}, \end{aligned} \tag{2.15}$$

where L denotes the length of the input sequence x, and $\overline{\mathbf{K}} \in \mathbb{R}^L$ represents the SSM convolution kernel.

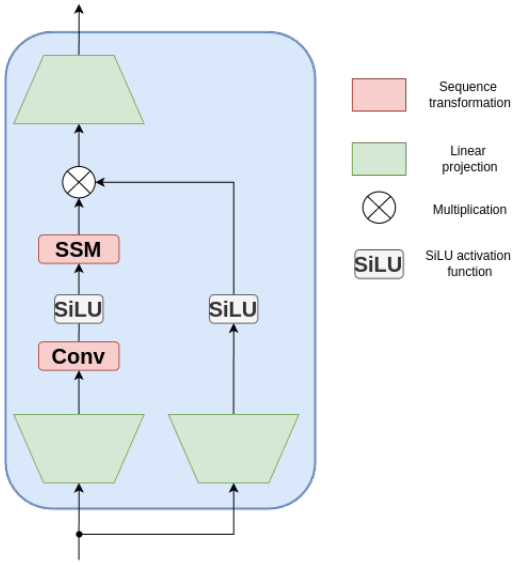


Figure 2.5: Mamba block architecture.

To enhance the efficiency of SSM computations, Mamba [17] as illustrated in Figure 2.5, further incorporates data-dependence to capture contextual information in equation (2.12). It proposes a novel parameterization method for SSMs that integrates an input-dependent selection mechanism, referred to as S6. Although the recurrent nature of SSMs limits full parallelization, Mamba employs structural reparameterization tricks and a hardware-efficient parallel scanning algorithm to significantly improve overall efficiency.

As a result, many studies have adapted Mamba from Natural Language Pro-

cessing (NLP) to the vision domain, including applications in image classification [13,38], multimodal learning [39], and others [40–42]. These adaptations aim to further enhance the capability of SSMs in handling complex and non-linear time-series data. Inspired by these advancements, we aim to implement this type of neural network for image compression.

2.3. Auto-Encoder-Based Coding Schemes

Auto-encoder-based image compression techniques have gained significant attention in recent years due to their effectiveness. In these methods, the encoder acts similarly to the transformation stage, the quantization process mimics sampling, and bitrate estimation serves as a form of regularization on the prior distribution. These similarities make auto-encoders well-suited for image compression. Many approaches have leveraged increasingly sophisticated neural network architectures for each component to improve compression performance. However, despite their promising results, these models have notable limitations. In addition to their high computational cost, they are typically trained end-to-end, meaning a single trained model can usually only generate one fixed bitrate for a given image.

Since the optimal bitrate is often unknown prior to compression, this restricts the practical utility of such models. To address this, several studies have proposed solutions for variable bitrate compression using deep learning. The core concept is to train a single model, or as few models as necessary, that can generate a variety of bitrates during inference, allowing users to select their preferred bitrate. Furthermore, our survey highlights that a key component of these compression models is the entropy model. This model is crucial for accurately estimating the information content of an image, which directly influences overall compression

efficiency. A well-designed entropy model enables the framework to allocate bits adaptively and efficiently, resulting in improved performance in both compression rate and image quality. Over the past two decades, there have been significant advancements in both neural network architecture and probabilistic models for entropy estimation.

In the following sections, we first provide an overview of the generic architecture and end-to-end learning approach of auto-encoder-based coding schemes for image compression. Next, we review the most relevant models for this PhD thesis, categorizing them based on their support for variable bitrate compression. Finally, we explore various approaches that use different entropy models.

2.3.1. Generic NN-based Image Compression Systems

In what follows, we outline the primary architecture and its associated end-to-end learning approach, which are commonly found in most neural network-based image compression methods.

2.3.1.1. Flowchart of the Coding Architecture

A generic neural network-based image compression system is depicted in Figure 2.6. As illustrated in this Figure 2.6, for a given input image x, the following steps are performed:

• **Encoder**: this component, denoted as E in what follows, is parameterized by θ_E and generates a compact representation y often referred to as the latent representation of the original image x which serves as the encoder's input:

$$y = E(x; \theta_E) \tag{2.16}$$

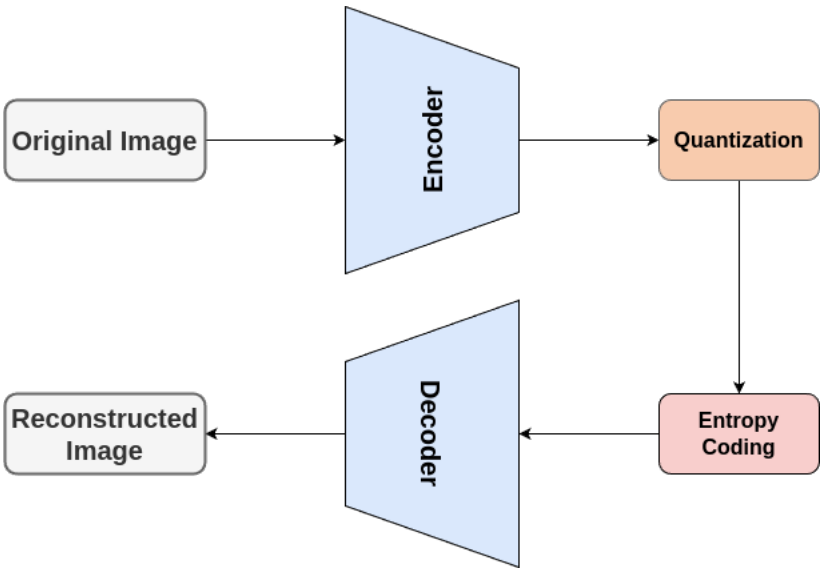


Figure 2.6: Block diagram NN-based image compression system.

• **Quantization and Entropy Coding**: A quantizer Q, followed by an entropy coding model, is then applied to the latent representation y, resulting in the quantized coefficients $\hat{y}$:

$$\begin{aligned} \hat{y} &= Q(y) \\ &= Q(E(x;\theta_E)) \end{aligned} \tag{2.17}$$

• **Decoder**: this component, represented by D and parameterized by θ_D, aims to reconstruct the image $\tilde{x}$ from the encoded representation:

$$\begin{aligned} \tilde{x} &= D(\hat{y};\theta_D) \\ &= D(Q(E(x;\theta_E));\theta_D) \end{aligned} \tag{2.18}$$

This structure is known as an Autoencoder architecture, commonly used in deep learning-based compression methods. The main distinctions among existing AE-

based approaches typically revolve around two key aspects: the employed neural network models (Section 2.2) and the loss function along with the end-to-end training approach used to optimize the model weights, which will be discussed in the following sections.

2.3.1.2. End-to-End Learning Approach

An end-to-end learning approach involves jointly optimizing the entire system, from encoding to decoding, within a single training pipeline. This means that both the encoder and decoder, represented by the parameters (θ_E and θ_D) respectively, are trained simultaneously with θ_P which represents the parameter of the probability distribution estimation model (additional details will be provided below).

To minimize a loss function that balances two main objectives: reducing the bitrate of the encoded image while preserving the quality of the reconstructed image. Consequently, a typical loss function $\mathcal{L}$ corresponds to a Rate-Distortion (R-D) criterion defined as follows:

$$\mathcal{L}(\theta_D, \theta_E, \theta_P, \lambda) = \mathcal{R}(\theta_E, \theta_P) + \lambda \mathcal{D}(\theta_E, \theta_D) \tag{2.19}$$

where λ regulates the trade-off between the rate $\mathcal{R}(\theta_E, \theta_P)$ and distortion $\mathcal{D}(\theta_E, \theta_D)$ terms.

Distortion Term – The distortion term $\mathcal{D}(\theta_E, \theta_D)$ can be defined as follows:

$$\mathcal{D}(\theta_E, \theta_D) = \mathbb{E}_{x \sim p_x}(d(x, \hat{x})) \tag{2.20}$$

where p_x corresponds to the probability distribution of the original image x, and $\hat{x}$ is the reconstructed image. The distortion is typically calculated using a metric such as Mean Squared Error (MSE) or Structural Similarity Indexing Measure (SSIM).

Rate Term – The rate term $\mathcal{R}(\theta_E, \theta_P)$ is defined as follows:

$$\mathcal{R}(\theta_E, \theta_P) = \mathbb{E}_{y \sim p_y}(-log(q_y(y))) \tag{2.21}$$

where p_y refers to the probability distribution of the latent representation y and $q_y(y)$ is the estimated probability distribution of y, and θ_P refers to the parameters of the probability distribution estimation model. Further details on the latter are provided below.

a) Entropy Estimation and Coding Challenges – Entropy coding is computationally intensive and inherently non-differentiable, making it challenging to integrate directly into the training process of deep learning models. As a result, entropy estimation becomes a crucial alternative. According to information theory, the bitrate of an encoded signal is fundamentally constrained by its entropy, which can be derived from the probability distribution of the symbols within the signal. To address this, a probability distribution estimation model, denoted as θ_P, is employed to estimate the latent entropy. This entropy is then incorporated into the loss function to optimize and minimize the coding rate, as shown in Equation 2.21. This method allows for effective rate-distortion optimization without the computational burden of actual entropy coding during training.

b) Quantization Approximation – In the quantization step, the derivative of the rounding function, is almost zero except at integers, making the quantizer challenging to incorporate directly into the gradient-based optimization process. To overcome this issue, the R-D optimization problem can be relaxed using specific

approximations for the quantization. For example, Theis et al. [43] and Ballé et al. [8] proposed substituting the gradient of the quantizer and additive uniform noise during training respectively, most existing methods follow Ballé's approach by adding a uniform noise to the latent representation during training [9,10,44,45], the rate function is then approximated using a given entropy model.

It is important to note that higher values of λ increase the distortion term $\lambda\mathcal{D}$, penalizing the model by prioritizing the distortion reduction. While this leads to improved reconstruction quality, but it also requires higher bitrates for storage. As a result, different models are trained for various levels of reconstruction quality in approaches that do not support variable bitrates, as discussed in the following section 2.3.2.

2.3.2. Single Rate NN Model-based Approaches

The field of image compression has experienced remarkable advancements with the emergence of deep learning-based auto-encoder methods. Numerous methods and techniques have been developed, each designed to enhance compression effectiveness through innovative architectures. In this section, rather than attempting to review all methods, we aim to categorize them based on the key techniques that have significantly contributed to the literature, emphasizing the major advancements in the state of the art. This is what we present below.

a) Incorporating a Layer of Generalized Divisive Normalization – One of the earliest contributions to NN-based image compression was made by Ballé *et al.* [8], who introduced the use of a single layer of Generalized Divisive Normalization (GDN) as an analysis transform, along with an Inverse Generalized Divisive Normalization (IGDN) as a synthesis transform. GDN was found to be more efficient than a sequence of linear operations with point-wise nonlinearities [26], due

to its spatial adaptability and inherent nonlinearity, This improves normalization and stabilization during the training process, contributing to more efficient and robust model performance. As a result, GDN has become a fundamental building block for many image compression coding schemes [8], [44], [9], [10]. GDN can be expressed as follows. Given an input vector v and its corresponding i^{th} channel v_i, the output of GDN u_i at i^{th} channel is expressed as follows:

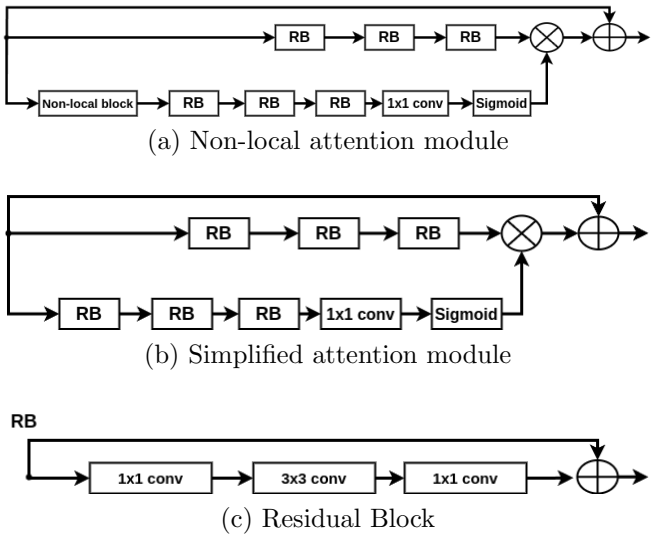
$$u_i(m,n) = \frac{v_i(m,n)}{(\beta_i + \sum_j \gamma_{i,j}(v_j(m,n)^2))^{\frac{1}{2}}} \tag{2.22}$$

where (m,n) denotes the spatial location in the vector v_i, j corresponds to the channel number, and β_i and $\gamma_{i,j}$ are GDN parameters optimized during the training process. To retrieve the original information, the Inverse Generalized Divisive Normalization is applied as follows:

$$v_i(m,n) = u_i(m,n)(\beta_i + \sum_j \gamma_{i,j}(u_j(m,n)^2))^{\frac{1}{2}}. \tag{2.23}$$

b) Improved Architectures Using GDN and Hyperprior – The work in [44] builds upon the single-layer method introduced in [8] by proposing a Factorized Model that incorporates GDN and its inverse IGDN into a stride-based convolutional neural network for both the encoder and decoder. In this approach, each convolutional layer is followed by a GDN layer, improving data normalization and stability throughout the network.

An extended version of the Factorized Model [44] was developed in [9] by introducing a Hyperprior model, enabling the network to more effectively capture the spatial dependencies in the encoder's output. This was accomplished by stacking an additional auto-encoder on top of the existing structure. The work by Ballé et al. [9] established the foundation for numerous Hyperprior architectures in im-



(a) Non-local attention module

(b) Simplified attention module

(c) Residual Block

Figure 2.7: An outline of the simplified attention module(schemes taken from [2]).

age compression [10], [46], [2], [47]. Further details on these architectures will be discussed in Section 2.3.4.

c) Incorporating Residual Blocks with Attention Mechanisms – While Ballé et al. [8, 9, 44] and Minnen et al. [10] deployed CNN-based encoder and decoder networks, studies such as in [2,46–48] and [49] explored different networks using attention modules that have been proven to improve image restoration and compression, as well as residual blocks that allow an increased receptive field and optimize the rate-distortion performance. Although CNN-based methods perform well in resource-constrained environments due to their hardware-efficient convolution operations, their localized receptive fields [50] limit their ability to model global context, thereby constraining overall compression performance.

The attention mechanism performs well in the global perception and thereby benefits redundancy reduction. For instance, Chen et al. [46] propose a hybrid method that uses residual blocks and Non-Local Attention Module (NLAM) as basic units in the encoder-decoder pair. It is composed of three parts as illustrated

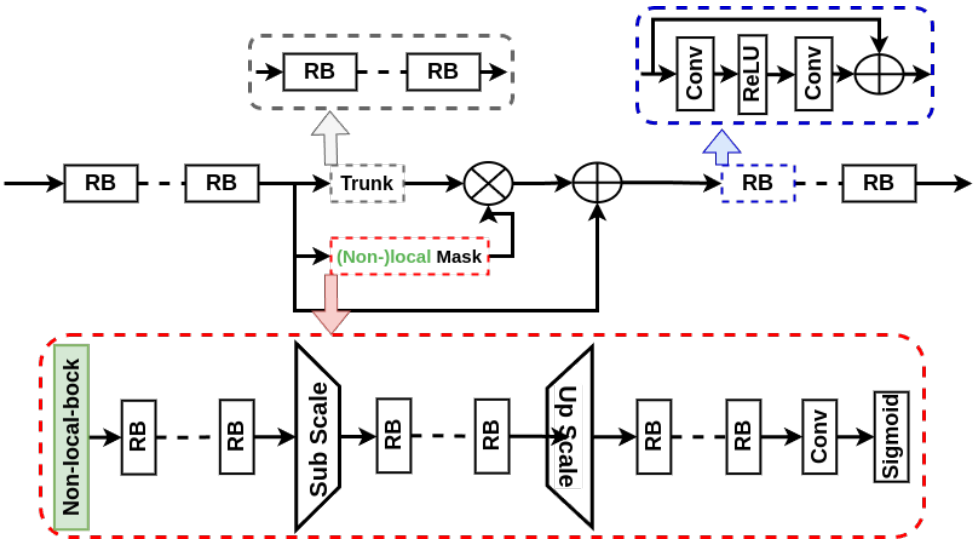


Figure 2.8: An outline of the Residual (non-)local attention block (RNAB).

in Figure 2.7a. The first part consists of three convolution-based residual blocks to generate features. The second part is a Non-Local Network (NLN) proposed in [51], and the last part is a set of three residual blocks, one 1×1 convolution and the sigmoid activation. To achieve adaptive processing in the network, the attention mask is applied to the input feature map via an element-wise multiplication. Chen et al. [2] proposed a simplified version of the NLAM proposed in [46], which omits the NLN component to reduce training time. Their approach utilizes only residual blocks, a convolutional layer, and a sigmoid activation function, as illustrated in Figure 2.7b.

Building on the works of [46] and [2], Patel et al. [47] proposed an image compression scheme that features two encoder-decoder pairs composed of convolutional residual blocks integrated with non-local/self-attention layers. A key aspect of their approach is the incorporation of a saliency model and a salient mask, driven b the insight that the human eye is more sensitive to artifacts in textured regions of images. The saliency model processes the original image to

generate a saliency mask, which serves two primary functions. First, it effectively masks the quantized latent representations, ensuring that more critical areas of the image are prioritized during compression. Second, it enables the distortion loss to assign greater importance to salient regions, thus optimizing the overall visual quality of the compressed image. Once the saliency mask is generated, it is combined with an importance mask using point-wise multiplication to produce the final latent representation. This dual-mask strategy enhances the fidelity of the compression process, allowing details critical to viewers to be better preserved.

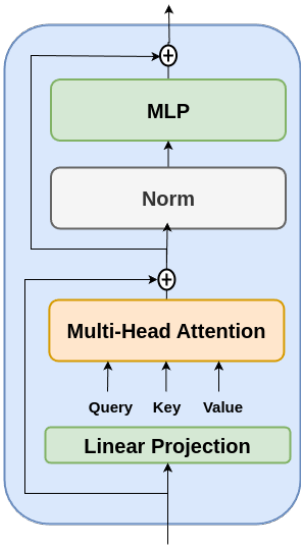


Figure 2.9: An illustration of the Transformer encoder.

In [48, 49, 52], a different Residual (Non-)local Attention Block (RNAB) is employed, which learns mixed attention maps in both channel- and spatial-wise dimensions. This block mainly consists of a trunk branch and a (non-)local mask branch. The trunk branch, illustrated by the blue dashed block in Figure 2.8, consists of several Residual blocks that extract hierarchical features. Meanwhile, the (non-)local mask branch is designed to simultaneously learn mixed attention maps across both channel-wise and spatial-wise, adaptively re-scaling hierarchical

features, as illustrated by the red dashed block. While the trunk branch focuses on capturing local structures through convolutional operations, the (non-)local attention mask branch addresses long-range dependencies across the entire feature map.

d) Transformer-Based Compression Models – Following the introduction of the transformers in vision task [33], many studies have proposed transformer-based approaches for image and video compression [6, 7, 53, 54]. These methods leverage the Multi-Head Self Attention (MHSA) block introduced in [33], as illustrated in Figure 2.9, replacing convolutional operators in the encoder and decoder of the VAE and/or in the entropy model.

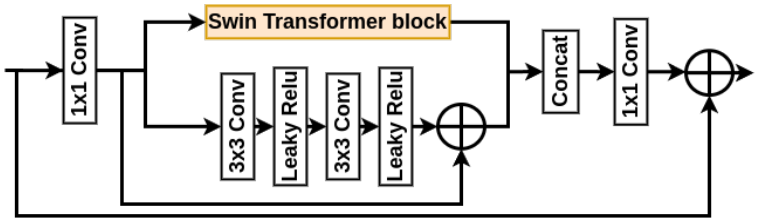


Figure 2.10: An illustration of the hybrid block of Learned Image Compression with Mixed Transformer Convolutional Neural Network Architectures [3].

In [6] and [53], the MHSA block [33] is employed in the entropy model and context model, respectively (for more details on the context model, see Section 2.3.4).

In [7, 54], the authors utilized the Swin-Transformer block, which incorporates a Shifted Window mechanism introduced in [5]. This hierarchical transformer enhances efficiency by restricting self-attention computation to non-overlapping local windows, while still enabling cross-window connections. Additionally, it offers linear computational complexity with respect to image size, in contrast to the quadratic complexity of the standard transformer block [33]. A hybrid method [3] has also been proposed that combines both CNN and Swin-Transformer blocks [5]

(see Figure 2.10).

e) Incorporating Residual Blocks with Sub-Pixel Convolutions – While deep neural networks excel over traditional image codecs in terms of rate-distortion performance, they can be computationally intensive, particularly when incorporating attention mechanisms and transformer blocks. Additionally, since each model is optimized for a single trade-off parameter, the network could end up being heavy. To address this issue, an alternative type of auto-encoder utilizing residual blocks is proposed in [43].

In this work [43], the encoder and decoder are composed of a sequence of residual blocks. The encoder primarily applies convolutions to downsampled data to optimize computation time, while the decoder employs sub-pixel convolutions for upsampling. This sub-pixel architecture enhances the network's computational efficiency, making it well-suited for high-resolution images. Furthermore, they finetuned a pre-trained auto-encoder for different rates by introducing scale parameters, which will be further discussed in Section 2.3.3 on variable-rate neural network models.

f) Recent Advances: State-Space Models for Image Compression – Recently, following the adoption of Mamba from NLP to the vision domain [13,38], [55] introduced the first image compression framework utilizing Visual State Space (VSS) blocks [13] in both the encoder and decoder. Following the transformer encoder architecture in figure 2.9 they replaced the *Multi-head attention* with S6 blocks.

2.3.3. Variable-Rate NN Models-Based Approaches

Although the rate-distortion training framework is highly effective for image compression, developing a compression scheme with an adjustable bitrate is crucial for

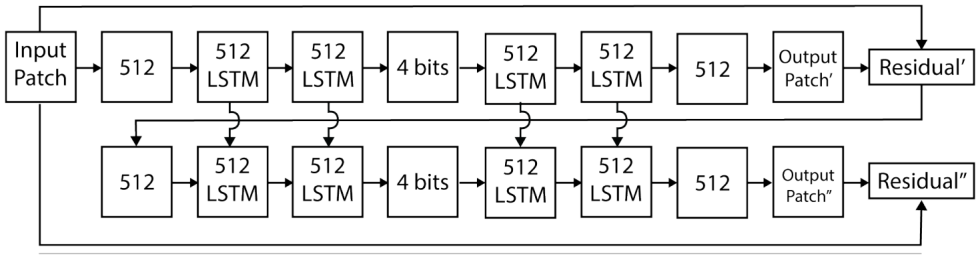


Figure 2.11: Multistage LSTM auto-encoder (schemes taken from [4]).

practical applications that require precise control over target rates. Traditional image compression methods achieve this flexibility using quantization tables to control the quality level of the compressed image, which indirectly influence the final bitrate.

Introducing adjustable bitrate capabilities would significantly enhance both the efficiency and practicality of the compression system by removing the need for retraining when trade-off parameters are modified. In previous approaches (as discussed in Section 2.3.2), each model is tailored for a specific trade-off parameter, leading to the Creation of multiple models that burden the network and reduce overall efficiency.

a) RNN-based Methods – Toderici *et al.* [4] addressed this challenge by introducing the first variable-rate learned image compression method. This innovative approach employs a convolutional LSTM network within a recurrent structure composed of multiple residual auto-encoders. This design enables the network to be trained only once, while progressively transmitting bits to enhance reconstruction quality at incrementally higher bitrates. Let F_t represent the t^{th} residual auto-encoder, E_t and D_t denoting its corresponding encoder and decoder, and B the binarization function. Thus, F_t can be defined as follows:

$$F_t(r_{t-1}) = D_t(B(E_t(r_{t-1}))) \tag{2.24}$$

where r_t represents the residual error at iteration t, with r_0 corresponding to the original input image. The LSTM-based architecture is designed to predict the original input image at each stage, allowing the residual to be computed in relation to this original input:

$$r_t = F_t(r_{t-1}) - r_0 \tag{2.25}$$

The network is trained by minimizing $\|r_t\|_2^2$, where $t \in \{1, .., N\}$ and N represents the number of residual auto-encoders. Figure 2.11 illustrates the architecture of the multistage LSTM auto-encoder.

b) Enhancements in RNN-based Methods – Subsequently, [4] laid the foundation for advanced LSTM-based image compression techniques [56], [57]. In this context, [56] used a recurrent neural network (RNN)-based auto-encoder, introducing both one-shot and additive reconstruction architectures, along with residual scaling. Using the same notation for the encoder, decoder, and binarization function, a single iteration of the network can be defined as follows:

$$\hat{x}_t = D_t(B(E_t(r_{t-1}))) + \gamma \hat{x}_{t-1} \tag{2.26}$$

$$r_t = x - \hat{x}_t, r_0 = x, \hat{x}_0 = 0 \tag{2.27}$$

where $\hat{x}_t$ denotes the progressive reconstruction of the original image x, and $r_t = x - \hat{x}_t$ is the residual between x and the reconstruction $\hat{x}_t$, with $r_0 = x$ and $\hat{x}_0 = 0$. When $\gamma = 0$ (the one-shot reconstruction case), the model proposed by [56] behaves similarly to that of [4]. In this scenario, the original image is predicted at each stage, with the subsequent stage receiving only the residual error between

the original image and its predicted counterpart.

One potential problem with the one-shot reconstruction is that the decoder might be unable to infer the original image. To address this, additive reconstruction is introduced, where γ is set to 1, and the reconstructed image is the sum of the outputs of all iterations. In both reconstruction approaches, the residual starts with a high range that shrinks as iterations progress. To optimize the efficiency of the encoder and decoder, an iteration-dependent gain factor is added to scale the residuals.

In [57], the authors expanded upon the work of Toderici et al. [4], modifying the recurrent architecture to improve information diffusion through the network's hidden states. They also introduced a spatially adaptive bit allocation algorithm and employed a pixel-wise loss weighted by SSIM to enhance reconstruction quality.

c) Bottleneck Scaling – In [43], the authors introduced scale parameters across spatial dimensions, denoted as $\lambda \in \mathbb{R}^M$, into the loss function:

$$-log_2([f(x) \circ \lambda] + u) + \beta d(x, g([f(x) \circ \lambda]/\lambda)) \tag{2.28}$$

where f, g represent the encoder and decoder, respectively and $\circ$ denotes point-wise multiplication. The division is also performed point-wise, with u representing additive uniform noise, x as the input image, and β as the Lagrange multiplier. By incorporating sub-pixel and residual blocks, the network becomes more computationally efficient and is well-suited for high-resolution images. The model was trained incrementally.

d) Feature Modulation – In [58], the authors proposed a conditional auto-encoder, illustrated in figure 2.12, which conditions the model on the Lagrange multiplier across the decoder, encoder, and entropy model. Two rate control

parameters are presented: the Lagrange multiplier λ and the quantization bin size Δ, both serving as conditioning variables for the network. Coarse rate adaptation to a target bitrate is achieved by adjusting the Lagrange multiplier λ, while finer rate adjustments are made by modifying the bin size Δ used during the quantization of the encoded representation.

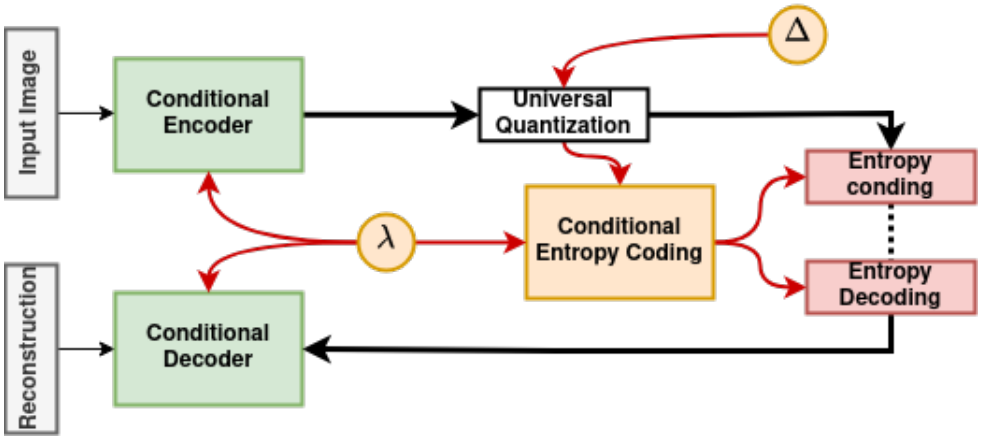


Figure 2.12: Conditional auto-encoder on Lagrange multiplier λ and quantization bin size Δ.

Yang *et al.* [59] extend the concept of the scale parameter in latent space, noting that R-D performance tends to degrade at lower bitrates, limiting the effective bitrate range. To address these issues, they formulate the problem of variable R-D optimization for deep image compression and propose Modulated Auto-Encoders (MAEs), as shown in Figure 2.13. In this approach, a modulation network adjusts the representations of a shared auto-encoder to achieve specific R-D tradeoffs, with the modulation network being jointly trained with the modulated auto-encoder. Akbari *et al.* [60] propose variable rate auto-encoder by altering the loss function in Eq 2.29 to learn multi-rate image features, enabling the model to

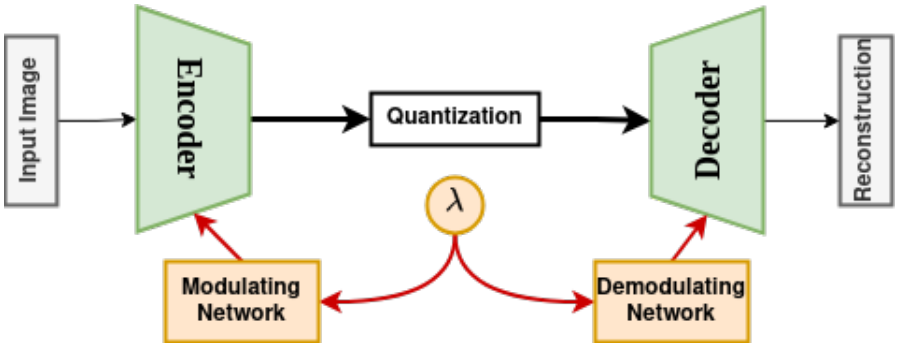


Figure 2.13: Modulated auto-encoder.

operate at different bit rates:

$$\begin{aligned}
\mathcal{L}(\Phi, \Psi) &= 2\mathcal{L}_2 + \mathcal{L}_{MS}, \\
\mathcal{L}_2 &= \sum_{B \in \mathcal{R}} \|x - x'_B\|_2, \\
\mathcal{L}_{MS} &= -\sum_{B \in \mathcal{R}} I_M(x, x'_B) \prod_{j=1}^{M} C_j(x, x'_B) \cdot S_j(x, x'_B),
\end{aligned} \tag{2.29}$$

where x'_B denotes the reconstructed image using B-bit quantizer, with B taking values from the set $R = \{2, 4, 8\}$. This allows the network to be trained simultaneously at different rates. The Multi Scale Structural Similarity Index Measure (MS-SSIM) metric $\mathcal{L}_{MS}$ uses luminance I, contrast C, and structure S to compare the pixels and their neighbors in x and x'. Moreover, MS-SSIM operates at multiple scales where the images are iteratively downsampled the image by factors of 2^j for $j \in [1, M]$.

Another study that addresses variable bit-rate compression is presented in [16], where the problem of continuous rate adaptation problem is explored. Specifically, the authors introduce an Asymmetric Gained Variational Auto-encoder (AG-VAE) which incorporates a pair of gain units into the classical auto-encoder

architecture to achieve discrete rate adaptation within a single model, all while incurring minimal additional computational cost. Furthermore, to enable continuous rate adaptation while maintaining good performance, exponential interpolation is used. The main goal of the gain unit is to exploit the existing redundancies between the channels of the latent representation. The design of the gain unit consists of a gain matrix $M \in \mathbb{R}^{c \times n}$ where n corresponds to the number of gain vectors and c refers to the number of channels in the latent space. Let $m_s = \{m_{s,0}, ..., m_{s,c-1}\}$ represent the gain vector s in the gain matrix, and y be the latent representation with the i^{th} channel denoted by y_i. The rescaling operation in the gain unit can be defined as:

$$\overline{y}_{s,i} = y_i \times m_{s,i}. \tag{2.30}$$

This gain unit simply applies a rescaling operation that leads to a finely adjusted latent representation and a network that allocates more bit rates for the channels with the highest impact on the reconstruction quality.

e) Slimming – Fei et al. [45] introduced Slimmable Compressive Auto-Encoders (SlimCAEs), where the model is trained using a rate-distortion loss across different trade-off parameter values. A layer is considered slimmable when part of its parameters can be discarded—typically by setting them to zero without affecting the model's functionality. Consequently, SlimCAE consists of K sub-Auto-Encodes (subAEs), with each subAE utilizing more parameters than its predecessor. The SlimCAE architecture employs slimmable convolutional layers alongside GDN/IGDN layers, and to ensure optimal rate-distortion performance at varying model capacities, a multi-rate-distortion (multi-RD) optimization loss function is used during training.

2.3.4. Entropy Coding/Entropy Models

Entropy estimation involves the calculation of the unpredictability or randomness of a data source, measuring the amount of information contained in a system. Higher entropy indicates greater randomness. However, since the entropy encoding is computationally expensive and impossible to incorporate into the training structure. To this end, it becomes necessary to estimate the unknown probability distribution of the latent representation. Consequently, multiple entropy estimation frameworks have been studied.

In this respect, multiple entropy models are proposed using a learned Hyperprior [9], [10], [61], statistical analysis [43], [44], [62], [63], [64], and contextual prediction and analysis [10], [46], [65], [56], [57], [61], [66], [67].

a) Incorporating Context Model – Due to the high correlation between spatial neighborhood pixels, a binary context model is proposed in [56] and [57] to estimate the probability distribution. In contrast to [57], who implemented a Spatially Adaptive Bit Rate (SABR) post-processing to adjust the local bit rate for a specified reconstruction quality, Toderici *et al.* used a PixelRNN context-model in [56], drawing inspiration from [68]. Their approach for probability estimation can be described in three steps. First, an initial 7×7 masked convolution is used to eliminate dependencies on future codes and impose a causality constraint, and the larger kernel size is used to increase the receptive field of the LSTM state. Then, a masked convolution-based LSTM scans the convolution's output line by line, and it can capture both short- and long-term dependencies. Finally, to better learn binary code patterns, two 1×1 convolutions are applied. A faster version of the context model PixelRNN [68] is the fully convolutional PixelCNN [69] that

is adopted by [70], [71], [66] and [48].

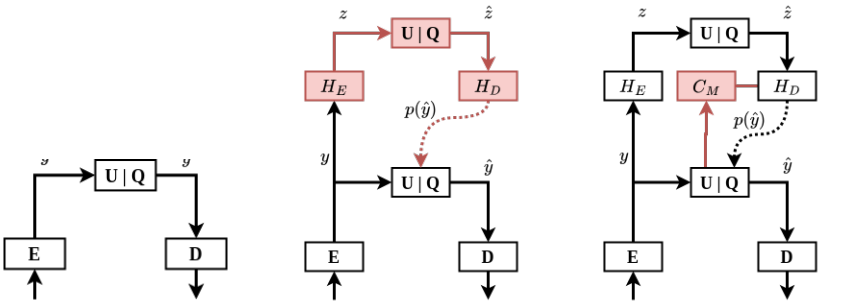
While PixelRNN enforces causality through LSTMs, PixelCNN imposes this by using masked convolutions in all of its layers. However, the sequential encoding and decoding inherent in a context model can be computationally intensive. In this reduce this computational cost, a masking approach is employed. In this context, a mask, combined with a binarization function, efficiently encodes the binary latent. This masked approach is implemented in [70] and [66] involving three steps to create the mask. First, an intermediate feature map from the encoder is fed into an importance map network, generating a content-weighted importance map. Then, each element in the importance map is quantized. Finally, the importance mask is derived from the quantized importance map.

b) Incorporating Side-information – Another interesting approach for entropy estimation is a side-information-guided one. In this respect, in [9], Ballé proposed a hyperprior model that uses side information to capture spatial dependencies among the latent representation.

In fact, a zero-mean Gaussian distribution model with scale parameter σ^2 (variance) is considered to estimate the distribution of the quantized latent distribution. Building on [9], subsequent studies have introduced more accurate entropy models [10], [48], [2], and [61].

For example, in [10], [48], and [61], a joint model is utilized that combines an autoregressive context model with a mean and scale Gaussian distribution hyperprior.

Although the context model benefits from the neighboring elements as input, the fixed shape of a single Gaussian Distribution may have some limits. To address this, a Gaussian Mixture model is used in [2] to achieve arbitrary likelihoods. The operational diagram of the different hyperprior-based coding schemes is illustrated in Figure 2.14.



(a) Baseline (b) Hyperprior (c) Joint Hyperprior and Context model

Figure 2.14: Operational diagrams of learned compression models.

c) Enhancements in Context Model – Enhancements in context modeling have been proposed in [16, 52, 72–74]. A multi-scale context model was implemented using multiple masked convolutions with varying kernel sizes to learn diverse spatial dependencies simultaneously. Studies [46, 70, 75] incorporated 3D masked convolutions to jointly exploit cross-channel and spatial correlations.

Minnen et al. [76] introduced a channel-wise autoregressive context model that segments the channels of the latent tensor, coding each segment sequentially with the assistance of previously coded segments. This method reduces the number of sequential steps and surpasses the 2D context model in [10], but it only uses cross-channel correlations, neglecting spatial correlations.

Qian et al. [77] developed a context model that combines 2D masked convolutions with template matching to enhance the receptive field and provide context adaptivity. This method identifies similar patches in previously coded positions and uses the best match as a global reference for the entropy model.

Guo et al. [78] extended the approach in [77] by splitting the channels of the latent tensor into two segments. The first segment is coded using a 2D masked convolution, akin to [76]. The second segment is coded using two mechanisms:

MaskConv+, an enhanced version of 2D masked convolutions, and global prediction. MaskConv+ utilizes spatially co-located elements from the first segment along with local neighbors. The global prediction involves calculating the similarity among all elements in the first segment and selecting the top k similar elements to include in the entropy model.

e) Incorporating Attention Mechanisms and Transformers – Qian et al. [6] replaced the CNN-based hyperprior and context model with a transformer-based one, increasing the adaptivity of the entropy model. They proposed two architectures for their context model: serial and parallel. The serial model processes the latent tensor sequentially, similar to [10], while the parallel model employs a checkerboard grouping strategy from [79] to enhance decoding speed.

A. Burakhan *et al* [80] introduced Contextformer as an alternative to multi-scale context models, the latent processing of the rearranged latent sequence in a spatial and patch-wise manner. Similar to [76], in Multi-reference entropy model for Learned Image Compression (MLIC) [81] proposed Multi-Reference Entropy Model that uses previously decoded slices as context, employing the attention map of these slices to predict global correlations in the current slice. Local context is captured using two enhanced checkerboard context-capturing techniques. In Efficient Learned Image Compression (ELIC) [82], the authors proposed a space-channel context model that Combines their uneven grouping model with existing context models.

2.4. Other Categories of Neural Networks-Based Image Compression Techniques

While most existing NN-based compression techniques rely on the auto-encoder architecture (Section 2.3), other categories of works have been developed by exploiting NN in traditional predictive as well as transform coding schemes. Therefore, we will describe these two kind of methods below.

2.4.1. Neural Networks-Based Transform Coding Schemes

Transform coding schemes have garnered significant attention in the literature. Consequently, numerous research efforts have been focused on enhancing Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) based coding schemes [83–85]. For example, in [83], a CNN architecture is employed to design a DCT-like transform for image compression. In [84], a discrete wavelet transform is applied to the image, and the resulting wavelet subbands are processed by a CNN to generate new high-frequency coefficients. Similarly, Akyazi et al. [85] utilized a discrete wavelet transform on the input image, followed by a convolutional auto-encoder to create a new latent representation. Additionally, Li et al. [86, 87] implemented a lifting scheme-based framework followed by two neural network stages to produce the final wavelet subbands. The first stage is a low-to-high stage that aims to reduce redundancies in the high-frequency subbands to maximize energy compaction, the second high-to-low stage to eliminate aliasing in the low-frequency subbands.

2.4.2. Neural Networks-Based Intra-Prediction Coding Schemes

Intra-prediction is a widely employed technique for efficient image compression that leverages pixel similarities within local neighborhoods by partitioning the image into blocks. Each block is predicted based on its surrounding blocks, with only the residuals being encoded. This approach draws inspiration from video compression standards such as High Efficiency Video Coding (HEVC) [88].

Among the earliest neural network-based intra-prediction schemes, a fully connected network was developed in [23] to learn the mapping of a target block using neighboring reconstructed pixels. Unlike traditional methods that rely on a single reference line, this network utilizes multiple horizontal and vertical reference lines to offer richer contextual information, thereby enhancing prediction accuracy. A dataset of images compressed at various quantization levels is employed to ensure robust generalization across different bit rates. Two training strategies are implemented: the first trains the model on the entire dataset while excluding outliers, and the second trains separate models on angular and non-angular directions, resulting in an improvement in prediction capabilities.

To enhance prediction performance, Hu *et al.* [89] introduced a spatial RNN for intra-prediction. RNNs are stacked orthogonally to model two-dimensional data, improving their ability to capture complex patterns through hierarchical spatial RNNs. Convolutional layers initially map the input image into the feature space, and training is conducted using a Sum of Absolute Transformed Difference (SATD) loss function [90]. The Hadamard transform is applied to the difference between the target and predicted block, allowing the separation of high and low-frequency components during training, which aligns better with the rate-

distortion cost.

Cui *et al.* [91] employed a CNN for intra-prediction, using the three nearest blocks as references rather than a single line. This approach enhances the prediction performance of the target block while refining predictions of neighboring blocks. Dumas *et al.* [92] proposed a dynamic approach that combines CNNs with fully connected networks. FCNNs are found to be optimal for smaller block sizes (8×8 or less), while CNNs perform better for larger blocks with fewer parameters. Inputs for FCNNs are flattened, whereas CNNs maintain a 2D structure, processing top and left contexts through parallel convolutional layers.

Li *et al.* [93] proposed a cross-channel prediction method using a hybrid neural network to reduce redundancies between color image channels. The network take as input the top and left neighboring reference lines along with the co-located luma block in a YUV setting, predicting the chroma block. The model consists of two branches: fully connected layers extract cross-channel information from reference lines, while convolutional layers extract spatial features from the luma block. The outputs of these branches are fused and processed through a convolutional trunk to predict the chroma components. The training utilizes a transform domain loss function that computes the ℓ_1 norm of the transformed residue.

Schiopu *et al.* [94] developed a lossless compression method for photographic images using a CNN pixel-wise predictor. The CNN model is trained on highly textured image regions, enabling the production of high-resolution images. Both the CNN-based prediction paradigm and LOCO-I [95] are considered, with a local entropy descriptor determining the appropriate predictor. Another pixel-wise prediction approach using deep learning was proposed in [96], encoding the residual image with a context-tree-based bit-plane codec. Unlike [94, 96], a block-wise prediction approach using CNN for HEVC lossless compression was developed and validated in [97] and [98].

2.5. Conclusion

After reviewing key concepts related on popular neural network models, this chapter provided an overview of image compression techniques utilizing deep learning. The existing works have been classified into three main categories: (1) Auto-encoders-based coding schemes, (2) NN-based transform coding schemes, and (3) NN-based intra-prediction schemes.

We focused on the auto-encoders-based coding schemes (first category), where the flowchart of the coding architecture with its associated strategy has been described. This category is divided into three aspects: (1) Single rate NN model-based approaches, (2) Variable-rate NN models-based approaches, and (3) Entropy coding/Entropy models.

Our proposed coding approaches, outlined in Chapters 3 and 4, fall within the first category, specifically under single-rate neural network model-based approaches (first aspect).

3. Convolutional Transformer-Based Image Compression

Contents

Summary of this Chapter

In this chapter, we address the image compression challenge and introduce the Swin Non-Positional Encoding (SwinNPE) transformer.

SwinNPE improves the efficiency of the SwinT transformer while reducing the number of model parameters. We generalize the Swin cell and propose the Swin convolutional block, which can better handle the local correlation between image patches. Additionally, the Swin convolutional block can capture the local context between tokens without relying on positional encoding, thereby reducing the model complexity. Our results show that SwinNPE outperforms state-of-the-art CNN-based architectures in terms of the trade-off between bit-rate and distortion, achieving results comparable to SwinT with 16% less computational complexity on the Kodak dataset. These contributions were published in *IEEE International on Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA) 2023* and *22ème édition de la conférence Compression et Représentation des Signaux Audiovisuels (CORESA) 2023*, further validating the effectiveness of SwinNPE in advancing the field of image compression.

3.1. Introduction

In the context of image compression, the advantages of positional encoding have been illustrated through improved R-D performance in works such as [6,7]. Specifically, the authors of [6] demonstrated that a 2D diamond-shaped relative position encoding is particularly effective and offers distinct benefits. Despite its numerous advantages, incorporating positional encoding in transformers can increase the dimensionality of embeddings, leading to higher computational costs during training and limiting the flexibility of the models. Recently, the authors of [99] showed that positional encoding can be omitted in the attention module for image classification without sacrificing performance. This was achieved by incorporating convolution in the tokenization process of patches and within the self-attention block, thereby maintaining local spatial information. It is asserted that this combination of convolution and the attention mechanism leverages the strengths of both the advantages of convolutional neural networks and transformers.

In this chapter, we present a new image compression framework called SwinNPE. It is based on our proposed convolutional Swin block, which combines patch convolution and shift window-based attention in Swin without positional encoding. We believe this approach enhances the capture of spatial contextual information. Our experiments show that SwinNPE achieves comparable results to the SwinT architecture [7], without the need for positional encoding and with fewer parameters. The rest of this chapter is structured as follows. Positional Encoding in the Context of Image Compression is discussed in Section 3.2, Section 3.3 presents the proposed Swin Non-Positional Encoding (SwinNPE). Section 3.4 illustrates the experimental results. Finally, Section 3.5 provides conclusions and perspectives.

3.2. Positional Encoding in Image Compression

As emphasized in Chapter 2, Positional encoding plays a crucial role in transformer models, which cannot inherently capture the positional relationships between input tokens. Since transformers use self-attention mechanisms, which treat inputs as unordered sets, they are inherently position-agnostic. This makes positional encoding essential for providing the model with information about the relative positions of tokens, which is especially important for capturing sequential and spatial patterns in data.

a) 1D Positional Encoding for Sequential Data – Originally introduced in the context of sequence-based tasks, such as natural language processing, positional encoding was designed to inject positional information into transformer models for 1D data. In such tasks, positional encoding enables the model to differentiate between different positions in a sequence, making it capable of recognizing order and dependencies over time or across sequences (more details are provided in chapter 2).

b) Extending Positional Encoding to 2D for Image Compression – In image compression, where the data is inherently two-dimensional, transformer models face a similar challenge. The spatial grid structure of images introduces the need for the model to not only understand local texture patterns but also capture broader spatial relationships across the image. Without explicit positional information, transformers struggle to effectively model these spatial relationships, limiting their potential for image compression.

To address this challenge, positional encoding is adapted to 2D grids in image data. This adaptation enables the model to grasp the spatial coherence between different patches or regions of an image. In the context of compression, this un-

derstanding is vital as it allows the model to exploit both local and global redundancies in the image, thereby improving compression efficiency while preserving important image details.

c) Approaches for 2D Positional Encoding in Transformers – In the context of images, positional encoding can be implemented in two main ways:

- Concatenating Horizontal and Vertical Encodings: Separate encodings for the horizontal and vertical positions are concatenated to provide the model with information about each patch's position within the 2D grid.

- Learnable 2D Positional Embeddings: the model learns a set of positional embeddings based on the spatial locations of the patches within the image. This learnable approach has become the most commonly used in transformer-based image compression models. Studies such as [6] and [7] have shown that learnable 2D positional embeddings improve the transformer's ability to comprehend spatial dependencies across the image.

For example, in [6], the authors enhanced the traditional positional encoding [36] by incorporating a diamond-shaped 2D boundary to represent spatial relationships, highlighting that closer context patches contribute more heavily to compression performance.

d) Computational Complexity of Positional Encoding – However, despite the advantages of 2D positional encoding in improving compression performance, it comes at a cost. Transformer models, especially when dealing with high-resolution images, require significant memory and computational resources. The introduction of positional encoding further increases this complexity, as the model must account for the positional relationships between a large number of patches in the image grid.

Typically, transformers for image compression divide an image into patches and then apply positional encoding to inform the model of each patch's location. While this approach helps the model effectively capture both local and global features, the increased complexity can be a limiting factor in practice, especially for real-time applications or when resources are constrained.

e) Recent Advances: Replacing Positional Encoding with Convolutions – To address the computational challenges associated with traditional positional encoding, recent research has explored alternative methods. One promising direction involves substituting positional encodings with convolutional layers. Convolutions inherently capture spatial locality due to their sliding window mechanism, allowing the model to learn positional relationships without the overhead of explicit positional encodings (refer to Chapter 2 for more details about the convolution operation).

For instance, in [99], convolutional layers were introduced to replace positional encodings in the classification transformer model. This approach significantly reduces the computational load while still enabling the model to capture spatial dependencies.

f) Applying Convolution-Based Positional Encoding to Image Compression – Although the replacement of positional encoding with convolutions has primarily been explored in fields such as natural language processing and vision transformers, our work proposes to adapt this approach specifically for image compression. By integrating convolutional layers into transformer-based compression models, we aim to reduce computational complexity while maintaining strong compression performance. This shift allows us to maintain the spatial awareness required for effective compression, without the computational overhead traditionally associated with positional encoding.

3.3. Proposed Swin Non-Positional Encoding (Swin-NPE)

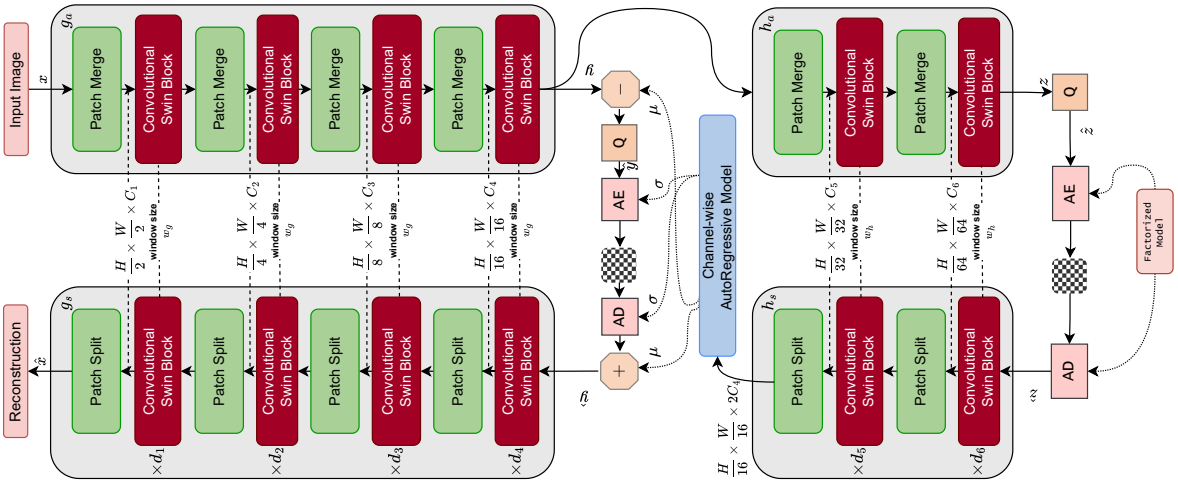


Figure 3.1: Network architecture of our proposed SwinNPE.

The proposed SwinNPE follows the same architecture described in Section 2.3.1, as depicted in Figure 3.1. This architecture is composed of an encoder-decoder structure and an entropy model. Specifically, the input image x is first encoded by the generative encoder $y = g_a(x)$, and the hyper-latent $z = h_a(y)$ is obtained. The quantized version of the hyper-latent $\hat{z}$ is modeled and entropy-coded with a learned factorized prior to passe through $h_s(\hat{z})$ to obtain μ and σ which are the parameters of a factorized Gaussian distribution $P(y|\hat{z}) = \mathcal{N}(\mu, diag(\sigma))$ to model y. The quantized latent $\hat{y} = Q(y - \mu) + \mu$ is finally entropy-coded and sent to $\hat{x} = g_s(\hat{y})$ to reconstruct the image $\hat{x}$.

We choose the loss function to optimize the trade-off between the bit-rate R and the quality of the reconstruction D (all the details of the loss function have been provided in chapter 2 specifically in Section 2.3.1.2):

$$L = D(x, \hat{x}) + \beta R, \tag{3.1}$$

The rate is calculated as the negative log probability of the latent representation given its estimated distribution with the learned factorized prior for z latent space and channel-wise autoregressive model for y (for more details refer to chapter 2 about the Hyperprior):

$$R = -\sum_{i} \log_2 \left(P(y_i) \right), \tag{3.2}$$

$P(y_i)$ represnts the probability of the i element of the latent space y.

The distortion D is defined as the Mean Squared Error (MSE) between the original image x and the reconstructed image $\hat{x}$ in Red Green and Blue (RGB) color space:

$$D(x, \hat{x}) = \frac{1}{n \times m \times 3} \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{c=1}^{3} \left(x_{i,j,c} - \hat{x}_{i,j,c} \right)^2, \tag{3.3}$$

$I_{i,j,c}$ and $\hat{I}_{i,j,c}$ denote the pixel value at position (i, j) in the channel c (RGB) of the original and constructed images, respectively.

3.3.1. SwinNPE Encoder

The SwinNPE encoder is composed of both a generative encoder g_a, and a hyper-prior encoder h_a, each constructed using a combination of the patch merge block, and the proposed Convolutional Window-based Multi-head Self-Attention (CW-MSA).

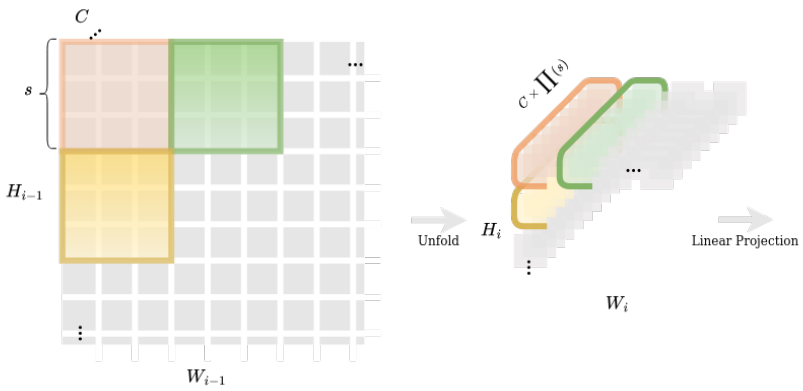


Figure 3.2: Patch Merge block.

3.3.1.1. Patch Merge Block

Following [7], the patch merge block contains the *Depth-to-Space* operation [7] operation for down-sampling, as shown in Figure 3.2. This block also incorporates a normalization layer and a linear layer to project the input to a specified depth C_i. In g_a, the depth C_i of the latent representation increases as the network gets deeper which allows for getting a more abstract representation of the image. The size of the latent representation decreases accordingly. In each stage, we down-sample the input feature by a factor of 2 using a stride of 2 and kernel size s in the *Depth-to-Space* operation [7].

3.3.1.2. Proposed CW-MSA Block

Before introducing the CW-MSA block, we first provide an overview of the standard Window-based Multi-head Self-Attention (W-MSA) block in the following subsection.

3.3.1.2.1 Standard W-MSA Swin Block

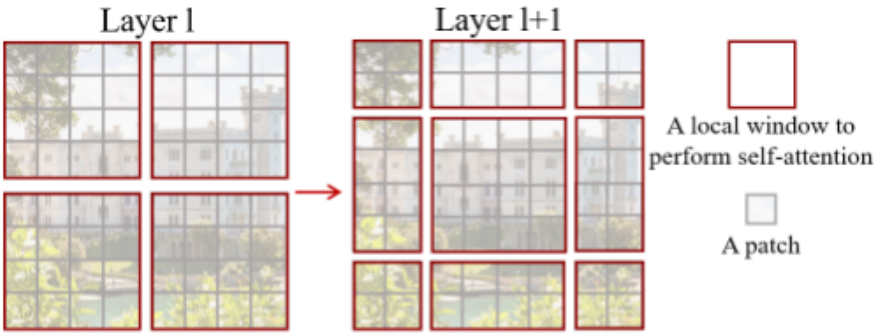
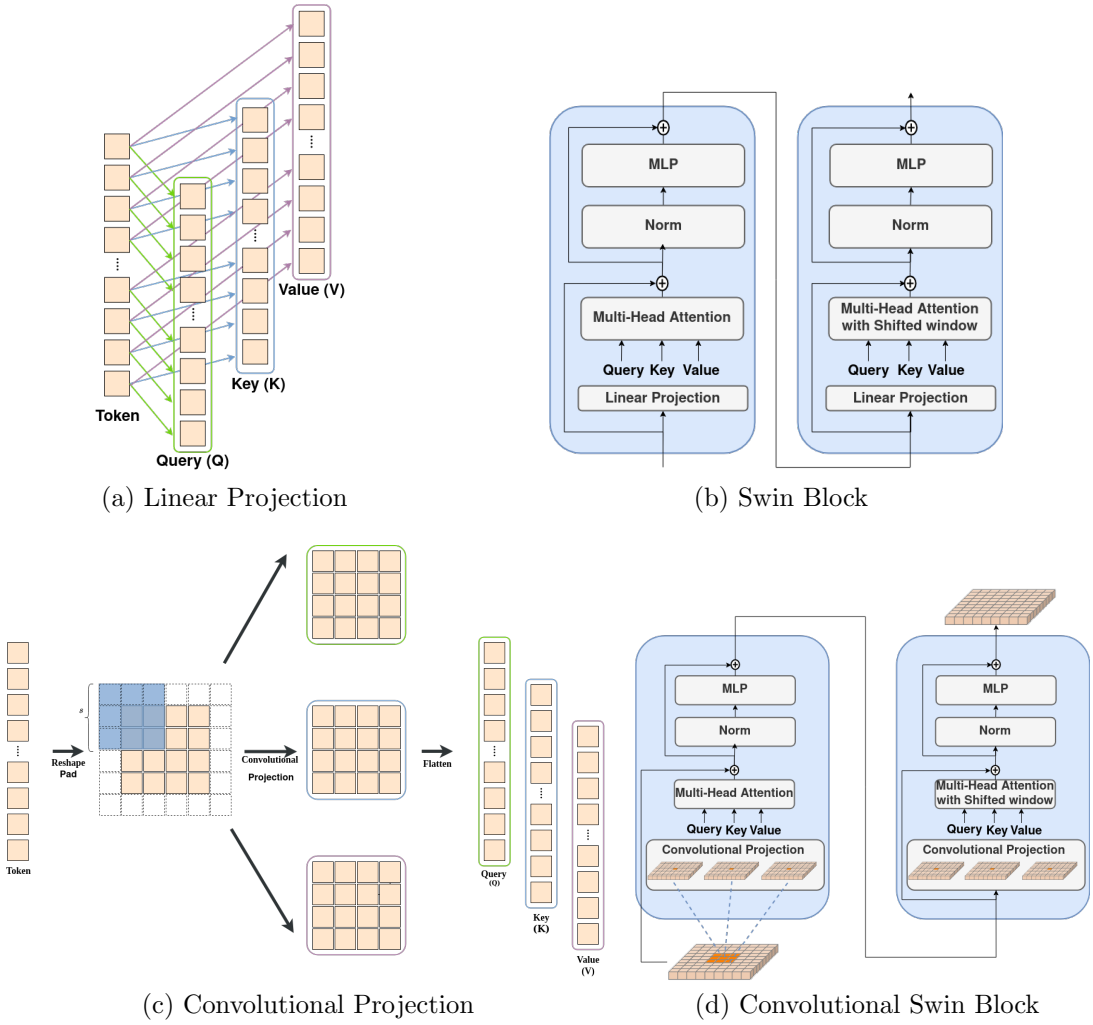


Figure 3.3: Shifted window approach in Swin block (schemes taken from [5]).

Swin block [5] is a window-based multi-head self-attention block in which the input image is partitioned into non-overlapping fixed-size windows (w_g, w_h for generative and Hyperprior part respectively in swinNPE, as shown in Figure 3.1). Unlike the global self-attention mechanism in standard Transformers presented in Section 2.2.4, which computes attention across all tokens, the W-MSA restricts attention to patches within each window. This approach significantly reduces the computational complexity while retaining local context.

The Shifted Window Multi-head Self-Attention (SW-MSA) mechanism of Swin block is illustrated in Figure 3.3. In layer l (left), a regular window partitioning scheme is applied, where self-attention is computed independently within each window. In the subsequent layer $l+1$ (right), the partitioning is shifted, creating a new set of windows. This shift allows the self-attention in layer $l+1$ to span across the boundaries of the previous windows from layer l, thereby facilitating connections between them and improving global context aggregation.



(a) Linear Projection (b) Swin Block

(c) Convolutional Projection (d) Convolutional Swin Block

Figure 3.4: Architecture of the proposed Convolutional Swin Block.

3.3.1.2.2 Convolutional Swin Block

The proposed *convolutional Swin block* generalizes the Swin cell [5]. As illustrated in Figure 3.4, we replace the position-wise linear projection (see Figure 3.4(a)) with a convolutional projection (see Figure 3.4(c)) to generate the Key (K), Query (Q), and Value (V) matrices within the attention block (for more details on the attention block, refer to Section 2.2.4).

Specifically, as depicted in Figure 3.4(c) and following [99], we first reshape the tokens into a 2D format before applying the convolutional projection. Afterward, the tokens are flattened to obtain the Key, Query, and Value representations for use in the Swin block [5] instead of the MHSA block:

$$K, Q, V = \text{Flatten}(\text{Conv2d}(\text{Reshape2D}(x))). \tag{3.4}$$

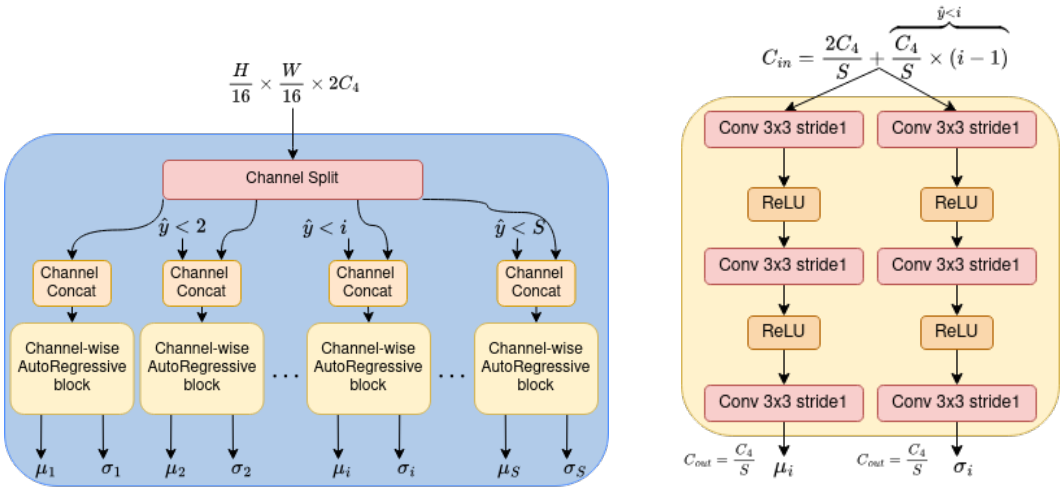
This makes the attention module more sensitive to spatial context. Instead of relying on hand-crafted positional encodings, we enable the convolution layer to capture the positional information directly.

We use depth-wise separable convolution [99] due to its parameter efficiency. More specifically. Specifically, the depth-wise separable convolution first applies a 2D convolution independently to each feature channel. The outcome is then concatenated and passed through another convolution layer. This approach reduces both the number of parameters and computational complexity while increasing representational efficiency, as it addresses not only the spatial dimension but also the depth dimension. It is important to note that the proposed block is not limited to convolution operations; different forms of convolution [100, 101] can be utilized, making the proposed convolutional Swin block particularly flexible.

3.3.2. SwinNPE Decoder

The SwinNPE decoder is composed of both a generative decoder g_s, and a hyperprior decoder h_s, each constructed using the patch split block and the convolutional Swin block. In the patch split block, we reverse the merging sequence of the patch merge block and use *Space-to-Depth* operation [7] for up-sampling.

3.3.3. Quantization and Entropy Model in SwinNPE



(a) Channel-wise AutoRegressive Model

(b) Channel-wise AutoRegressive block

Figure 3.5: The architecture of the Channel-wise AutoRegressive Model.

We use the classical strategy of adding uniform noise to simulate the quantization operation, rendering it differentiable. The channel-wise autoregressive block [76] is designed to learn an auto-regressive prior, factorizing the distribution of the latent as a product of conditional distributions that incorporate predictions from the causal context of the latents [10, 61, 76].

Figure 3.5 illustrates the architecture of a channel-wise autoregressive model used for processing latent representations in SwinNPE. The model is designed to predict and decode the latent tensor components sequentially, leveraging channel-wise autoregressive dependencies. The input feature (output of h_s) is first split into S slices, which are then processed sequentially by a series of channel-wise autoregressive blocks. Each block predicts the parameters μ_i and σ_i for the re-

spective slice y_i. During this processing, previously reconstructed slices $\hat{y} < i$ are concatenated with y_i along the channel dimension channel dimension. As a result, the channel dimension becomes $C_{in} = \frac{2C_4}{S} + \overbrace{\frac{C_4}{S} \times (i-1)}^{\hat{y}<i}$, illustrating how the context expands as more channels are processed.

The structure ensures that each slice can only be decoded after the preceding slice has been processed and its parameters μ_i and σ_i are available. This sequential execution is crucial for maintaining the autoregressive property of the model.

Each Channel-wise autoregressive block consists of several layers of convolutional operations, followed by ReLU activations. Specifically, there are Conv 3×3 stride 1 layers, indicating convolution operations with a kernel size of 3×3 and a stride of 1. The output of these convolutions is used to estimate the conditional mean μ_i and standard deviation σ_i for the subsequent latent space entropy coding. The output channel dimension is given by $C_{out} = \frac{C_4}{S}$ reflecting the contribution of each block to the final representation for μ_i and σ_i.

3.4. Experimental Results

This section evaluates the SwinNPE architecture and compares its image compression results to the state-of-the-art approaches. In this respect, after describing the experimental settings, we assess various End-to-End Neural network model-based approaches in terms of compression performance and complexity.

3.4.1. Experimental Settings

The SwinNPE is trained on the CLIC2020 training set for 3.3 million steps. During training, each batch consists of eight randomly cropped images with a

size of 256×256 pixels. The SwinNPE's performance is evaluated on the Kodak and JPEG-AI test dataset [11, 12]. All images are center-cropped to multiples of 256 to avoid padding. We selected Lagrange multiplier values β from the set $\{0.003, 0.001, 0.0003, 0.0001\}$ for the loss function in Equation 3.1.

ADAM algorithm is used with a learning rate equal to 10^{-4} and the hyper-parameters of the architecture shown in Figure 3.1 are as follows: $(d_1, d_2, d_3, d_4, d_5, d_6) = (2, 2, 6, 2, 5, 1)$, $(w_g, w_g) = (8, 8)$, $(w_h, w_h) = (4, 4)$, and $(C_1, C_2, C_3, C_4, C_5, C_6) = (128, 192, 256, 320, 192, 192)$. For the autoregressive model, we use the model proposed in [76], as presented in Figure 3.5 with $S = 10$ slices. The kernel size for all convolutional Swin blocks using depth-wise separable convolution is set to 3.

These implementations were carried out by using Keras and TensorFlow on an NVIDIA A100 40 Go Graphics Processing Unit (GPU) and an Intel Xeon Gold 6330 3.10 GigaHertz (GHz) Central Processing Unit (CPU).

3.4.2. Performance Analysis

We compare our proposed SwinNPE model with the results of two transformers-based architectures [6, 7] and some of the most used CNN-based image compression architectures and standard codecs. The results of SwinT [7] and Entroformer [6] are obtained from their respective papers.

The rate-distortion curves of the different methods are illustrated in Figure 3.6 and Figure 3.7 on the Kodak dataset [11] and JPEG-AI test-set [12] respectively. In these two figures, the Peak Signal to Noise Ratio (PSNR) and the rate values represent the averages across all images of their respective datasets.

We summarize the number of parameters and Giga Multiply-ACcumulate oper-

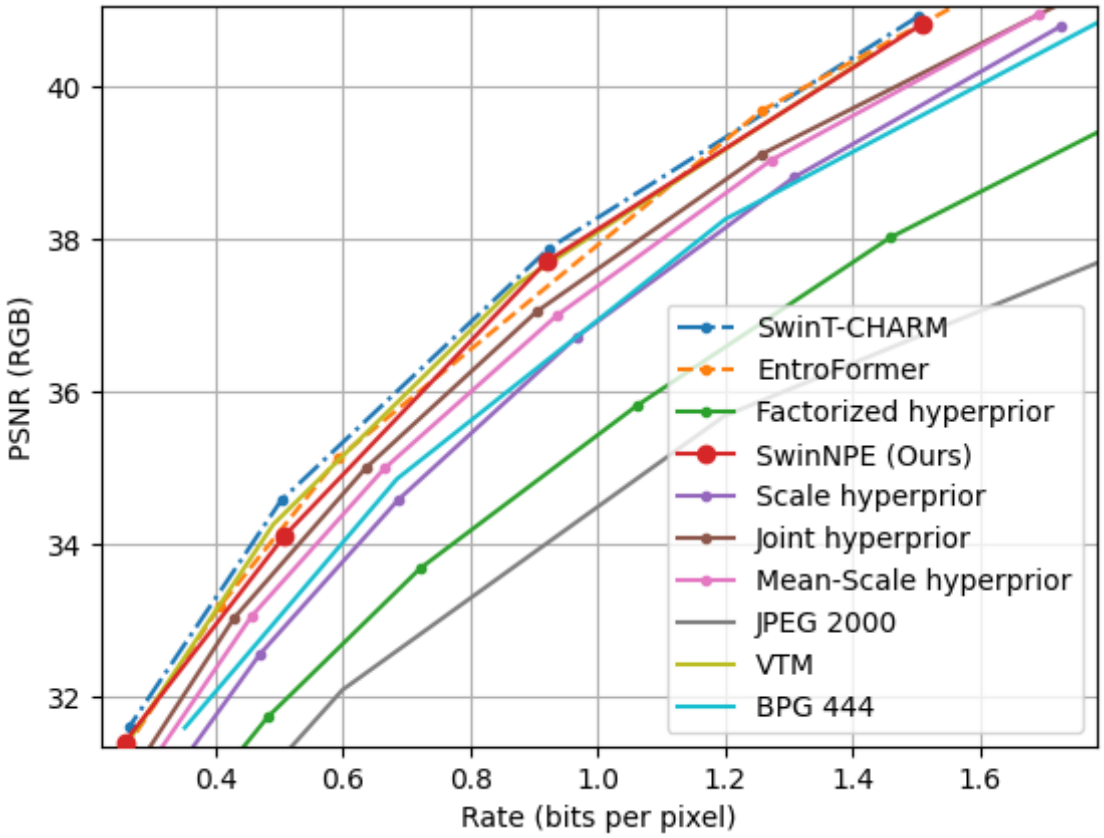


Figure 3.6: SwinNPE achieves nearly the same results as Entroformer [6] and SwinT-CHARM [7] that relying on Positional encoding and better RD performance than CNNs-based methods Factorized [8], Scale [9], Mean-Scale [10], Joint Hyperprior [10] and standard codecs on the Kodak [11] image set.

ations per second (GMACs) [1] for the tested transformer-based architectures in Table 3.1 where we also illustrate the Bjontegaard Delta-Rate (BD-rate), using the SwinT-CHARM as the reference for the Kodak dataset [11].

From Figure 3.6, we can clearly see that the SwinNPE outperforms all of the tested CNN-based architectures in terms of the bit-rate/distortion tradeoff. It

[1]For SwinNPE, the FLoating-point Operations Per second (FLOPs) were computed using TensorFlow. The Multiply-ACcumulate operations per second (MACs) for SwinNPE were estimated by dividing the FLOPs by 2, Based on the standard assumption that, in many models, each multiply-accumulate operation corresponds to two floating-point operations.

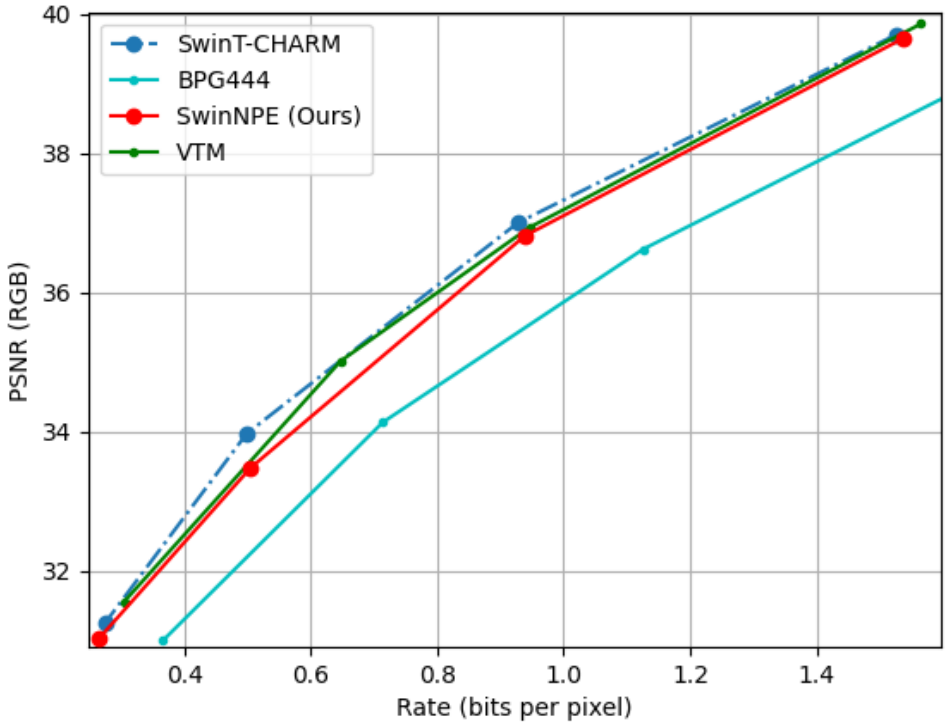


Figure 3.7: SwinNPE achieves nearly the same results as SwinT-CHARM [7] and better RD performance than standard codecs on the JPEG-AI test-set [12].

is particularly interesting to notice that our proposed approach achieves almost the same results to those of Entroformer [6] (orange dashed line in Figure 3.6) while using significantly fewer model parameters (see Table 3.1). Specifically, SwinNPE demonstrates a bit-rate saving of 5.46% compared to SwinT-CHARM, which provides the optimal saving bit-rate saving, bringing it close to the performance of Entroformer, resulting in only a 1.13 Percentage Point (p.p.) difference from Entroformer.

We argue that this performance can be attributed to the convolutional layers in the proposed convolutional Swin block, which effectively capture the local contextual information. As demonstrated in [6], closer contextual information has

a greater impact on bitrate efficiency. Additionally, SwinNPE reduces correlations among direct neighboring latent representations, thereby enhancing compression efficiency. More details on this process are provided in the following subsection.

From Figure 3.6 and Figure 3.7, we can observe that the proposed SwinNPE achieves results comparable to SwinT-CHARM across both datasets, despite having fewer parameters. We emphasize that our proposed architecture offers significant advantages over SwinT-based architecture without positional encoding [2] highlighting the benefits of combining convolutions and transformers for image compression.

Method	#parameters (M)	GMACs	Positional Encoding (PE)	BD-rate
SwinT-CHARM* [7]	32	223	PE Relative 2D	0%
Entroforme* [6]	142.7	-	PE Relative 2D	4.33%
SwinNPE	**27**	**178**	-	**5.46%**

Table 3.1: BD-rate performance using SwinT-CHARM [7] as reference on Kodak [11]. We use $*$ to indicate the approaches for which the numbers are sourced from their respective papers.

Figure 3.8 provides the R-D curves on each image of the Kodak dataset [11]. Each curve presents the evaluation of an image using different versions of β value (i.e. different SwinNPE models). As expected, experiments with the same β values reveal variability in PSNR and rate values across different images. From the curves, a significant difference can be seen between images, highlighting the substantial discrepancy in results when using the same approach on different images. This observation highlights the dependence of the results on the individual characteristics of the images in the dataset.

[2] The results are detailed in the ablation studies in [7].

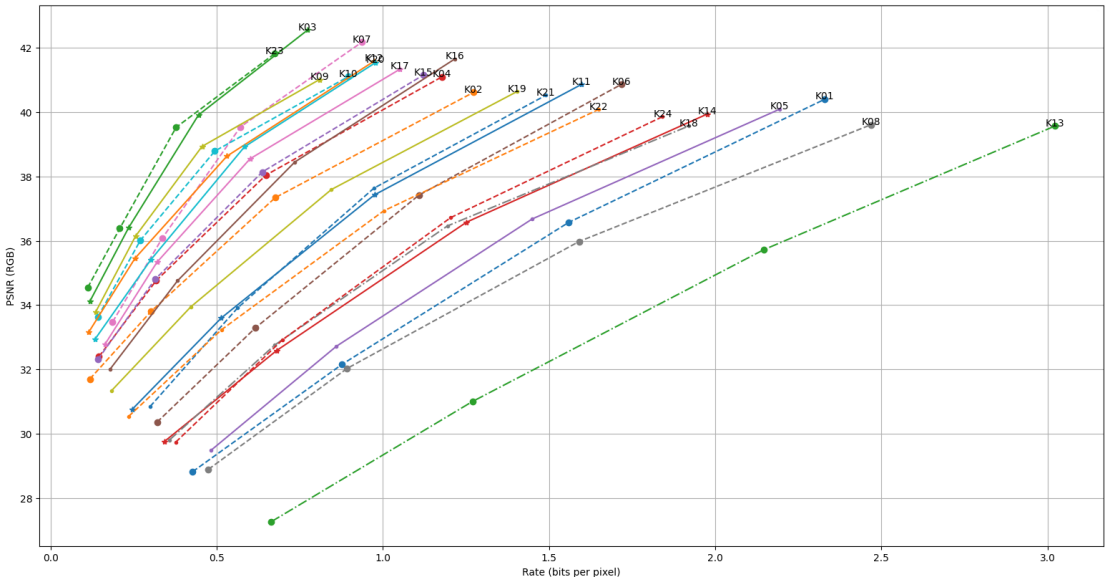


Figure 3.8: Rate-Distortion (RD) performance of SwinNPE on each image of Kodak [11] dataset.

3.4.3. Latent Space Analysis

Transform coding is driven by the principle that encoding becomes more efficient in the transform domain compared to the original signal space. An ideal transform would decorrelate the source signal, enabling the use of simple scalar quantization and a factorized entropy model without sacrificing coding performance. Additionally, an effective prior model would introduce context adaptivity, leveraging distant spatial relationships within the latent representation to further enhance the compression process. This approach optimizes the trade-off between complexity and performance while maintaining high coding efficiency.

The effectiveness of the analysis transform g_a can be assessed by examining the level of correlation in the latent signal y. Specifically, we are interested in evaluating the correlation between nearby spatial positions, which tend to be highly correlated in the source domain for natural images. In Figure 3.9, we

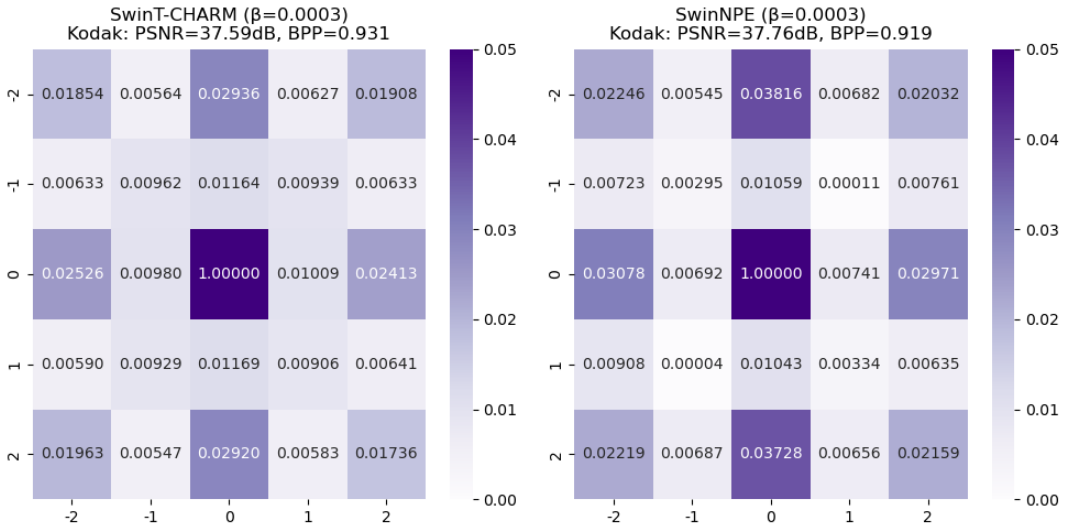


Figure 3.9: The average of spatial correlation of all images on Kodak [11]. SwinNPE (right) achieves smaller correlation than SwinT-CHARM in direct neighboring spatial positions (left).

present a visualization of the normalized spatial correlation of y, averaged across all latent channels, comparing SwinT-ChARM[3], and the proposed SwinNPE at $\lambda = 0.0003$.

The spatial correlation in Figure 3.9 at index (i, j) corresponds to the normalized cross-correlation of latents $(y-\mu)/\sigma$ at spatial location (w, h) and $(w+i, h+j)$, averaged across all latent elements of all images on Kodak [11].

The results indicate that, although all approaches yield low cross-correlation, SwinNPE demonstrates a slight improvement in decorrelating the latent signal in direct neighboring spatial positions compared to SwinT-ChARM[3], which has a significant impact on bitrate efficiency [6]. This suggests that combining Transformer-based transforms with convolutional-based transforms introduces less redundancy across spatial latent positions, enabling more efficient encoding with lower bitrate costs than using Transformer-based transforms alone.

This efficiency is evident in Figure 3.10, which shows that SwinNPE achieves

lower bitrate costs compared to SwinT-CHARM [3] while maintaining the same reconstruction quality in terms of PSNR. In Figure 3.11, for the same bitrate, SwinNPE achieves better reconstruction quality (PSNR) than SwinT-CHARM [3]. Furthermore, SwinNPE has fewer parameters and lower complexity compared to SwinT-CHARM [3], as outlined in Table 3.1. Additional details on the spatial correlation for various λ versions of SwinNPE can be found in Appendix A.

3.4.4. Ablation Study

Method	Positional Embedding (PE)	Linear Projection (LP)	Conv (Kernel Size)	Estimated Training Time (3M steps)
SwinNPE	✗	✗	3x3 (Conv)	1 month, 20 days for λ_i
SwinNPE with LP (SwinT [7] without PE)	✗	✓ (Linear)	✗	1 month, 20 days for λ_i
SwinNPE with PE	✓	✗	3x3 (Conv)	1 month, 20 days for λ_i
SwinNPE with PE and LP (SwinT [7])	✓	✓ (Linear)	✗	1 month, 20 days for λ_i
SwinNPE variant	✗	✗	5x5 (Conv)	1 month, 20 days for λ_i
SwinNPE variant	✗	✗	8x8 (Conv)	1 month, 20 days for λ_i

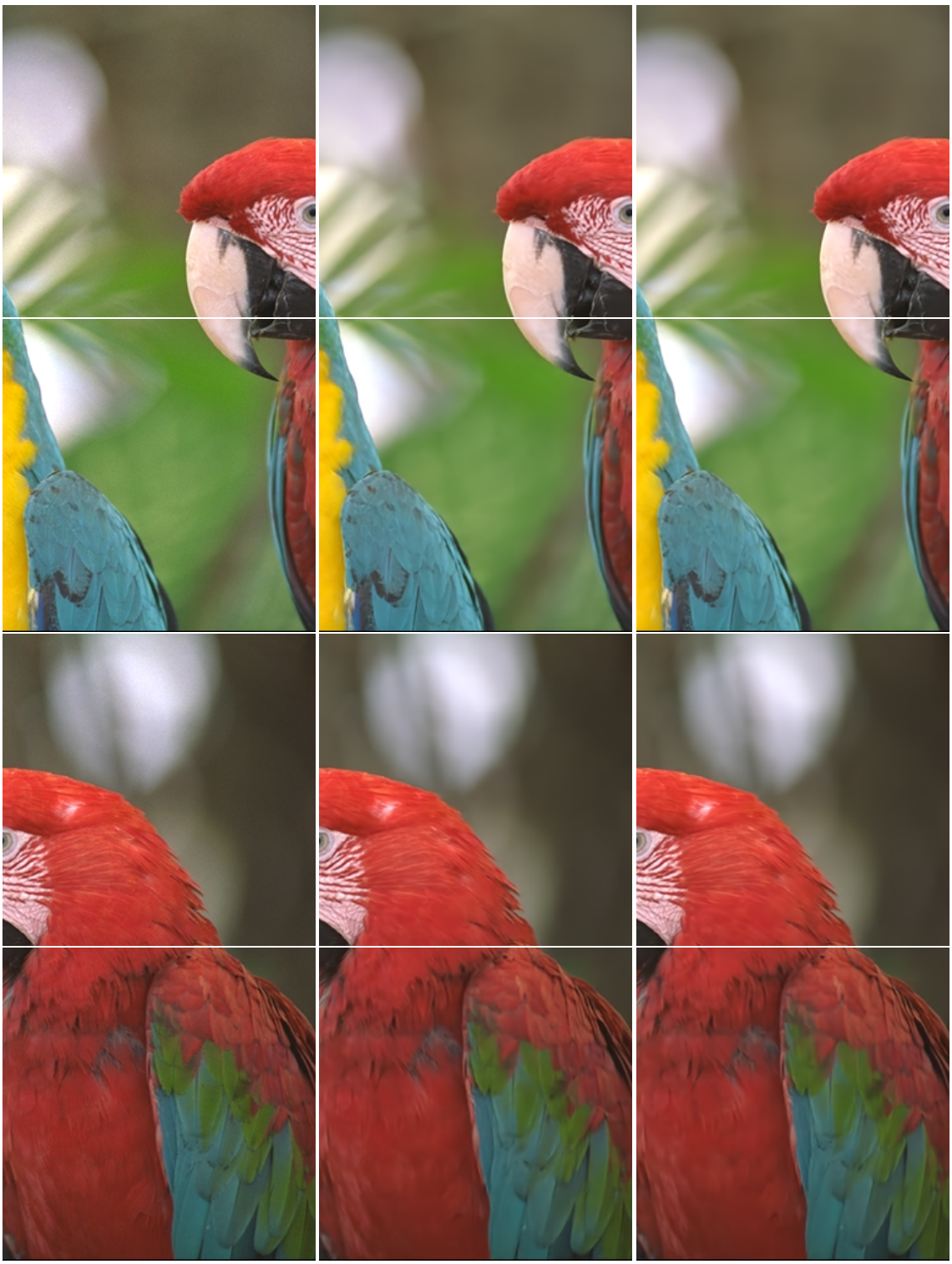
Table 3.2: Time Required for a Comprehensive Ablation Study.

While we aimed to conduct comprehensive ablation studies to evaluate the individual contributions of the components in our proposed framework, we encountered significant constraints due to limited computational resources. Such studies

[3]For the latent analysis, we retrained SwinT-CHARM using the publicly available code from this repository `https://github.com/Nikolai10/SwinT-ChARM/tree/master`. The retrained model produced results closely matching those reported in the original paper, ensuring the validity of our comparisons.

would have allowed us to isolate and analyze the effects of key elements, including the convolutional Swin block and positional encoding.

Table 3.2 provides a summary of the ablations we aimed to investigate to analyze the impact of each element. The last column indicates the estimated time required to train a single model for a specific point on the rate-distortion curve with λ_i. However, the substantial computational cost associated with training multiple model variants posed a significant challenge. Despite these challenges, we believe the results presented still provide strong evidence of the framework's effectiveness.



Ground Truth	SwinT	SwinNPE
Kodim23.png	[0.395/39.56/20.13]	[0.378/39.52/20.18]

Figure 3.10: Visualization of the reconstructed images from the Kodak dataset*"Kodim23"*. The metrics of SwinNPE [↓ bpp/≈ PSNR(dB)/↑ MS-SSIM(dB)] compared to those of SwinT, while SwinNPE employs fewer parameters and exhibits reduced complexity in comparison to SwinT.

Ground Truth	SwinT	SwinNPE
Kodim01.png	[1.560/36.43/22.31]	[1.558/36.57/22.27]

Figure 3.11: Visualization of the reconstructed images from the Kodak dataset*"Kodim01"*. The metrics of SwinNPE [≈ bpp/↑ PSNR(dB)/↓ MS-SSIM(dB)] compared to those of SwinT, while SwinNPE employs fewer parameters and exhibits reduced complexity in comparison to SwinT.

3.5. Conclusion

In this chapter, we propose SwinNPE, a transformer-based image compression model built with convolutional Swin blocks without positional encoding. SwinNPE achieves comparable results to state-of-the-art methods while using fewer model parameters and outperforming CNN-based architectures. The proposed convolutional Swin block allows for better exploitation of spatial context without the need for positional encoding, resulting in greater flexibility and fewer parameters.

While it would be interesting to conduct ablation studies with different convolution sizes to further enhance spatial context, we were limited by computational resources. This constraint motivated us to focus on reducing the model's complexity. For future work, exploring alternative neural network architectures to replace the attention mechanism could be a promising direction to further lower model complexity while maintaining performance. This will be the focus of the next chapter.

4. Efficient Image Compression Using Advanced State Space Models

Contents

Summary of this Chapter

Transformers have revolutionized learning-based image compression methods, often outperforming traditional approaches. However, these methods frequently suffer from high complexity, limiting their practical applicability. To address this issue, various strategies such as knowledge distillation and lightweight architectures have been explored, to enhance efficiency without significantly sacrificing performance. This chapter proposes a State Space Model-based Image Compression (SSMIC) architecture. This novel architecture balances performance and computational efficiency, making it suitable for real-world applications. Experimental evaluations confirm the effectiveness of our model in achieving a superior BD-rate while significantly reducing computational complexity and latency compared to competitive learning-based image compression methods. These results were presented at *IEEE 26th International Workshop on Multimedia Signal Processing (MMSP) 2024* and *23ème édition de la conférence Compression et Représentation des Signaux Audiovisuels (CORESA) 2024*.

4.1. Introduction

Deep learning-based compression approaches often face high complexity, which limits their practicality in real-world applications. While it is common practice to propose scaled-down models with reduced sizes, these models, although less complex and faster, frequently suffer from significant reductions in compression performance [7]. Other approaches have been deployed to accelerate these models while maintaining their performance. Knowledge distillation is one effective method for accelerating neural networks across various computer vision tasks [102].

In this context, Efficient single-model Variable-bit-rate Codec (EVC) [103] leverages mask decay in a teacher-student training framework. The core architecture replaces traditional transformers with depth-wise convolution blocks to build a more efficient model. The mask decay technique introduces mask layers between the convolutional layers of a large teacher model to gradually sparsify the parameters. This leads to the automatic transformation of the larger model into a smaller student model. These mask layers, by applying a novel sparsity regularization loss during training, prune unnecessary channels. As the mask parameters become sparser, the teacher model is compressed into a more efficient structure while retaining the knowledge gained from the larger model, enhancing the performance of the smaller model without the need for training from scratch. Despite its advantages, there are challenges associated with mask decay, including computational overhead during training, as the process involves inserting additional mask layers and optimizing them, adding complexity compared to training from scratch.

Another challenge lies in balancing sparsification with performance. The mask

decay process must effectively sparsify the teacher model's parameters to align with the smaller student architecture while still maintaining good compression performance, which can be difficult to achieve. Additionally, there is a risk of degrading model accuracy during the pruning process, making it challenging to find the optimal balance between sparsification and overall model performance.

Another promising direction involves the development of lightweight architectures [104, 105]. Recent advancements in lightweight attention mechanisms have resulted in architectures that offer improved inference speeds for image compression [74].

In [74], Residual connected Lightweight Attention Units (RLAUs) are introduced to efficiently extract global spatial information while minimizing complexity. This is achieved by simplifying attention mechanisms and incorporating residual connections. The Lightweight Attention Module streamlines the typical self-attention mechanism by ensuring that the Query and Key projection matrices are of the same size, thereby reducing the overall number of parameters. RLAUs allow the model to capture global spatial dependencies efficiently without the heavy computational load associated with standard self-attention mechanisms. Additionally, the Channel-Gained Adaptive Module (CGAM) is used to dynamically adjust the importance of different channels in the latent space, enhancing better compression efficiency.

Another significant contribution is found in [106], where the authors focus on reducing the decoding complexity by introducing shallow decoding transforms. This approach replaces traditional deep convolutional decoders with lightweight, linear transforms. Drawing on the insight that decoder transforms in neural compression function similarly to orthogonal transforms in traditional coding, the authors developed a JPEG-like synthesis transform. This method divides the latent tensor into blocks and applies efficient block-wise linear transforms, thereby

reducing computational complexity while enabling end-to-end optimization. To counterbalance potential performance losses from a shallow decoder, the technique exploits asymmetrical computational budgets by pairing a lightweight decoder with a robust heavy encoder.

The works in [79, 107] aim to enhance the efficiency of entropy models. Although AR context-based entropy models effectively capture spatial dependencies in image latent, their sequential processing results in significant computational complexity, limiting parallelization. To overcome this limitation, He et al. [79] propose a parallelized Checkerboard Context Model (CCM). This model reorganizes the decoding order, thereby removing the spatial location constraints imposed by AutoRegressive (AR) models. Ali et al. [107] introduce a correlation loss that forces spatial decorrelation of latent, achieving an improved balance between performance and complexity without depending on AR context models.

Despite these advancements, many approaches still face significant performance degradation when attempting to lower computational complexity, especially in high-resolution contexts. This challenge prompted us to explore alternative solutions that effectively reduce computational demands while ensuring competitive performance against state-of-the-art methods.

Recently, the SSMs and their variant, the Mamba model, have gained significant attention in the field of computer vision. Originally introduced for sequence modeling [108] in deep neural networks, SSMs unify the strengths of previous sequence models, including Continuous-Time Models (CTMs), RNNs, and CNNs (for more details about SSMs, CNNs, and RNNs, refer to Chapter 2 Section 2.2). Despite their potential, SSMs have not been widely adopted due to their high computational and memory requirements associated with the latent state representations, connecting the input and output. The Mamba model [37] addresses these limitations by integrating a selection mechanism into the structured SSMs

variants, thereby enhancing context-based reasoning ability. In this chapter, we introduce a State Space Model-based Image Compression (SSMIC) architecture, which emphasizes optimizing rate-distortion performance while minimizing computational complexity and latency.

The remainder of this chapter is organized as follows. Section 4.2 details the proposed SSMIC. Following that, Section 4.4 presents and discusses the experimental results. Finally, Section 4.5 concludes with insights and future perspectives.

4.2. Proposed State Space Model-based Image Compression (SSMIC)

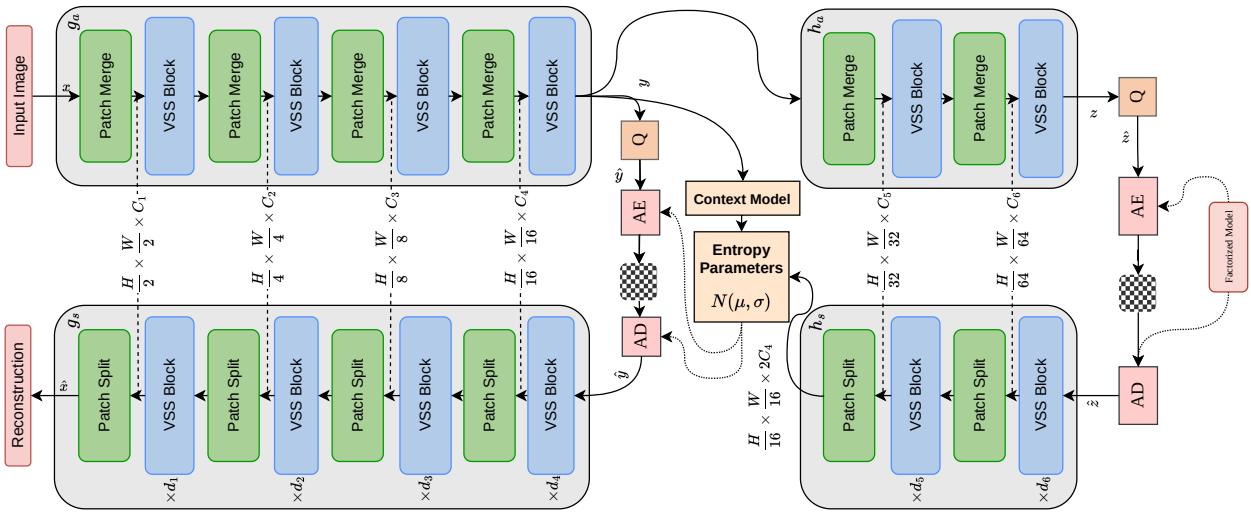


Figure 4.1: Network architecture of our proposed SSMIC model.

The proposed SSMIC architecture follows the same architecture described in Sec-

tion 2.3.1, as depicted in Figure 4.1. The process begins with encoding the input image x using the generative encoder $y = g_a(x)$. Next, the hyper-latent representation $z = h_a(y)$ is obtained through the encoder of the hyper-prior network. The quantized version of the hyper-latent, denoted as $\hat{z}$, is modeled and entropy-coded using a learned factorized model before being passed through $h_s(\hat{z})$. The output of h_s, along with the output of the context model, feeds into the entropy parameters network [10]. This network generates the mean μ and scale σ parameters for a conditional Gaussian entropy model $P(y|\hat{z}) = \mathcal{N}(\mu, \sigma^2)$ to model y. Finally, the quantized latent $\hat{y} = Q(y)$ undergoes entropy-coded Arithmetic Encoding (AE)/Arithmetic Decoding (AD) and is then sent to $\hat{x} = g_s(\hat{y})$ to reconstruct the original image $\hat{x}$.

4.2.1. SSMIC Encoder

The generative and the hyper-prior encoder, g_a and h_a, are built with the patch merge block and the VSS block illustrated in Figure 4.2.

4.2.2. Patch Merge Block

The patch merge block employed is identical to that used in SwinNPE (refer to Chapter 3 in Subsection 3.3.2) patch merge block. In contrast to the convolutional layer utilized in [55] for a similar purpose, this patch merge block provides enhanced computational efficiency. Its streamlined design reduces complexity, making it not only more resource-friendly but also easier to integrate into the overall model architecture.

4.2.3. Visual State Space (VSS)

In the following section, we present the architecture of the VSS block and its 2D Selective Scan (SS2D) mechanism [13], designed to bridge the gap between traditional 1D array scanning and 2D plane traversal. This innovative mechanism extends selective SSM, enabling more efficient processing of visual data. While numerous selective scan mechanisms and visual state space blocks have been introduced in the literature (see [109]), we chose the VSS block with its SS2D mechanism for its balance of consistent performance and ease of implementation.

The VSS block is formulated by substituting the S6 module, which serves as the core of the Mamba architecture [17]. The S6 module is notable for concurrently achieving global receptive fields, dynamic weights (i.e., selectivity), and linear complexity. By integrating the SS2D, this solution delivers robust results while maintaining simplicity in deployment.

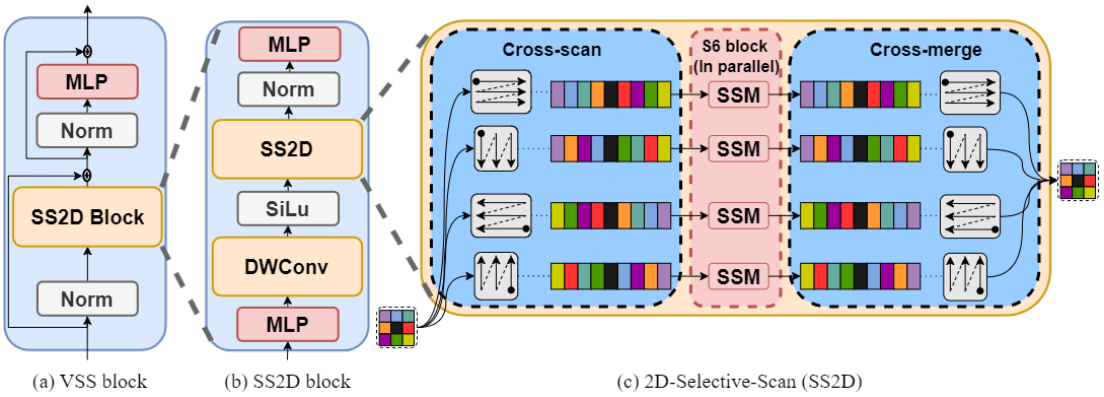


Figure 4.2: A VSS block [13] consists of an SS2D block, which performs selective scans in four parallel patterns.

4.2.3.1. VSS Block

VSS block, originally proposed in [13], consists of a single network branch with two residual modules, mimicking the architecture of the vanilla Transformer block [1]. Specifically, each stage in our SSMIC consists of a sequence of VSS blocks, and the number of blocks in stage i is denoted as d_i (see in Figure 4.1). Given an input feature maps $\mathbf{f} \in \mathbb{R}^{H \times W \times C}$, we get $\mathbf{f}''$ from a first residual module:

$$\mathbf{f}'' = \mathbf{f} + \mathbf{f}', \tag{4.1}$$

where $\mathbf{f}'$ is obtained through multiple layers as follows:

$$\mathbf{f}' = \mathrm{MLP}_2(\mathrm{LN}_2(\mathrm{SS2D}(\sigma(\mathrm{DWConv}(\mathrm{MLP}_1(\mathrm{LN}_1((f)))))))). \tag{4.2}$$

As illustrated in Figure 4.2 (see (a) and (b)), the output of the VSS block is given by:

$$\mathbf{f}_{out} = \mathrm{MLP}_3(\mathrm{LN}_3(\mathbf{f}'')) + \mathbf{f}'', \tag{4.3}$$

where LN represents the normalization layer; SS2D is the 2D selective scan module; σ is the SiLU activation [110]; DWConv is the depthwise convolution; and MLP is the learnable linear projection.

Unlike VSS block in [13], we employ Root Mean Square Normalisation (RMSNorm) [111] instead of LayerNorm. RMSNorm normalizes the input by centering it around zero and scaling it based on its magnitude. In contrast to standard layer normalization, which depends on the mean and variance, RMSNorm focuses on the root mean square of the input. This approach not only enhances convergence speed during training but is also computationally simpler, as it eliminates the need to compute the mean and instead emphasizes the magnitude of the inputs.

4.2.3.2. Selective Scan Approach

We adopt the selective scan approach proposed in [13], which adapts input-dependent selection mechanism [17] to vision data without compromising its advantages. The SS2D process consists of three steps, as shown in Figure 4.2 (c).
- *Cross-scan*: unfolds input features into sequences along four distinct traversal paths;
- *Selective scan*: processes each path with a distinct S6 in parallel;
- *Cross-merge*: subsequently reshapes and merges the resultant sequences to form the output feature, enabling effectively integrating information from other pixels in different directions relative to each current pixel.

4.2.4. SSMIC Decoder

The SwinNPE decoder consists of a generative decoder g_s and a hyper- prior decoder h_s, both constructed using the patch split block and the VSS block. The patch split block is identical to that used in the SwinNPE architecture described in Chapter 3 Subsection 3.3.2.

4.2.5. Quantization and Entropy Model in SSMIC

To simulate the quantization operation, we employ the same strategy as in SwinNPE (Chapter 3 Section 3.3). Our approach utilizes the context model and the entropy parameters network from [10], which draws inspiration from traditional coding techniques that predict the probability of unknown codes based on previously decoded latent. Unlike the architecture in [55], our approach emphasizes computational efficiency, allowing us to achieve superior performance while main-

taining a streamlined and practical design.

4.3. Proposed State Space Model-based Image Compression with Channel Wise Autoregressive (SSMIC_CW)

We propose an alternative version of State-Space Model-based Image Compression (SSMIC_CW) that incorporates the Channel-wise AutoRegressive Model [76], as outlined in Chapter 3 and illustrated in Figure 3.5. While this model enhances performance, it also incurs a significant increase in computational time. This version allows us to assess the effectiveness of our method when combined with a robust autoregressive model, providing a contrast to the original SSMIC.

4.4. Experimental Results

This section evaluates the proposed SSMIC architecture and compares its image compression results with state-of-the-art approaches. After outlining the experimental settings, we will examine various End-to-End Neural network model-based approaches in terms of their compression performance and computational complexity.

4.4.1. Experimental Settings

The SSMIC model was trained on the CLIC2020 training set [15], employing the same loss function used by the SwinNPE model, as outlined in Chapter 3,

Equation 3.1. For a comprehensive understanding of the loss function, please refer to Chapter 2, Section 2.3.1.2, where its formulation is provided as follows:

$$L = D + \lambda R,$$

where R represents the bitrate and D denotes the distortion.

The MSE in the RGB color space serves as the distortion metric. The Lagrangian multiplier λ regulates the R-D trade-off. We trained the SSMIC model with values of λ set $\in \{100, 50, 30, 10\}$. Each training batch consisted of 8 randomly cropped images, each sized 256×256. We performed a total of 1 million (1M) iterations using the ADAM optimizer, with the learning rate fixed at 10^{-4}. The hyper-parameters of the architecture, as shown in Figure 4.1, are as follows: $(d_1, d_2, d_3, d_4, d_5, d_6) = (2, 2, 6, 2, 5, 1)$ and $(C_1, C_2, C_3, C_4, C_5, C_6) = (128, 192, 256, 320, 192, 192)$.

The entropy parameters network, as in [10], comprises three consecutive convolutional layers with stride equal to 1, and channel numbers $640, 512, 384$ respectively. The context model [10] utilizes a masked convolution [69] with a kernel size of 5×5 and a channel number of 384 with stride equal to 1.

Our SSMIC architecture was implemented in Pytorch using the CompressAI library [112] on an NVIDIA A100 80 Go GPU and an Intel Xeon Gold 6330 3.10 GHz CPU.

We evaluate the performance of the model on three datasets: Kodak [11], JPEG-AI [12], and the test set of CLIC2020 [15] including both mobile and professional categories. During inference, images that are not multiples of 256 were zero-padded. The proposed SSMIC model is compared against several models: SwinT [7], MambaVC [55], LightweightLIC [74], LIC_TCM [3], MLIC+ [81], ELIC [82] and our previous SwinNPE model in Chapter 3. These models were se-

Method	Kodak [11]	CLIC2020 [15]	JPEG-AI [12]	Average
BPG444 [113]	29.86%	32.77%	43.77%	35.46%
SwinT* [7]	-10.52%	-6.47%	-2.78%	-6.03%
MambaVC* [55]	-15.37%	-16.69%	-12.47%	-14.69%
SwinNPE (Chapter 3)	-5.85%	-17.50%	-23.56%	-15.63%
LightweightLIC [74]	-7.76%	-23.60%	-29.86%	-20.40%
ELIC [82]	-11.14%	-27.45%	-31.31%	-23.30%
MLIC+* [81]	**-15.86%**	-	-15.89%	-
LIC_TCM [3]	-13.76%	**-30.65%**	**-33.37%**	**-25.92%**
SSMIC (Ours)	-9.81%	-29.91%	-25.55%	-21.75%
SSMIC_CW (Ours)	-11.75%	-27.15%	-23.92%	-20.94%

Table 4.1: BD-rate performance using VTM-15.0 [14] as reference.

lected for their competitive compression performance or computational efficiency.

We also present the compression performance of Better Portable Graphics (BPG) [113] and VTM [14] as baseline comparisons. Detailed information on the specific commands and software used to generate these results can be found in Appendix A.

For LightweightLIC [74], LIC_TCM [3], and ELIC [82] and our previous SwinNPE (Chapter 3), we evaluated their compression performance and complexity efficiency using their provided pre-trained models under the same configuration as SSMIC. For SwinT [7], MambaVC [55], and MLIC+ [81], as pre-trained models were not available, we assessed their computational complexity using their untrained models and referenced the RD-curves from their respective papers to evaluate compression performance. All experiments were carried out on an A100 80 Go GPU and an Intel Xeon Gold 6330 3.10 GHz CPU.

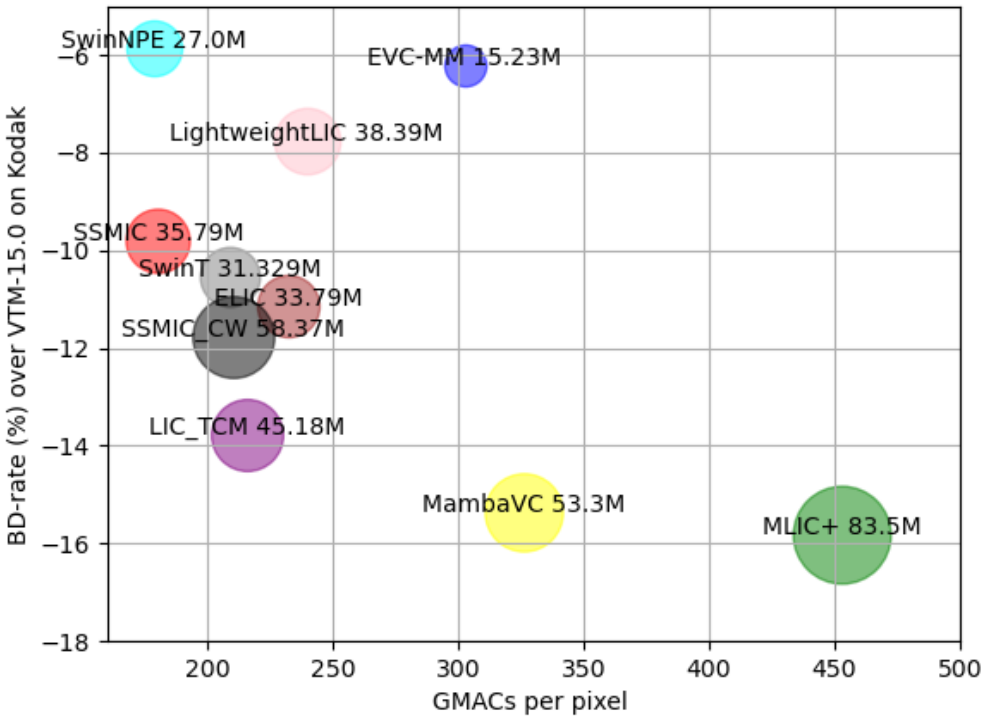


Figure 4.3: BD-rate performance over VTM-15.0 [14] vs computational complexity (GMACs) on Kodak [11].

4.4.2. Results and Discussion

We summarize in Table 4.1, the BD-rate of SSMIC and the competitive state-of-the-art models across three datasets. The BD-rate was calculated covering approximately 0.4 to 1.2 Bit Per Pixel (BPP), using VTM-15.0 [14] as the reference, We use $*$ to indicate the approaches for which the numbers are sourced from their respective papers. Note that the evaluation of MLIC+ on the CLIC2020 dataset is not provided in [81].

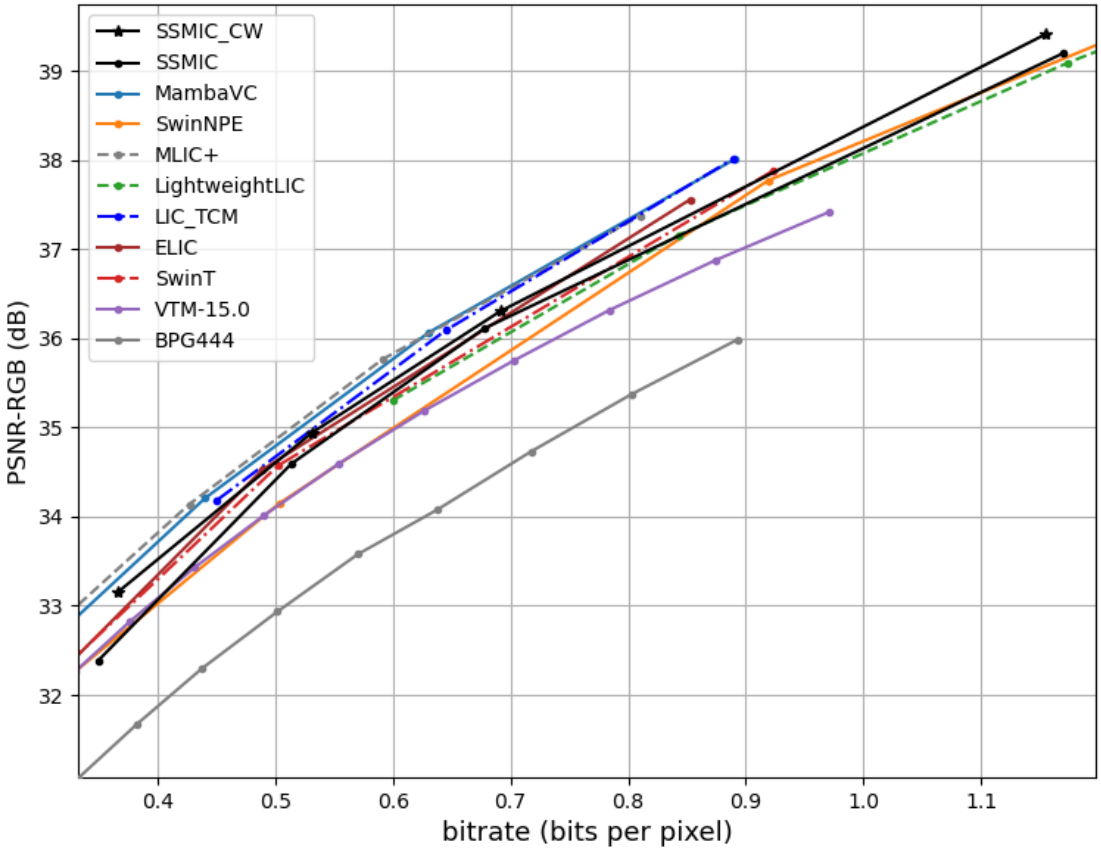


Figure 4.4: Performance evaluation on the Kodak dataset [11].

On average, SSMIC achieves a -21.75% BD-rate performance compared to VTM-15.0 and 4.17 p.p.,translating to a relative increase from LIC_TCM [3] which delivers the best performance among the selected methods. However, the latter is significantly more demanding in terms of computational complexity and number of parameters. This is illustrated in Fig. 4.3, which plots the BD-rate with VTM-15.0 as an anchor versus the GMACs[4] of various approaches on the Kodak dataset. The radius of the circles represents the number of parameters of the model. The methods positioned further towards the bottom left of the graph demonstrate better performance. These findings confirm that our SSMIC offers a good trade-off between BD-rate performance and computational complexity.

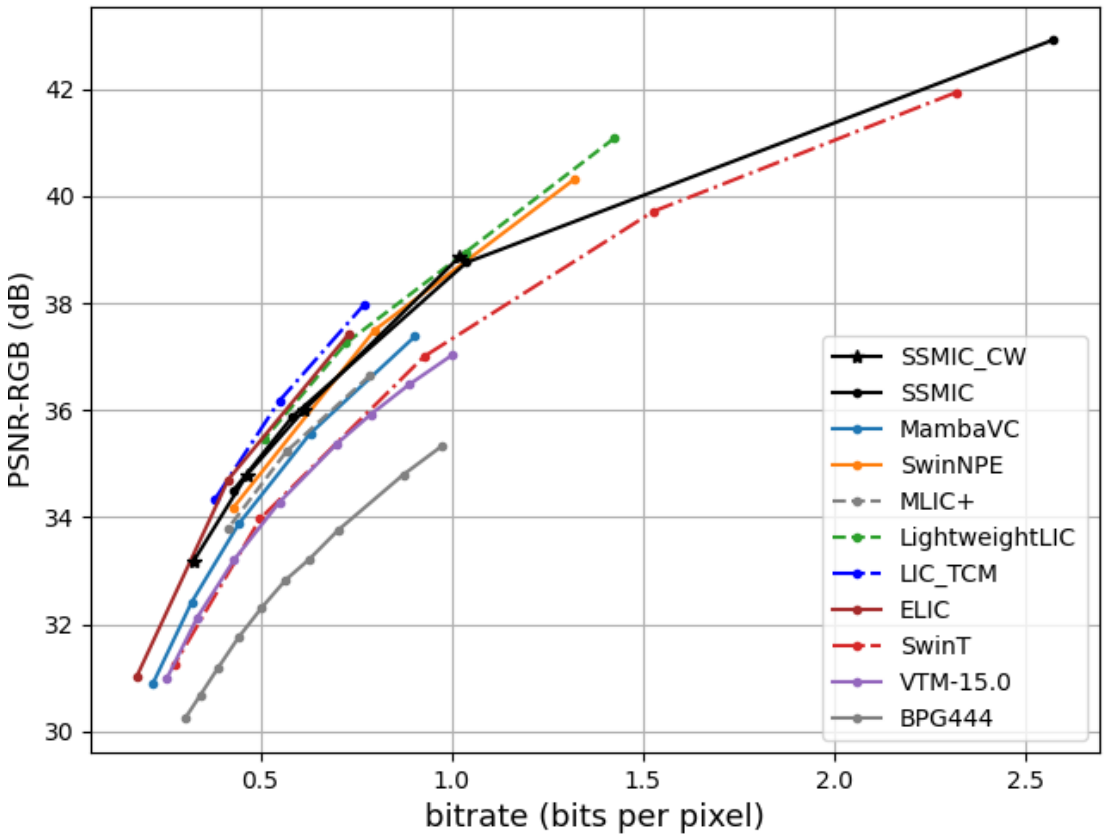


Figure 4.5: Performance evaluation on the JPEG-AI dataset [12].

We also evaluate the compression performance of different models across various bitrate ranges on the Kodak, JPEG-AI, and CLIC2020 datasets, as illustrated in Figures 4.4,4.5 and 4.6 respectively. The results show that the proposed model consistently achieves performance comparable to the state-of-the-art methods while maintaining competitive computational efficiency. Indeed, we show in Tables 4.3 and 4.4 the computational complexity in terms of MACs[4] and FLOPs[4], respectively. The results cover three different resolutions, selected for their com-

[4]MACs and FLOPs for most models were calculated using the calflops library `https://github.com/MrYxJ/calculate-flops.pytorch`. However, for SwinNPE (as discussed in Chapter 3), the FLOPs were computed using TensorFlow. The MACs for SwinNPE were approximated by dividing the FLOPs by 2, following the standard assumption that, in many models, each multiply-accumulate operation corresponds to two floating-point operations.

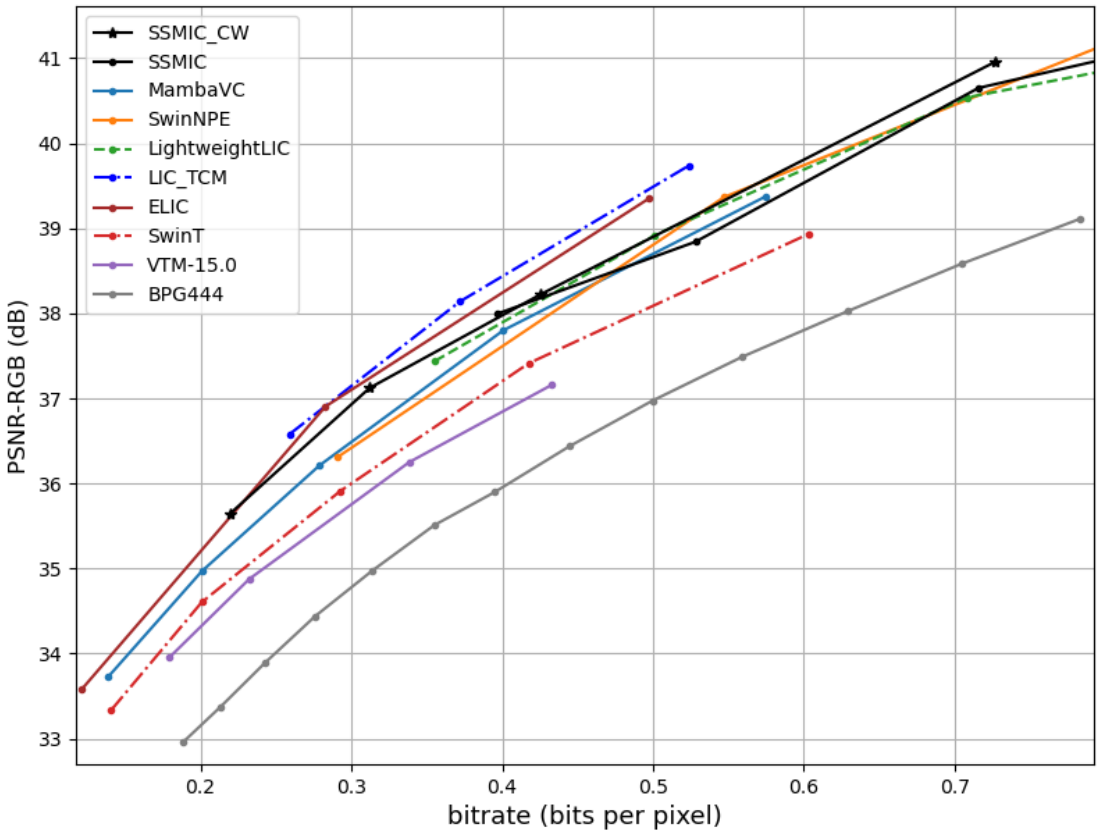


Figure 4.6: Performance evaluation on the CLIC2020 dataset [15].

mon usage and to avoid out-of-memory issues on the utilized GPU. It is clear that our SSMIC significantly reduces the computational complexity in terms of MACs and achieves competitive results in terms of FLOPs compared to SwinNPE in Chapter 3.

Table 4.2 displays the average latency over 2,000 images at a resolution of 256×256 on the utilized GPU. Our model shows competitive decoding times when compared to lightweightLIC [74], while also achieving similar encoding times. Figures 4.8 and 4.7 present a quantitative comparison between SSMIC and SSMIC_CW. Notably, SSMIC_CW consistently outperforms SSMIC by achieving superior reconstruction quality for the same bitrate, as illustrated in Figure 4.8

and 4.7. Additionally, further analyses of spatial correlation across different λ values are provided in Appendix A for a more comprehensive evaluation.

Method	Latency GPU (ms)	
	Encoding	Decoding
ELIC [82]	426.94	517.15
MambaVC* [55]	**14.01**	73.36
LIC_TCM [3]	15.23	52.46
SwinT* [7]	14.25	20.86
LightweightLIC [74]	15.62	**15.56**
SSMIC (Ours)	20.24	19.93
SSMIC_CW (Ours)	21.54	58.41

Table 4.2: Average latency, measured on an A100 80 Go GPU and an Intel Xeon Gold 6330 3.10 GHz CPU, using over 2000 images at 256×256 resolution. (*) We conducted the inference without loading the pre-trained weights because the pre-trained model was unavailable.

Resolution	SSMIC (Ours)	SwinNPE (Chapter 3)	SwinT [7]	LIC_TCM [3]	ELIC [82]	LightweightLIC [74]	MambaVC [55]	MLIC+ [81]
768×512	**180.053G**	$\approx 178.477G$	208.789G	215.316G	231.930G	239.213G	326.112G	452.622G
1024×768	**360.106G**	$\approx 356.955G$	417.570G	430.632G	463.861G	478.425G	652.224G	905.243G
1280×1280	**750.222G**	$\approx 743.655G$	869.956G	897.150G	966.376G	996.719G	OM	1.8859T
#parameters (M)	35.79	27.00	32.34	45.18	33.79	38.39	53.30	83.50

Resolution	SSMIC_CW (Ours)	SwinNPE (Chapter 3)	SwinT [7]	LIC_TCM [3]	ELIC [82]	LightweightLIC [74]	MambaVC [55]	MLIC+ [81]
768×512	210.262G	$\approx 178.477G$	**208.789G**	215.316G	231.930G	239.213G	326.112G	452.622G
1024×768	420.524G	$\approx 356.955G$	**417.570G**	430.632G	463.861G	478.425G	652.224G	905.243G
1280×1280	876.093G	$\approx 743.655G$	**869.956G**	897.150G	966.376G	996.719G	OM	1.8859T
#parameters (M)	58.37	27.00	32.34	45.18	33.79	38.39	53.30	83.50

Table 4.3: Multiply-Add Cumulation (MACs) for different image resolutions. The last line gives the number of parameters for each model. OM for Out of Memory. ($\approx$) For SwinNPE (Chapter 3) we follow the standard assumption that, in many models, each multiply-accumulate operation corresponds to two floating-point operations.

Resolution	SSMIC (Ours)	SwinNPE (Chapter 3)	SwinT [7]	LIC_TCM [3]	ELIC [82]	LightweightLIC [74]	MambaVC [55]	MLIC+ [81]
768×512	439.618G	**356.955G**	419.215G	441.378G	464.640G	480.031G	815.119G	905.845G
1024×768	879.247G	**713.911G**	838.430G	882.761G	929.279G	960.062G	1.6302T	1.8117T
1280×1280	1.8317T	**1.4873T**	1.7467T	1.8391T	1.9360T	2.0001T	OM	3.7744T

Resolution	SSMIC_CW (Ours)	SwinNPE (Chapter 3)	SwinT [7]	LIC_TCM [3]	ELIC [82]	LightweightLIC [74]	MambaVC [55]	MLIC+ [81]
768×512	501.182G	**356.955G**	419.215G	441.378G	464.640G	480.031G	815.119G	905.845G
1024×768	1.0024T	**713.911G**	838.430G	882.761G	929.279G	960.062G	1.6302T	1.8117T
1280×1280	2.0882T	**1.4873T**	1.7467T	1.8391T	1.9360T	2.0001T	OM	3.7744T

Table 4.4: Floating Point Operations (FLOPs) for different image resolutions. OM for Out of Memory.

Ground Truth *Kodim06.png*	SSMIC [0.466/31.30/14.69]	SSMIC_CW [0.467/32.13/15.58]

Figure 4.7: Visualization of the reconstructed images from Kodak dataset*"Kodim06"*. The metrics of SSMIC_CW [≈ bpp/↑ PSNR(dB)/↑ MS-SSIM(dB)] compared to those of SSMIC.

Ground Truth *Kodim05.png*	SSMIC [1.121/34.86/21.30]	SSMIC_CW [1.116/35.18/21.59]

Figure 4.8: Visualization of the reconstructed images from the Kodak dataset*"Kodim05"*. The metrics of SSMIC_CW [≈ bpp/↑ PSNR(dB)/↑ MS-SSIM(dB)] compared to those of SSMIC.

4.5. Conclusion

In this chapter, we introduced the State Space Model-based Image Compression (SSMIC) approach, which achieves competitive RD performance while significantly reducing computational complexity and latency. Leveraging the strengths of SSMs derived from the Mamba model, SSMIC enhances contextual reasoning while effectively managing computational and memory demands. Across benchmark datasets, SSMIC demonstrates a significant reduction in BD-rate compared to VTM-15.0, highlighting its effectiveness for practical applications. While this chapter focused on investigating state space models in neural networks for image compression, it would be worthwhile to explore a strategy that adjusts bit-rate using a single model rather than multiple models across different bitrates. This will be the focus of the next chapter.

5. Universal End-to-End Neural Network for Lossy Image Compression

Contents

Summary of this Chapter

This chapter presents variable bitrate lossy image compression using a VAE-based neural network. We propose an adaptable image quality adjustment strategy that innovatively adjusts the input scale exclusively during the inference phase, leading to an exceptionally efficient rate-distortion mechanism. Through extensive experimentation, across diverse VAE-based compression architectures (CNN, Vision Transformer (ViT)) including the proposed models presented in Chapter 3, 4 and training methodologies (MSE, SSIM), our approach exhibits remarkable universality. This success is attributed to the inherent generalization capacity of neural networks. Unlike methods that adjust model architecture or loss functions, our approach emphasizes simplicity, reducing computational complexity and memory requirements. The experiments not only highlight the effectiveness of our approach but also indicate its potential to drive advancements in variable-rate neural network lossy image compression methodologies. These results were presented at *European Signal Processing Conference (EUSIPCO) 2024*, validating the innovative nature and impact of our approach in the field.

5.1. Introduction

In most auto-encoder-based coding schemes, the Variational Auto-encoder (VAE) seeks to optimize performance by minimizing the distortion between the original image and its compressed-decompressed version while adhering to a specified target bitrate constraint (Section 2.3). This complex rate-distortion optimization problem is tackled using Lagrange formalism, introducing a Lagrange multiplier into the VAE's objective loss function. This function typically consists of two terms: a distortion term, measuring the difference between the original image and its compressed-decompressed counterpart, and a rate regularization term which enforces the bitrate constraint. During the end-to-end training process, this function guides the VAE to optimally adjust the parameters of both the encoder and the decoder. This ensures not only optimal compression performance but also the preservation of the quality of the reconstructed image. By minimizing the objective function for a specific value of the Lagrange multiplier, we can identify the points on the convex hull that correspond to all possible rate-distortion combinations. Consequently, training the VAE for each rate-distortion pair with different values of the Lagrange multiplier results in a distinct compression model for each specific multiplier. However, This individualized training process is both computationally expensive and time-consuming, which restricts the flexibility of auto-encoder-based methods in practical applications where variable bitrates are required.

To address this issue, several strategies have been proposed to enable variable bitrate compression using a single trained model, as discussed in Section 2.3. For example, Yoojin et al. introduced a conditional auto-encoder that incorporates the Lagrange multiplier as a conditioning variable throughout the decoder,

encoder, and entropy model [58]. Theis et al. introduced a scale parameter that allows fine-tuning a pre-trained auto-encoder for various bitrate targets [43]. Moreover, the approach proposed in [59] extends this idea by integrating a modulated auto-encoder with a VAE. Guerin Jr et al. suggested a modification to the loss function to implement rate control within the VAE framework, although this method requires training multiple models, each fixed to specific bitrates [114]. Similarly, other strategies have been explored, including substituting the loss term with rate estimation or using gain units for rate adaptation [16,60]. Furthermore, a variable quantization method that adjusts the bitrate by controlling the size of the quantization bins has been proposed [58].

Despite these advancements, many of these strategies involve architectural modifications, additional modules, or changes to the loss function, often leading to increased computational complexity and memory usage. Furthermore, fine-tuning or integrating rate-conditioned modules can significantly extend times and elevate model complexity, and may still fall short of the compression efficiency achieved by traditional codecs such as BPG [60]. Incorporating scale parameters in the latent space may also introduce potential compatibility issues [43], further complicating the model's performance.

In this chapter, we propose a novel approach to variable bitrate lossy image compression using a VAE-based neural network, effectively addressing the limitations of existing methods. Our strategy stands out by allowing flexible adjustment of image quality using only a single trained model, eliminating the need for architectural modifications or retraining. Rather than depending on intricate rate control mechanisms, our approach adjusts the input scale in the image space exclusively during the inference process. This significantly simplifies the implementation while maintaining the flexibility to adapt to different bitrates. By leveraging this efficient scaling mechanism, we demonstrate that our method

achieves accurate performance, with lower computational complexity compared to existing state-of-the-art techniques. This streamlined strategy offers a practical solution for achieving variable bitrate compression.

5.2. Universal End-to-End VAE for Lossy Image Compression

The VAE designed for image compression efficiently learns to represent input data in a latent space by minimizing an objective loss function composed of two main components: faithful reconstruction of input data to reduce distortion, and regularization of the latent space to control the rate. The regularization component prevents the latent space from becoming too complex, promoting more understandable and useful representations. By computing all VAE parameters to minimize this objective function, the model adopts a holistic training approach. Essentially, the VAE is engineered to optimize the rate-distortion pair, with a fixed regularization point. This ensures a balanced compromise between data compression and reconstruction quality, tailored to specific requirements dictated by the regularization. To simplify the training process, this work suggests using a single end-to-end trained VAE model for image compression, emphasizing the minimization of the objective function.

5.2.1. Objective Loss Function

Assuming the VAE has been trained for a specific regularization point λ_K that yields an optimal rate-distortion pair (D_K, R_K) (e.g. the point furthest to the right on the rate-distortion curve), the minimization of the objective loss function

can be expressed as follows:

$$L(\theta_K, \phi_K, \lambda_K) = D_K + \lambda_K R_K, \tag{5.1}$$

with

$$D_K = D(x, \hat{x}) = D(x, g_{\phi_K}(Q(f_{\theta_K}(x)))), \tag{5.2}$$

and

$$R_K = R(\hat{y}) = R(Q(f_{\theta_K}(x))), \tag{5.3}$$

where D is the distortion between the original image x and its compressed-decompressed version $\hat{x}$ given by $\hat{x} = g_\phi(Q(f_\theta(x)))$. Q is the quantization operator. $f_\theta()$ concerns the encoder side with θ parameters and $g_\phi()$ being the decoder side with ϕ parameters. R is the estimated bit-rate.

5.2.2. Scaling Factor Strategy

Our universal codec image architecture is illustrated in Figure 5.1. More details are given below.

Now, suppose that the encoder and decoder are ready for use, meaning that all the parameters of the VAE have been carefully calculated for a given λ_K according to the objective function described below. To compress the original image at a bit-rate different from the one for which the codec was configured, we propose to reduce the dynamic range of the original image. To achieve this, we introduce a scaling factor, denoted s, belonging to the interval $]0, 1[$. The original image x, intended for compression, is then scaled by this factor s as follows:

$$x_s = s \times x \tag{5.4}$$

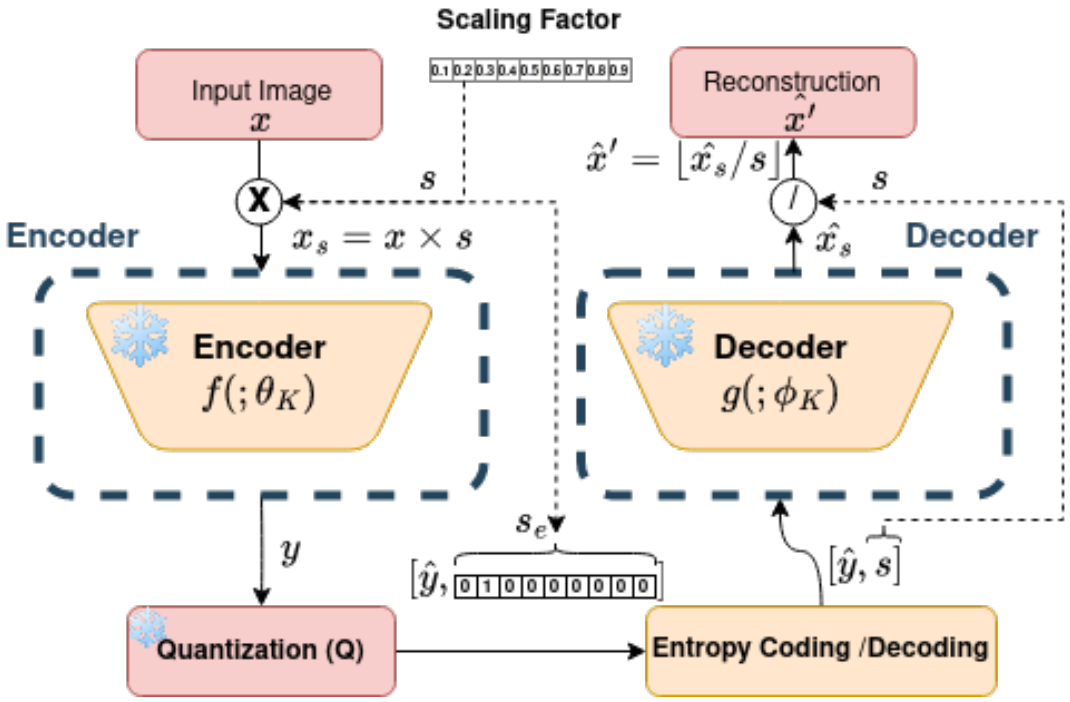


Figure 5.1: Universal codec image architecture.

before being fed into the VAE encoder for compression.

In the decoding process, the compressed stream $\hat{y}$ is sent to the VAE decoder. The compressed-decompressed image is then rescaled. In addition to the error induced by the already trained VAE (i.e., D_K), an additional error arises from the scaling and rounding operations, as the compressed-decompressed image is deduced from $\hat{x}' = \lfloor \frac{\hat{x}_s}{s} \rceil$ (where $\lfloor . \rceil$ denotes the rounding operation). Thus, we can conclude that:

$$D(x, \lfloor \frac{\hat{x}_s}{s} \rceil) \geq D_K = D(x, \hat{x}). \tag{5.5}$$

Let us recall that the proposed universal codec has been frozen for a specific objective function $L(\theta_K, \phi_K, \lambda_K)$ that corresponds to a given operating mode (i.e., D_K, R_k). According to Eq. 5.5, the distortion has increased compared to

the original distortion (ie.e, D_K) induced by the frozen encoder. To maintain this objective loss function $L(\theta_K, \phi_K, \lambda_K)$, a reduction in the bitrate is consequently enforced. As a result, a new rate-distortion pair (D_{K_s}, R_{K_s}) can be inferred without the need to retrain the VAE compression architecture for an additional regularization point.

This proposed scaling factor strategy enables the construction of multiple VAE codecs that operate in various modes, all relying on a single VAE frozen for a specific operational mode. This will be supported by the experiments we conducted, with the results analyzed in section 5.3.

5.2.3. Analysis and Discussion

The scaling strategy outlined in the previous section enhances the practicality of our method by enabling variable-rate compression without requiring additional training for different rate-distortion points. Unlike existing state-of-the-art methods, which often require complex architectural modifications or retraining for each bitrate setting, our strategy leverages the generalization ability of neural networks. This allows us to efficiently adjust compression parameters during inference, making our approach both universal and adaptable to various compression needs. Compared to state-of-the-art methods, our solution is simpler and more computationally efficient, while still delivering accurate performance in terms of both bitrate and image quality.

However, when the scaling factor s becomes too small (i.e. when the image dynamics are shrunk drastically), compression performance deteriorates significantly, as the added distortion outweighs the benefits of bitrate reduction. This is consistent with very low bitrate coding. This also depends on the chosen operating point of the frozen codec.

Additionally, attempting to apply scaling factors outside the $]0, 1[$ interval would result in performance issues, as the VAE was not trained on such values, potentially leading to domain-shifting problems that degrade neural network performance.

5.3. Experimental Results

This section shows that the strategy proposed in the previous section enables the construction of the complete rate-distortion curve with only one selected pair of points (D_K, R_K), relying solely on a single VAE model that has already been trained for the corresponding regularization point λ_K.

5.3.1. Experimental Settings

Different end-to-end neural network architectures have been selected:

- Fully convolutional based methods: Factorized Hyperprior, Scale Hyperprior [9] and Joint Autoregressive and Hierarchical Priors [10];
- Convolutional methods with attention modules: Discretized Gaussian Mixture Likelihoods and Attention Modules [2];
- Transformer-based methods: SwinNPE (Chapter 3) and (iv) Space State Model (Mamba model): SSMIC (Section 4).

The trained models are those provided by the CompressAI framework [112], including 8 different regularization values (i.e. λ_i), except for [2] with 6 regularization values. All models are optimized based on MSE and MS-SSIM as distortion metric in RGB color space, and the quantization is performed using *torch.round()*.

In the case of SwinNPE (Chapter 3), we trained the model on the CLIC2020 [15] dataset using 4 regularization values $\lambda_1 = 0.003, \lambda_2 = 0.001, \lambda_3 = 0.0003, \lambda_4 = 0.0001$ and employed *tf.round()* for quantization. The evaluation was performed on the Kodak dataset [11].

For SSMIC (Section 4), same as SwinNPE with different regularization values $\lambda_1 = 100, \lambda_2 = 50, \lambda_3 = 30, \lambda_4 = 10$, for Asymmetric Gained Deep [16] method, we extracted the results from their paper.

During the experimental process, we select the regularization point λ_K at which the VAE image compression has been trained to achieve an optimal rate-distortion pair (D_K, R_K) according to the minimization of the objective loss function. We define 9 scaling factors $s = 0.1, 0.2...0.8, 0.9$ that belong to the interval $]0, 1[$.

5.3.2. Impact of the Scaling Factor on the Codec Bitrate

We adopt a naive one-hot representation for the scaling factor s, as illustrated in Table 5.1, which encompasses 256 possible scaling factors. In this representation, each possible value of s is mapped to a specific 1-byte binary sequence. For example, $s = 0.1$ is encoded as 10000000 and $s = 1.0$ as 00000000.

Additionally, Table 5.2 illustrates the impact of this encoded scaling factor s on the bitrate per pixel for the Kodak image *"Kodim01"*. ΔR quantifies the difference in bitrate with and without the scaling factor. Specifically, the table details how encoding using one, two, or three bytes influences the bitrate increase, whether s is concatenated with the latent space $\hat{y}$ or encoded independently without entropy coding. Overall, the effect of encoding s on the bitrate is minimal and negligible, ensuring that the introduction of a scaling factor does not impact the overall compression performance. Additionally, one can observe that it is unnecessary to encode the scaling factor with the latent space.

Scaling Factor s	s_e (encoded s) on 1 byte
0.1	10000000
0.2	01000000
0.3	00100000
0.4	00010000
0.5	00001000
0.6	00000100
0.7	00000010
0.8	00000001
0.9	11111111
1.0	00000000

Table 5.1: An example of one-hot representation of the scaling factor s using 1 byte.

5.3.3. Results and Discussion

In the various plots presented in Figures 5.2,5.4 and 5.5, the solid rate-distortion curves correspond to reference curves obtained from the model trained at different regularization points, with inference performed on each trained model. In contrast, the dotted rate-distortion curves correspond to curves generated using a single trained model. For each new rate-distortion point, this trained model is inferred with an input image that has been scaled by a factor s.

Figure 5.2 depicts the rate-distortion reference curve of SwinNPE (Chapter 3). We choose $\lambda_K = \lambda_4$ and employ this trained SwinNPE to generate the dotted rate-distortion curve, as explained below. Notably, this curve fits perfectly with the reference curve for a bitrate greater than 0.4 bpp.

A performance comparison with the Asymmetric Gained Deep Image Compression With Continuous Rate Adaptation [16] shows that our strategy yields similar results. Figure 5.3 highlights the significance of selecting the trained SwinNPE model as a universal model. Indeed, it is observed that depending on the chosen

Encoded Scaling Factor s	1 byte	2 bytes	3 bytes
	ΔR	ΔR	ΔR
s concatenated with the latent space $\hat{y}$ using entropy coding	3×10^{-4}	6×10^{-4}	9×10^{-4}
s alone without entropy coding	2×10^{-5}	4×10^{-5}	6×10^{-5}

Table 5.2: Impact of the scaling factor on the birate. (ΔR) quantifies the difference in bitrate with and without the scaling factor, using the Kodak image *"Kodim01"* in two scenarios : (i) the scaling factor is represented using one, two, or three bytes; (ii) the scaling factor is concatenated to the latent space and undergoes entropy encoding.

regularization point (i.e., $\lambda_K = \lambda_1$, $\lambda_K = \lambda_2$, $\lambda_K = \lambda_3$, or $\lambda_K = \lambda_4$), the rate-distortion curve (referred to as the SwinNPE optimal curve in this figure) fits better to the reference curve while expanding the range of possible rates. Since the scale factor strategy increases distortion at the expense of reduced bitrate, relying on a model trained with a specific regularization point is crucial for ensuring excellent quality of the compressed-decompressed image while maintaining a low bitrate.

In Figure 5.4, the dotted distortion-rate curve is generated based on the top-right regularization point, i.e., λ_K, of the corresponding solid reference curve for each image compression method. One can observe that our strategy allows for fitting the rate-distortion of the reference curves for Scale Hyperprior [9], Joint Autoregressive and Hierarchical Priors [10], and Discretized Gaussian Mixture Likelihoods and Attention Modules [2]. However, for Factorized Hyperprior [9], the performance decreases compared to the reference (up to $\approx 1dB$) when significantly reducing the dynamic range of the image (i.e. when s is too small).

To grasp the impact of the scaling factor s, Figure 5.6 and 5.7 displays the normalized histograms of the latent space for SwinNPE (Chapter 3) and SSMIC

(Chapter 4.2) respectively, using the *kodim01.png* image from the Kodak dataset [11]. The histograms exhibit a Laplacian distribution that narrows as the scale s decreases, with increased amplitude indicating a higher level of sparsity in the latent space.

The common feature of the VAE used is that quantization is fixed, and the parameters of this quantization are not fine-tuned during the learning process. This imparts a level of flexibility to the regularization point, influencing the trade-off between rate and distortion. Introducing the scale factor is akin to incorporating an additional form of uniform quantization, which increases the distortion according to the rounding step to recover the original image.

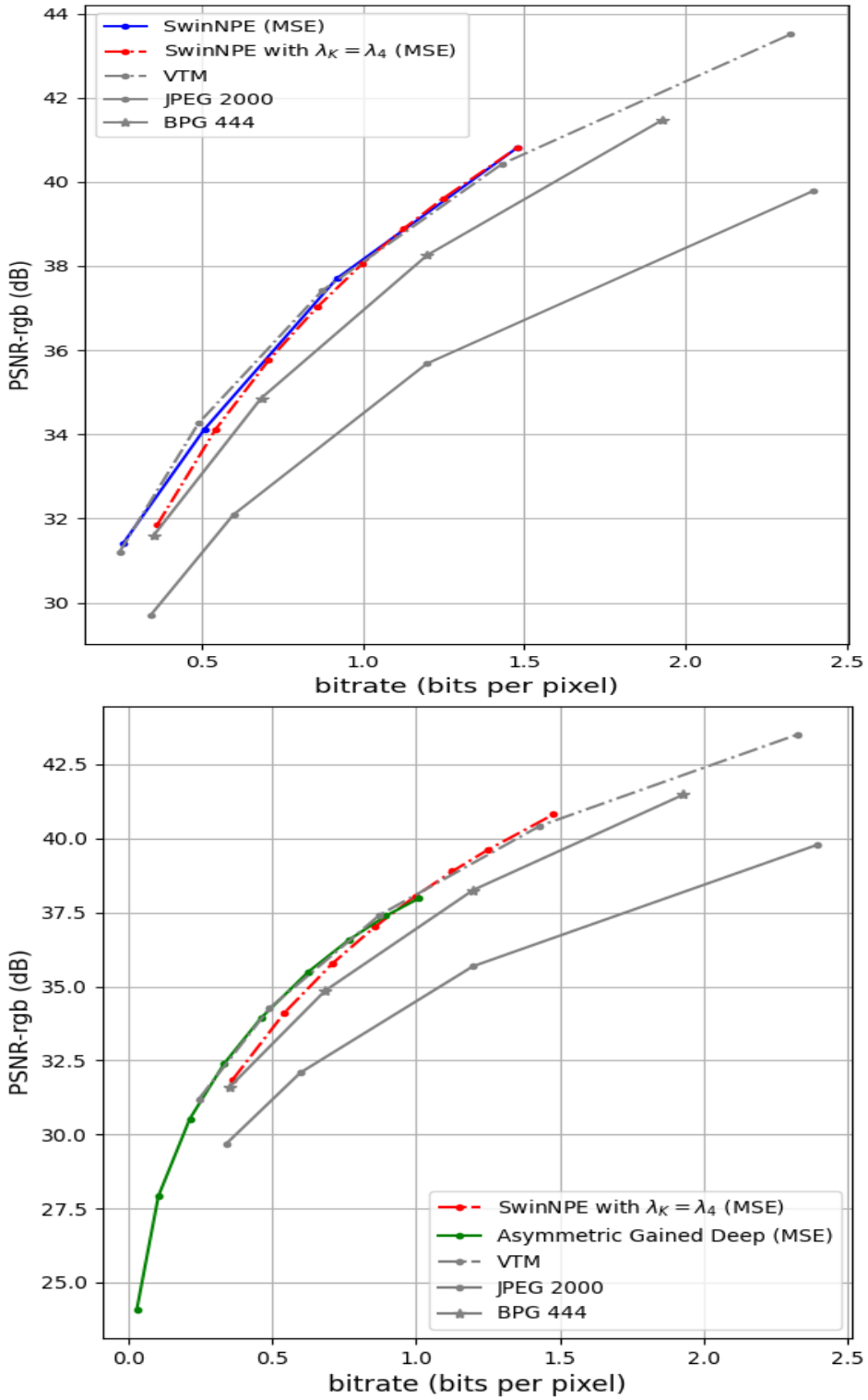


Figure 5.2: 1st Figure: Rate-distortion reference curve of SwinNPE and dotted curve depicts the rate-distortion achieved by employing a single regularization point. 2nd Figure: the same dotted curve (i.e. using the top right point of SwinNPE) with Asymmetric Gained Deep variable bitrate model [16] on Kodak dataset [11].

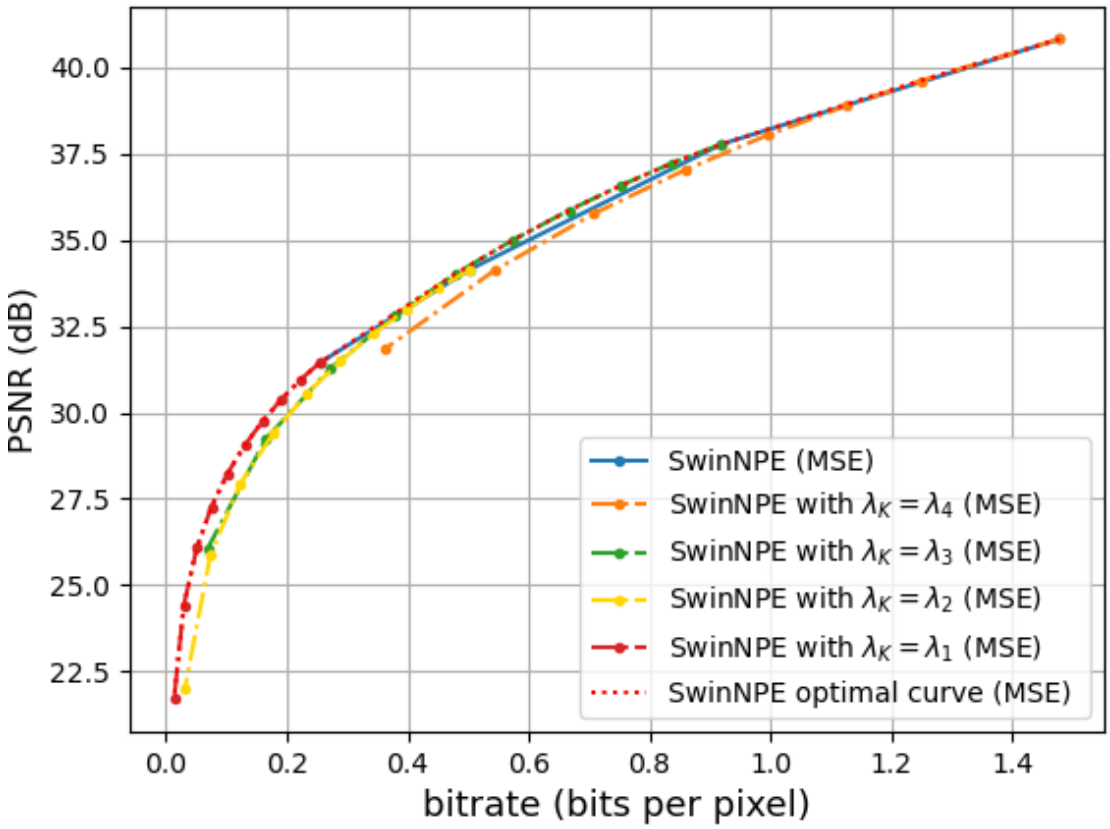
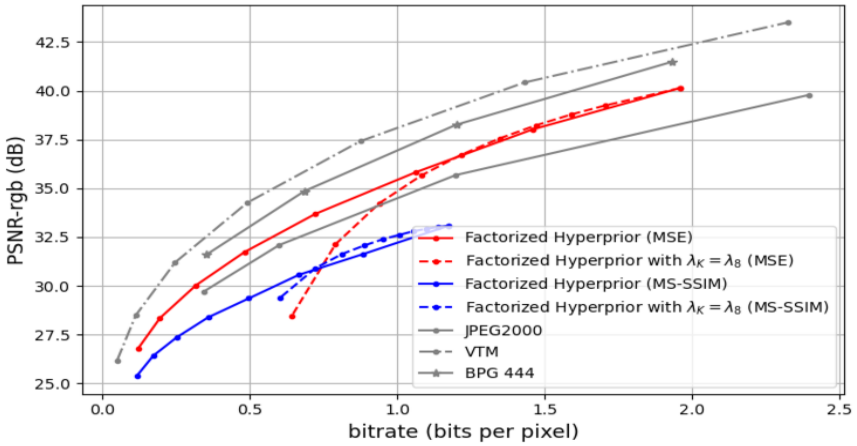
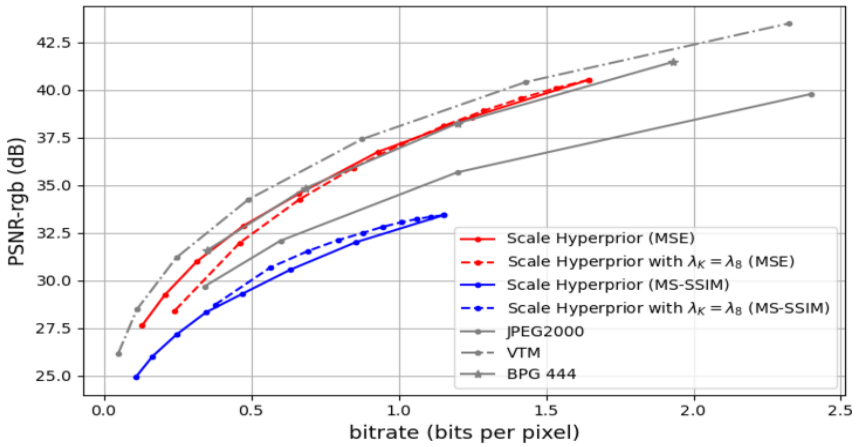
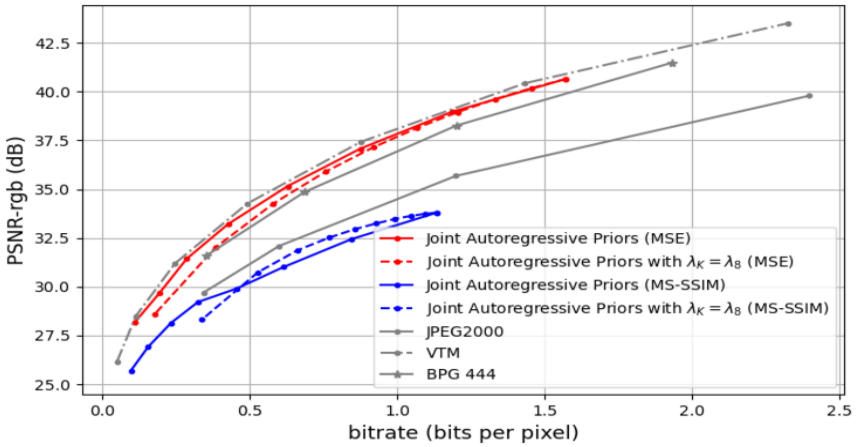


Figure 5.3: The dotted curve, SwinNPE optimal curve, represents the envelope of the four rate-distortion curves obtained when successively exploiting the four SwinNPE trained models (i.e. $\lambda_K = \lambda_1$, $\lambda_K = \lambda_2$, $\lambda_K = \lambda_3$, $\lambda_K = \lambda_4$) with different values of the scaling factor s on the Kodak dataset [11].







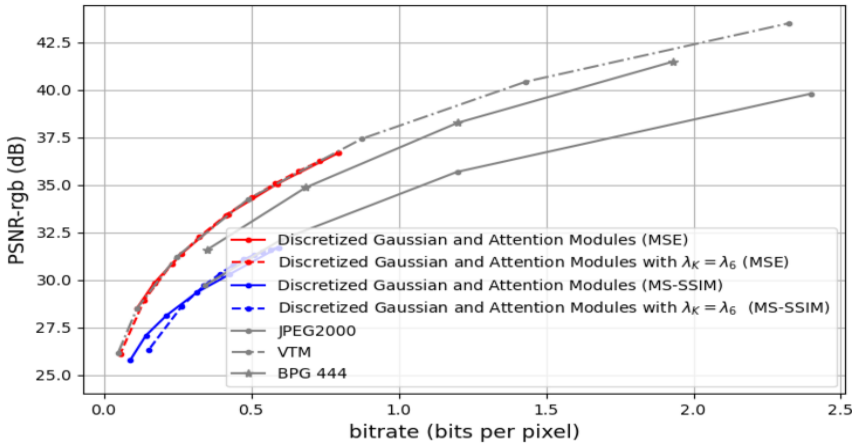


Figure 5.4: Rate-distortion reference curves with different values of the scaling factor s on the Kodak dataset [11].

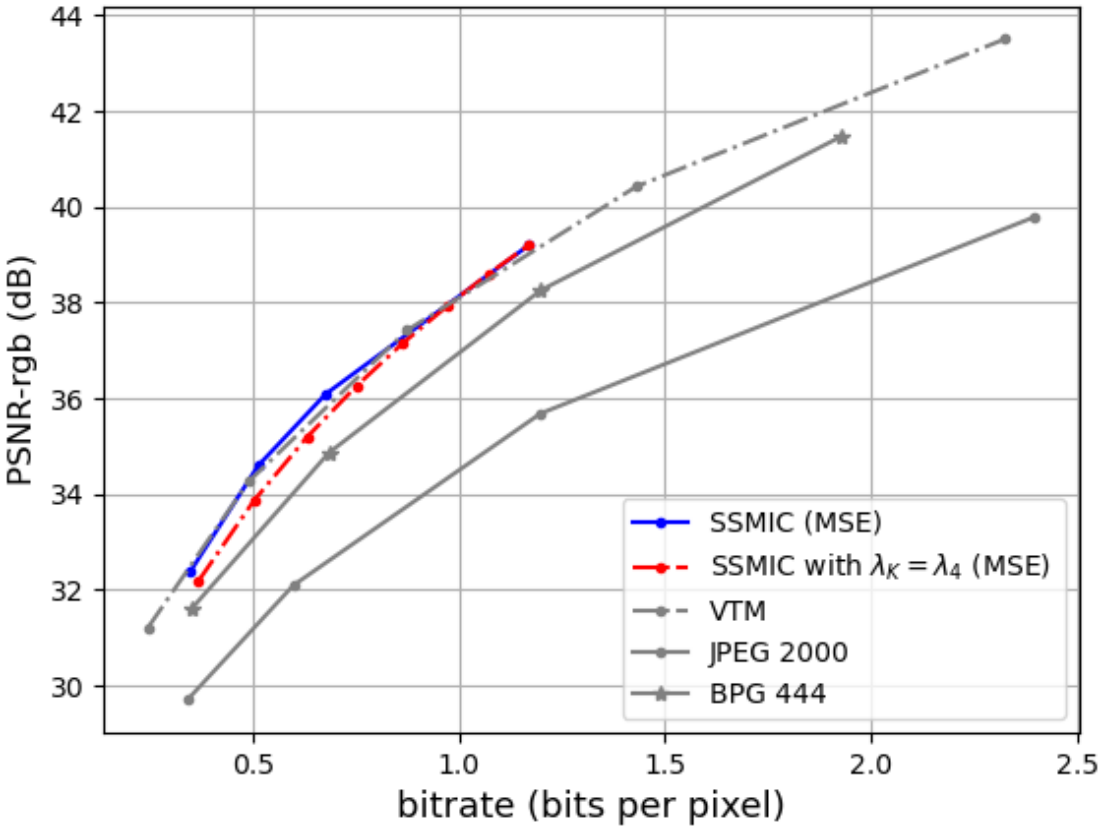
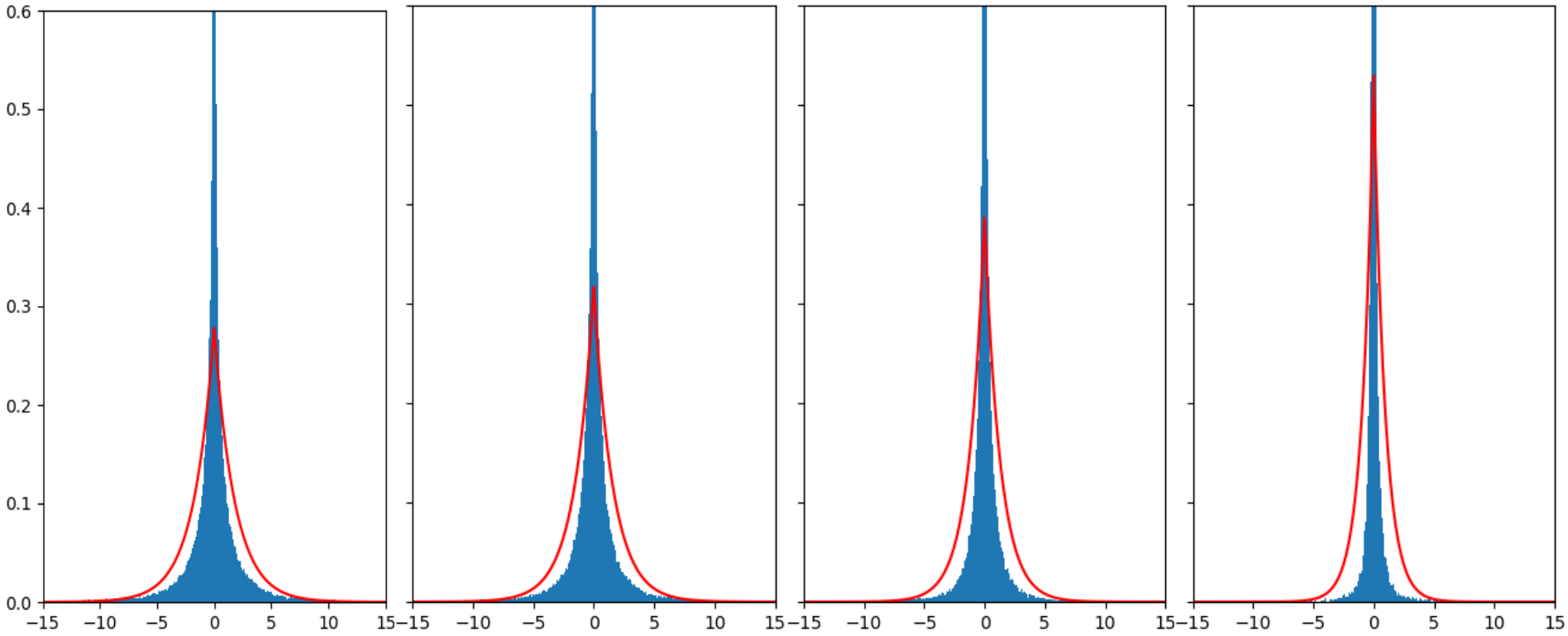


Figure 5.5: Rate-distortion reference curve of SSMIC (Section 4) and dotted curve depicts the rate-distortion achieved by employing a single regularization point on Kodak dataset [11].



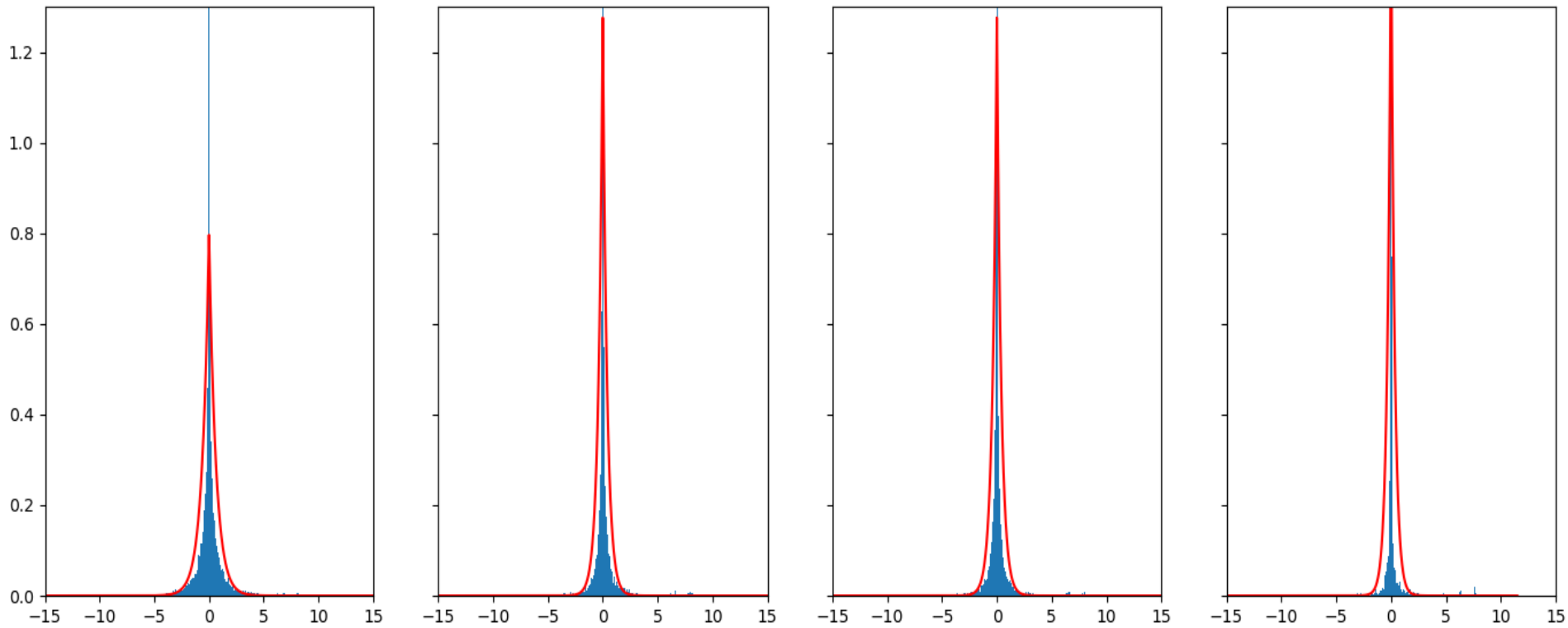
$\mu = -0.00275, b = 1.80,$
BPP=1.558
(a) $s = 1.0$ (without scale factor)

$\mu = -0.00318, b = 1.58,$
BPP=1.298
(b) $s = 0.8$

$\mu = -0.00330, b = 1.29,$
BPP=0.822
(c) $s = 0.5$

$\mu = -0.00197, b = 0.95,$
BPP=0.244
(d) $s = 0.2$

Figure 5.6: Normalized histograms of the latent space of *"kodim01.png"* image in Kodak dataset [11] with SwinNPE using three scaling factors $s = 0.2$, $s = 0.5$, and $s = 0.8$. The red curve shows the envelope of the estimated Laplacian distribution characterized by its mean μ and scale parameter b.



$\mu = 0.00066, b = 0.63,$
BPP=1.182
(a) $s = 1.0$ (without scale factor)

$\mu = 0.00049, b = 0.39,$
BPP=0.951
(b) $s = 0.8$

$\mu = 0.00049, b = 0.39,$
BPP=0.540
(c) $s = 0.5$

$\mu = -0.00011, b = 0.32,$
BPP=0.133
(d) $s = 0.2$

Figure 5.7: Normalized histograms of the latent space of *"kodim01.png"* image in Kodak dataset [11] with SSMIC using three scaling factors $s = 0.2$, $s = 0.5$, and $s = 0.8$. The red curve shows the envelope of the estimated Laplacian distribution characterized by its mean μ and scale parameter b.

5.4. Conclusion

This chapter proposed a universal end-to-end VAE for lossy Image Compression using a single image codec specifically designed to operate at variable bitrates. Our innovative strategy leverages a single trained VAE model to dynamically adjust the input image scale during the inference process, leading to an efficient rate-distortion mechanism. Our approach is applicable to various VAE-based image compression frameworks, including those using CNNs and attention mechanisms. Through several experiments, we have shown that this approach has the potential to enhance variable-rate neural network image compression techniques by minimizing complexity and memory usage. Future research will focus on evaluating our method using perceptual metrics and comparing it with other perceptual VAE-based approaches across diverse datasets.

6. Conclusion and Future Work

With the growing interest in vision-based applications, this thesis is primarily dedicated to developing an innovative framework for image compression that achieves high compression efficiency while minimizing computational complexity and latency. This makes it well-suited for real-world applications and resource-constrained environments. More specifically, driven by the advancements in deep neural network techniques, including convolutional and transformer models, as well as the integration of State Space Models (SSMs), our primary objective was to combine these powerful tools to create a robust and practical framework for image compression.

a) Summary of Contributions – In the following, we outline the key contributions of this work and discuss potential future extensions.

- Our first contribution is the development of a transformer-based image compression model, SwinNPE, which utilizes convolutional Swin blocks without positional encoding. SwinNPE achieves performance on par with state-of-the-art methods while requiring fewer model parameters and surpassing CNN-based architectures. The design of the convolutional Swin block enhances the ability to leverage spatial context without relying on positional encoding, leading to increased flexibility and reduced parameter count.

- The second contribution focused on the State Space Models (SSMs). We proposed a State Space Model-based Image Compression (SSMIC) approach

that leverages the strengths of SSMs, as derived from the Mamba model. This approach enhances contextual reasoning while effectively managing computational and memory requirements. As a result, SSMIC achieves competitive performance while significantly reducing computational complexity and latency, making it a promising candidate for high-quality real-time visual data compression with further optimizations.

- Our final contribution centers on variable bitrate image compression frameworks. We introduced a universal end-to-end VAE for lossy image compression, presenting a flexible and practical solution for variable-rate lossy image compression that can be applied to various VAE-based models. This approach eliminates the need to retrain the model for different compression rates, significantly enhancing the usability of compression systems across diverse scenarios while ensuring efficiency.

b) Perspectives – Here are some suggested future directions.

- It would be valuable to investigate the integration of diverse convolutional architectures, such as ConvNeXT, into the SwinNPE model. This exploration could facilitate more precise modeling of complex spatial relationships and patterns, potentially enhancing image compression performance without sacrificing the computational complexity.

- A promising direction to explore is the development of hybrid methods that combine convolutional architectures with state-space models. Convolutional models are highly effective at capturing local dependencies, while state-space models, paired with selective scan paths, can efficiently capture both local and global dependencies. This hybrid approach could improve

the factorization of pixel dependencies and provide richer spatial context, leading to better compression outcomes.

- In SSMIC, we utilized a selective scan approach that adapts an input-dependent selection mechanism for vision tasks without being restricted to any specific task. An interesting avenue for future research is to tailor this selective scanning method specifically for image compression. By proposing optimized traversal paths, we aim to enhance the factorization of dependencies among pixels, reduce redundancy, and maintain a lightweight architecture.

- Moreover, SSMIC shows promising potential for video compression as well. By extending the selective scanning method to handle residual or temporal correlations, we can more effectively exploit the redundancies between consecutive frames in video sequences. This approach could lead to more efficient compression by dynamically adjusting the scan paths based on both spatial and temporal dependencies, which would optimize the overall compression process for video data.

- Another intriguing research direction is studying the performance of the universal end-to-end VAE for lossy image compression across various perceptual and deep learning-based metrics. It would be particularly interesting to investigate how this universal approach performs in video compression where temporal correlations are incorporated, potentially opening new avenues for next-generation video compression techniques.

Bibliography

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, Curran Associates, Inc., 2017.

[2] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learned image compression with discretized gaussian mixture likelihoods and attention modules," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7936–7945, 2020.

[3] J. Liu, H. Sun, and J. Katto, "Learned image compression with mixed transformer-cnn architectures," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14388–14397, 2023.

[4] T. George, M. O. Sean, H. Sung Jin, V. Damien, M. David, B. Shumeet, C. Michele, and S. Rahul, "Variable rate image compression with recurrent neural networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[5] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9992–10002, 2021.

[6] Y. Qian, X. Sun, M. Lin, Z. Tan, and R. Jin, "Entroformer: A transformer-based entropy model for learned image compression," in *International Conference on Learning Representations (ICLR)*, 2022.

[7] Y. Zhu, Y. Yang, and T. Cohen, "Transformer-based transform coding," in *International Conference on Learning Representations (ICLR)*, 2022.

[8] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimization of nonlinear transform codes for perceptual quality," in *Picture Coding Symposium (PCS)*, pp. 1–5, 2016.

[9] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *International Conference on Learning Representations (ICLR)*, 2018.

[10] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[11] R. Franzen, "Kodak lossless true color image suite," `http://r0k.us/graphics/kodak`, 1999.

[12] JPEG-AI, "Jpeg-ai test images," `https://jpegai.github.io/test_images/`, 2020.

[13] Y. Liu, Y. Tian, Y. Zhao, H. Yu, L. Xie, Y. Wang, Q. Ye, J. Jiao, and Y. Liu, "VMamba: Visual state space model," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[14] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G. Sullivan, and J.-R. Ohm, "Overview of the versatile video coding (vvc) standard and its applica-

tions," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 31, pp. 3736–3764, 10 2021.

[15] G. Toderici, W. Shi, R. Timofte, L. Theis, J. Balle, E. Agustsson, N. Johnston, and F. Mentzer, "Workshop and challenge on learned image compression (clic2020)," 2020.

[16] Z. Cui, J. Wang, S. Gao, T. Guo, Y. Feng, and B. Bai, "Asymmetric gained deep image compression with continuous rate adaptation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10527–10536, 2021.

[17] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *ArXiv*, vol. abs/2312.00752, 2023.

[18] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Transactions on Image Processing (TIP)*, vol. 1, pp. 205–220, April 1992.

[19] S. Mallat and F. Falzon, "Analysis of low bit rate image transform coding," *IEEE Transactions on Signal Processing (TSP)*, vol. 46, no. 4, pp. 1027–1042, 1998.

[20] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological review*, vol. 65(6), pp. 386–408, 1958.

[21] O. T. Ilya, N. Houlsby, K. Alexander, B. Lucas, Z. Xiaohua, U. Thomas, Y. Jessica, K. Daniel, U. Jakob, L. Mario, and D. Alexey, "Mlp-mixer: An all-mlp architecture for vision," in *Neural Information Processing Systems (NeurIPS)*, 2021.

[22] H. B. Pedro, Z. Oleksandr, and T. João, Paulo, "A covid-19 time series forecasting model based on mlp ann," *Procedia Computer Science*, vol. 181, pp. 940–947, 2021.

[23] J. Li, B. Li, J. Xu, R. Xiong, and W. Gao, "Fully connected network-based intra prediction for image coding," *IEEE Transactions on Image Processing (TIP)*, vol. 27, no. 7, pp. 3236–3247, 2018.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[25] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.

[26] V. L. Ballé, Johannes and E. P. Simoncelli, "Density modeling of images using a generalized normalization transformation," in *International Conference for Learning Representations (ICLR)*, 2016.

[27] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587, 2014.

[28] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "(1986) de rumelhart, ge hinton, and rj williams, learning internal representations by error propagation, parallel distributed processing: Explorations in the microstructures of cognition, vol. i, de rumelhart and jl mcclelland (eds.) cambridge, ma: Mit press, pp. 318-362," 1988.

[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[31] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Association for Computational Linguistics, Oct. 2014.

[32] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. rahman Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Annual Meeting of the Association for Computational Linguistics*, 2019.

[33] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, 2021.

[34] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision (ECCV) 2020*, pp. 213–229, Springer International Publishing, 2020.

[35] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[36] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *North American Chapter of the Association for Computational Linguistics*, 2018.

[37] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," *ArXiv preprint ArXiv:2111.00396*, 2021.

[38] L. Zhu, B. Liao, Q. Zhang, X. Wang, W. Liu, and X. Wang, "Vision mamba: Efficient visual representation learning with bidirectional state space model," *ArXiv*, vol. abs/2401.09417, 2024.

[39] Y. Qiao, Z. Yu, L. Guo, S. Chen, Z. Zhao, M. Sun, Q. Wu, and J. Liu, "Vl-mamba: Exploring state space models for multimodal learning," *ArXiv*, vol. abs/2403.13600, 2024.

[40] G. Chen, Y. Huang, J. Xu, B. Pei, Z. Chen, Z. Li, J. Wang, K. Li, T. Lu, and L. Wang, "Video mamba suite: State space model as a versatile alternative for video understanding," *ArXiv*, vol. abs/2403.09626, 2024.

[41] H. Guo, J. Li, T. Dai, Z. Ouyang, X. Ren, and S.-T. Xia, "Mambair: A simple baseline fornbsp;image restoration withnbsp;state-space model," in *European Conference on Computer Vision (ECCV) 2024, Proceedings*, p. 222–241, Springer-Verlag, 2024.

[42] J. Ma, F. Li, and B. Wang, "U-mamba: Enhancing long-range dependency for biomedical image segmentation," *ArXiv*, vol. abs/2401.04722, 2024.

[43] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," in *International Conference on Learning Representations (ICLR)*, 2017.

[44] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *International Conference on Learning Representations (ICLR)*, 2017.

[45] Y. Fei, H. Luis, C. Yongmei, and M. Mikhail G., "Slimmable compressive autoencoders for practical neural image compression," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[46] T. Chen, H. Liu, Z. Ma, Q. Shen, X. Cao, and Y. Wang, "End-to-end learnt image compression via non-local attention optimization and improved context modeling," *IEEE Transactions on Image Processing (TIP)*, vol. 30, pp. 3179–3191, 2021.

[47] Y. Patel, S. Appalaraju, and R. Manmatha, "Saliency driven perceptual image compression," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 227–236, 2021.

[48] L. Zhou, Z. Sun, X. Wu, and J. Wu, "End-to-end optimized image compression with attention mechanism," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.

[49] Y. Zhang, K. Li, K. Li, B. Zhong, and Y. Fu, "Residual non-local attention networks for image restoration," in *International Conference on Learning Representations (ICLR)*, 2019.

[50] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems (NeurIPS)*, NIPS'16, p. 4905–4913, Curran Associates Inc., 2016.

[51] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7794–7803, 2018.

[52] V. Hajihashemi, H. E. Najafabadi, A. A. Gharahbagh, H. Leung, M. Yousefan, and J. M. R. S. Tavares, "A novel high-efficiency holography image compression method, based on HEVC, wavelet, and nearest-neighbor interpolation," *Multimedia Tools and Applications*, vol. 80, p. 31953–31966, 2021.

[53] A. B. Koyuncu, H. Gao, A. Boev, G. Gaikov, E. Alshina, and E. Steinbach, *Contextformer: A Transformer with Spatio-Channel Attention for Context Modeling in Learned Image Compression*, p. 447–463. Springer Nature Switzerland, 2022.

[54] R. Zou, C. Song, and Z. Zhang, "The devil is in the details: Window-based attention for image compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17492–17501, June 2022.

[55] S. Qin, J. Wang, Y. Zhou, B. Chen, T. Luo, B. An, T. Dai, S.-T. Xia, and Y. Wang, "Mambavc: Learned visual compression with selective state spaces," *ArXiv*, vol. abs/2405.15413, 2024.

[56] T. George, M. O. Sean, H. Sung Jin, V. Damien, M. David, B. Shumeet, C. Michele, and S. Rahul, "Full resolution image compression with recurrent neural networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[57] J. Nick, V. Damien, M. David, C. Michele, S. Saurabh, C. Troy, H. Sung Jin, S. Joel, and T. George, "Improved lossy image compression with priming and spatially adaptive bit rates for recurrent network," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[58] Y. Choi, M. El-Khamy, and J. Lee, "Variable rate deep image compression with a conditional autoencoder," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3146–3154, 2019.

[59] F. Yang, L. Herranz, J. v. d. Weijer, J. A. I. Guitián, A. M. López, and M. G. Mozerov, "Variable rate deep image compression with modulated autoencoder," *IEEE Signal Processing Letters (SPL)*, vol. 27, pp. 331–335, 2020.

[60] M. Akbari, J. Liang, J. Han, and C. Tu, "Learned variable-rate image compression with residual divisive normalization," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, 2020.

[61] L. Jooyoung, C. Seunghyun, and B. Seunghwa, "Context-adaptive entropy model for end-to-end optimized image compression," in *International Conference on Learning Representations (ICLR)*, 2019.

[62] A. Eirikur, M. Fabian, T. Michael, C. Lukas, T. Radu, B. Luca, and G. Luc Van, "Soft-to-hard vector quantization for end-to-end learning compressible representations," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[63] D. Minnen, G. Toderici, S. Singh, S. J. Hwang, and M. Covell, "Image-dependent local entropy models for learned image compression," in *IEEE International Conference on Image Processing (ICIP)*, pp. 430–434, 2018.

[64] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learning image and video compression through spatial-temporal energy compaction," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10063–10072, 2019.

[65] N. Ken, M. Shin-ichi, M. Takeru, and O. Daisuke, "Neural multi-scale image compression," in *Asian Conference on Computer Vision (ACCV)*, 2018.

[66] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang, "Learning convolutional networks for content-weighted image compression," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3214–3223, June 2018.

[67] J. Cai and L. Zhang, "Deep image compression with iterative non-uniform quantization," in *IEEE International Conference on Image Processing (ICIP)*, pp. 451–455, 2018.

[68] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *International Conference on Machine Learning (ICML)*, vol. 48, pp. 1747–1756, 2016.

[69] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with pixel-cnn decoders," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

[70] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. V. Gool, "Conditional probability models for deep image compression," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4394–4402, 2018.

[71] O. Rippel and L. Bourdev, "Real-time adaptive image compression," in *International Conference on Machine Learning (ICML)*, vol. 70, pp. 2922–2930, 2017.

[72] Z. Cui, J. Wang, B. Bai, T. Guo, and Y. Feng, "G-vae: A continuously variable rate deep image compression framework," *ArXiv*, vol. abs/2003.02012, 2020.

[73] J. Zhou, "Multi-scale and context-adaptive entropy model for image compression," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.

[74] Z. He, M. Huang, L. Luo, X. Yang, and C. Zhu, "Towards real-time practical image compression with lightweight attention," *Expert Systems with Applications*, vol. 252, p. 124142, 2024.

[75] H. Liu, T. Chen, Q. Shen, and Z. Ma, "Practical stacked non-local attention modules for image compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[76] D. Minnen and S. Singh, "Channel-wise autoregressive entropy models for learned image compression," in *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 3339–3343, IEEE, 2020.

[77] Y. Qian, Z. Tan, X. Sun, M. Lin, D. Li, Z. Sun, H. Li, and R. Jin, "Learning accurate entropy model with global reference for image compression," in *IEEE International Conference on Learning Representations (ICLR)*, pp. 1–17, May 2021.

[78] Z. Guo, Z. Zhang, R. Feng, and Z. Chen, "Causal contextual prediction for learned image compression," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 32, no. 4, pp. 2329–2341, 2021.

[79] D. He, Y. Zheng, B. Sun, Y. Wang, and H. Qin, "Checkerboard context model for efficient learned image compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14771–14780, 2021.

[80] A. B. Koyuncu, H. Gao, A. Boev, G. Gaikov, E. Alshina, and E. Steinbach, "Contextformer: A transformer with spatio-channel attention for context modeling in learned image compression," in *European Conference on Computer Vision (ECCV)*, pp. 447–463, Springer, 2022.

[81] W. Jiang, J. Yang, Y. Zhai, P. Ning, F. Gao, and R. Wang, "Mlic: Multi-reference entropy model for learned image compression," in *Proceedings of the 31st ACM International Conference on Multimedia*, MM '23, ACM, Oct. 2023.

[82] D. He, Z. Yang, W. Peng, R. Ma, H. Qin, and Y. Wang, "Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5708–5717, 2022.

[83] D. Liu, H. Ma, Z. Xiong, and F. Wu, "Cnn-based dct-like transform for image compression," in *MultiMedia Modeling*, pp. 61–72, Springer International Publishing, 2018.

[84] E. Ahanonu, M. Marcellin, and A. Bilgin, "Lossless image compression using reversible integer wavelet transforms and convolutional neural networks," in *Data Compression Conference (DCC)*, p. 1, March 2018.

[85] P. Akyazi and T. Ebrahimi, "Learning-based image compression using convolutional autoencoder and wavelet decomposition," in *Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, p. 5, June 2019.

[86] X. Li, A. Naman, and D. Taubman, "A neural network lifting based secondary transform for improved fully scalable image compression in jpeg 2000," in *IEEE International Conference on Image Processing (ICIP)*, pp. 1606–1610, 2022.

[87] H. Ma, D. Liu, R. Xiong, and F. Wu, "iwave: Cnn-based wavelet-like transform for image compression," *IEEE Transactions on Multimedia (TMM)*, vol. 22, no. 7, pp. 1667–1679, 2020.

[88] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 22, no. 12, pp. 1649–1668, 2012.

[89] H. Yueyu, Y. Wenhan, L. Mading, , and L. Jiaying, "Progressive spatial recurrent neural network for intra prediction," *IEEE Transactions on Multimedia (TMM)*, vol. 21, no. 12, p. 3024–3037, 2019.

[90] W. Pratt, J. Kane, and H. Andrews, "Hadamard transform image coding," *Proceedings of the IEEE*, vol. 57, no. 1, pp. 58–68, 1969.

[91] W. Cui, T. Zhang, S. Zhang, F. Jiang, W. Zuo, Z. Wan, and D. Zhao, "Convolutional neural networks based intra prediction for hevc," in *Data Compression Conference (DCC)*, pp. 436–436, 2017.

[92] T. Dumas, A. Roumy, and C. Guillemot, "Context-adaptive neural network-based prediction for image compression," *IEEE Transactions on Image Processing (TIP)*, vol. 29, pp. 679–693, 2020.

[93] Y. Li, Y. Yi, D. Liu, L. Li, Z. Li, and H. Li, "Neural-network-based cross-channel intra prediction," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 17, no. 3, 2021.

[94] I. Schiopu, Y. Liu, and A. Munteanu, "Cnn-based prediction for lossless coding of photographic images," in *Picture Coding Symposium (PCS)*, pp. 16–20, 2018.

[95] M. Marcellin, M. Gormish, A. Bilgin, and M. Boliek, "Overview of jpeg-2000," *Data Compression Conference Proceedings (DCC)*, 05 2000.

[96] I. Schiopu and A. Munteanu, "Deep-learning-based lossless image coding," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 30, no. 7, pp. 1829–1842, 2020.

[97] I. Schiopu, H. Huang, and A. Munteanu, "CNN-based intra-prediction for lossless HEVC," *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, vol. 30, no. 7, pp. 1816–1828, 2020.

[98] L. Wang, A. Fiandrotti, A. Purica, G. Valenzise, and M. Cagnazzo, "Enhancing hevc spatial prediction by context-based learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4035–4039, 2019.

[99] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, "Cvt: Introducing convolutions to vision transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22–31, October 2021.

[100] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 764–773, 2017.

[101] L. Chi, B. Jiang, and Y. Mu, "Fast fourier convolution," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 4479–4488, 2020.

[102] L. Beyer, X. Zhai, A. Royer, L. Markeeva, R. Anil, and A. Kolesnikov, "Knowledge distillation: A good teacher is patient and consistent," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2022.

[103] W. Guo-Hua, J. Li, B. Li, and Y. Lu, "Evc: Towards real-time neural image compression with mask decay," in *The Eleventh International Conference on Learning Representations (ICLR)*, 2022.

[104] G. Wu, W.-S. Zheng, Y. Lu, and Q. Tian, "Pslt: a light-weight vision transformer with ladder self-attention and progressive shift," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2023.

[105] S. Mehta and M. Rastegari, "Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer," in *International Conference on Learning Representations (ICLR)*, 2022.

[106] Y. Yang and S. Mandt, "Computationally-efficient neural image compression with shallow decoders," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 530–540, 2023.

[107] M. S. Ali, Y. Kim, M. Qamar, S.-C. Lim, D. Kim, C. Zhang, S.-H. Bae, and H. Y. Kim, "Towards efficient image compression without autoregressive models," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2024.

[108] A. Gu, I. Johnson, K. Goel, K. Saab, T. Dao, A. Rudra, and C. Ré, "Combining recurrent, convolutional, and continuous-time models with linear state space layers," *Advances in neural information processing systems (NeurIPS)*, vol. 34, pp. 572–585, 2021.

[109] R. Xu, S. Yang, Y. Wang, Y. Cai, B. Du, and H. Chen, "Visual mamba: A survey and new outlooks," *ArXiv preprint ArXiv:2404.18861*, 2024.

[110] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *ArXiv*, vol. abs/1710.05941, 2018.

[111] B. Zhang and R. Sennrich, "Root mean square layer normalization," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[112] J. Bégaint, F. Racap'e, S. Feltman, and A. Pushparaja, "Compressai: a pytorch library and evaluation platform for end-to-end compression research," *ArXiv*, vol. abs/2011.03029, 2020.

[113] F. Bellard, "Bpg image format," `http://bellard.org/bpg/`, 2018.

[114] N. D. Guerin, R. C. da Silva, M. C. de Oliveira, H. C. Jung, L. G. R. Martins, E. Peixoto, B. Macchiavello, E. M. Hung, V. Testoni, and P. G.

Freitas, "Rate-constrained learning-based image compression," *Signal Processing: Image Communication, Elsevier*, vol. 101, p. 116544, 2022.

A. Appendix

A.1. Spatial Correlation of Latent

In this Appendix, we provide visualizations of the spatial correlation maps for SwinNPE (in Chapter 3), SSMIC (in Chapter 4), and SSMIC_CW (in Chapter 4) at different λ, as shown in Figures A.1, A.2, and A.3 respectively.

One can observe that the correlations in SSMIC_CW are lower than those in SSMIC. Additionally, in the spatial correlation for SwinNPE, the larger the λ value, the lower the correlation in the latent space.

A.2. Image Compression

BPG444: We get BPG software from http://bellard.org/bpg/ and use command as follows:

```
bpgenc -e x265 -q [quality] -f 444
-o [encoded bitstream file] [input image file]
bpgdec -o [output image file] [encoded bitstream file]
```

VTM-15.0: VTM is sourced from https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM. The command is:

```
VVCSoftware_VTM/bin/EncoderAppStatic -i [input YUV file]
-c [config file]
-q [quality] -o /dev/null -b [encoded bitstream file]
```

```
-wdt 1976 -hpt 1312 -fr 1 -f 1
--InputChromaFormat=444 --InputBitDepth=8 --ConformanceWindowMode=1

VVCSoftware_VTM/bin/DecoderAppStatic -b [encoded bitstream file]
-o [output YUV file] -d 8
```

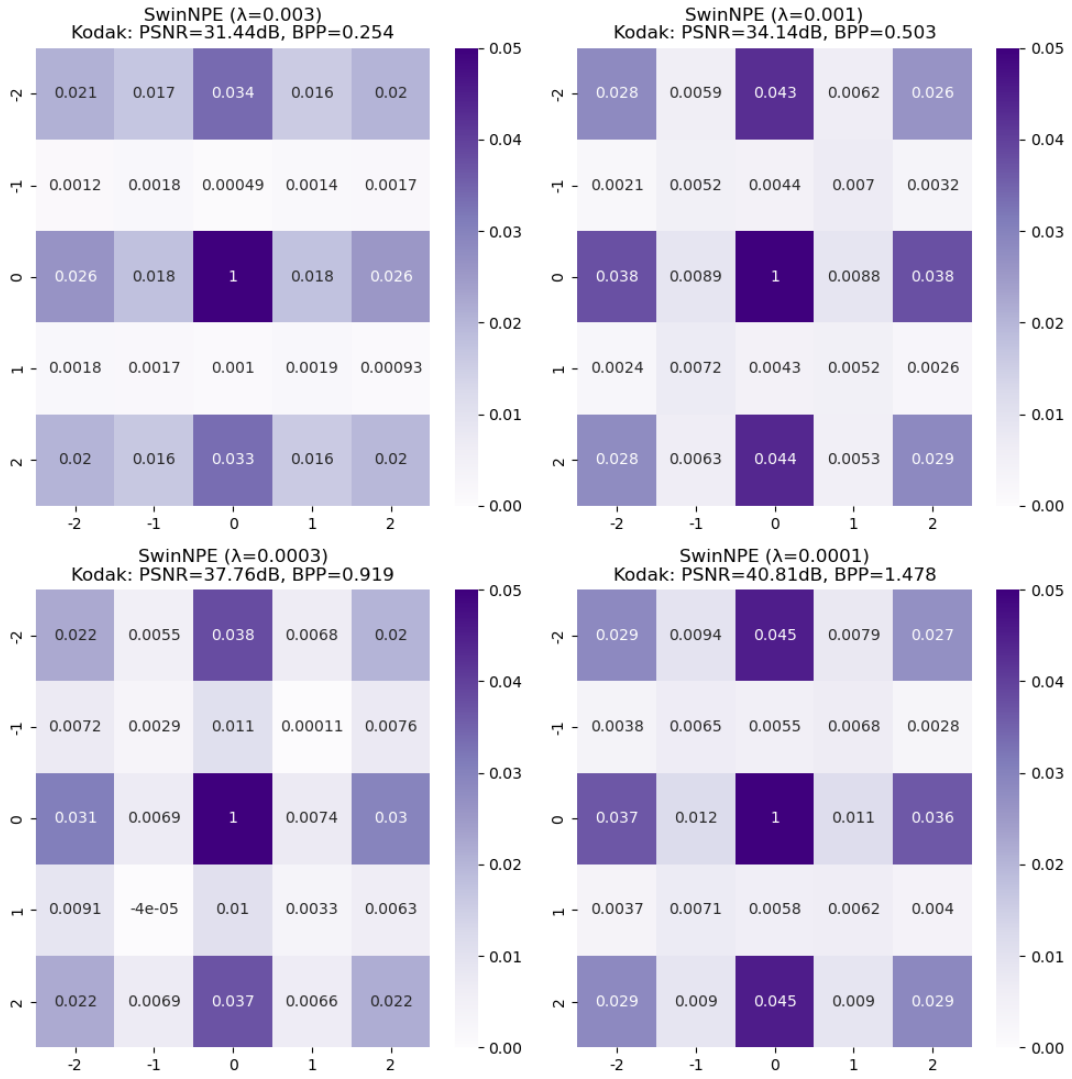


Figure A.1: Average of spatial correlations of all images on Kodak [11] with SwinNPE 3 in different λ.

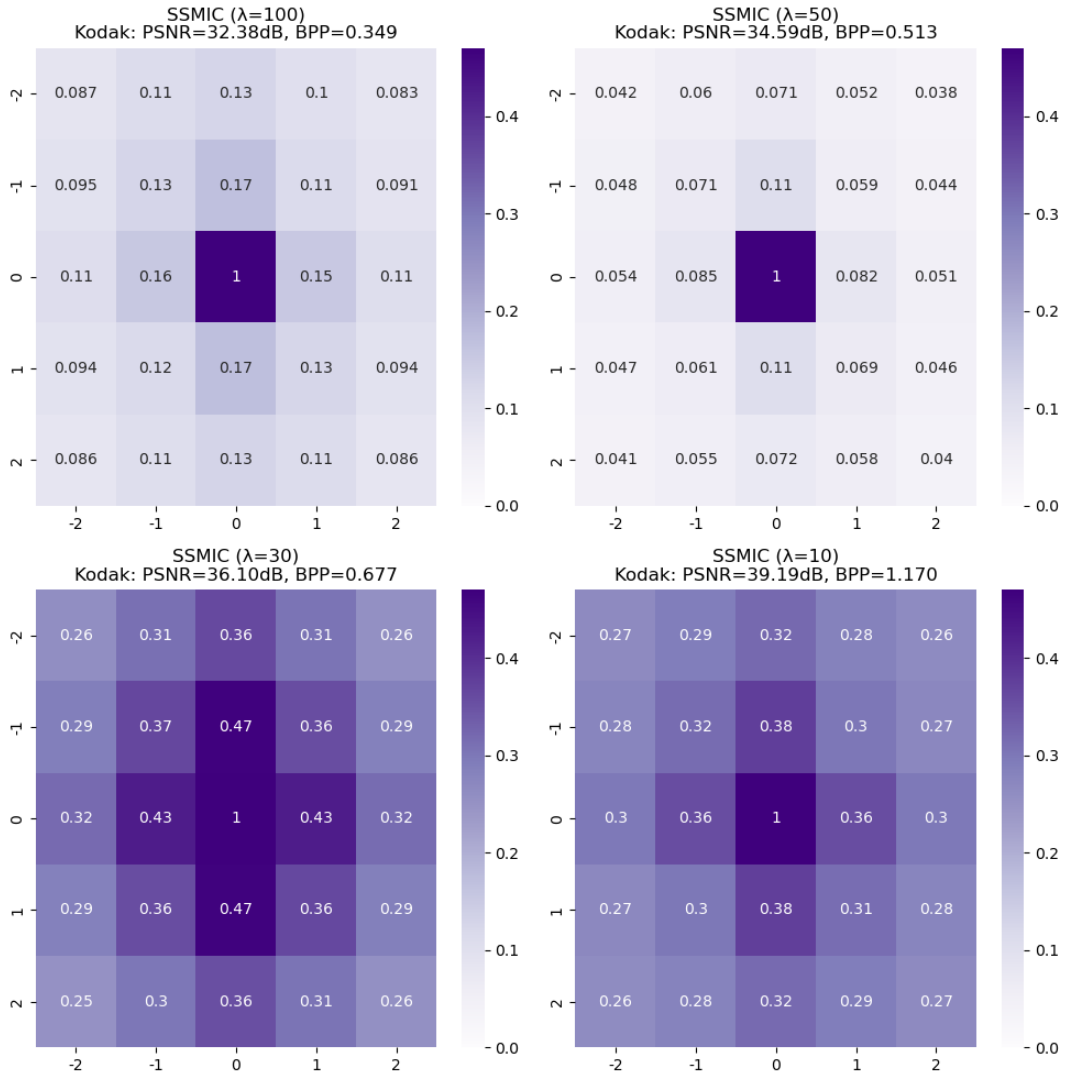


Figure A.2: Average of spatial correlations of all images on Kodak [11] with SSMIC 4 in different λ.

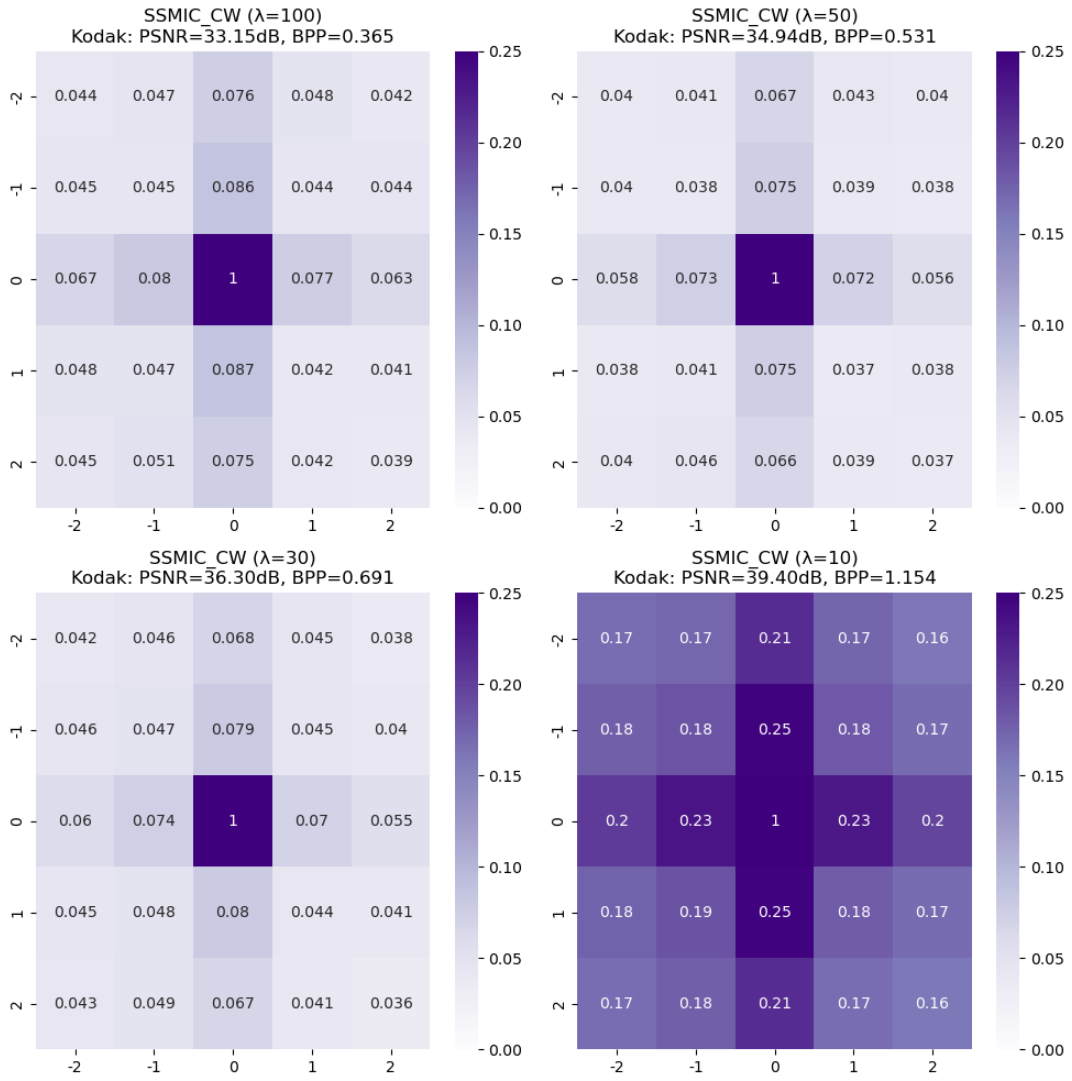


Figure A.3: Average of spatial correlations of all images on Kodak [11] with SSMIC_CW 4 in different λ.