Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Autoencoder-Based Image Compression with Target Bitrate Constraint

Nilson Donizete Guerin Júnior

Thesis presented as a requirement for the
completion of Doctorate in Compute Science

Advisor
Prof. Bruno Luiggi Macchiavello Espinoza, Dr.

Co-Advisor
Renam Castro da Silva, Dr.

Brasília
2024

Ficha Catalográfica de Teses e Dissertações

Está página existe apenas para indicar onde a ficha catalográfica gerada para dissertações de mestrado e teses de doutorado defendidas na UnB. A Biblioteca Central é responsável pela ficha, mais informações nos sítios:

http://www.bce.unb.br
http://www.bce.unb.br/elaboracao-de-fichas-catalograficas-de-teses-e-dissertacoes

Esta página não deve ser inclusa na versão final do texto.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Autoencoder-Based Image Compression with Target Bitrate Constraint

Nilson Donizete Guerin Júnior

Thesis presented as a partial requirement for the
completion of Doctorate in Compute Science

Prof. Bruno Luiggi Macchiavello Espinoza, Dr. (Advisor)
CiC/UnB

Prof. Eduardo Antônio Barros da Silva, Dr.
COPPE/PEE/Poli/DEL/UFRJ

Prof. Ricardo Lopes de Queiroz, Dr.
CiC/IE/UnB

Alexandre Zaghetto, Dr.
Sony's R&D Center US Laboratory

Prof. Rodrigo Bonifácio de Almeida, Dr.
Coordinator of the Graduate Program in Compute Science

Brasília, June 21, 2024

The present work was carried out with the support of the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES)*, through the Access to the Periodicals Portal.

Abstract

Learning-based image compression is emerging as a competitive alternative to conventional image coding techniques. Neural image coding has advanced significantly, evolving from struggling to match classical codecs to often surpassing them. Techniques such as variational autoencoders and recurrent neural networks have shown promise in optimizing the rate-distortion trade-off while preserving image content. Rate control is a critical feature, often a requirement for several still image coding applications. Achieving rate control for every input with minimal impact on rate-distortion performance remains challenging. Typically, learning-based lossy codecs need multiple trained models for different quality requirements. Although initiatives have aimed to enhance model flexibility by incorporating various rate-distortion points, the problem of consistent rate control—where a model achieves a specific rate across all compressed images—remains underexplored and poorly understood. This work proposes a non-constrained solution to the constrained problem of training a learning-based image codec for a specific bitrate. The solution involves modifying the loss function for autoencoder optimization. Additionally, inspired by reinforcement learning, a temporal-adaptive approach is introduced, which incorporates temporal behavior into the loss function, making the training process more robust against optimization challenges. Experiments conducted on the Kodak and JPEG AI datasets demonstrate that autoencoders trained with the proposed loss functions can achieve rate-constrained encoding with negligible losses in Structural Similarity Index Measure (SSIM) and Multi-scale Structural Similarity Index Measure (MS-SSIM). Some deterioration in peak signal-to-noise ratio (PSNR) is observed compared to the variational baseline architectures. However, this trade-off is expected, as restricted optimization scenarios are inherently more challenging than unrestricted ones.

Keywords: image coding; neural networks; rate-distortion; rate control; neural compression; reinforcement learning; loss-adaptive parameters

Compressão de Imagens com Controle de Taxa Baseado em Autoencoders

Resumo

A compressão de imagens baseada em aprendizado tem se tornado uma alternativa promissora às técnicas tradicionais de codificação. Os codecs neurais evoluíram rapidamente, superando muitas vezes os métodos clássicos. Abordagens como autoencoders variacionais e redes neurais recorrentes têm demonstrado eficiência na otimização do equilíbrio entre taxa de compressão e qualidade da imagem. O controle de taxa é uma necessidade em várias aplicações de codificação de imagens. No entanto, alcançar esse controle de forma consistente e com impacto mínimo na qualidade da imagem ainda é um desafio. Geralmente, codecs com perdas precisam de vários modelos treinados para diferentes níveis de qualidade. Apesar de avanços para tornar os modelos mais flexíveis, permitindo múltiplos pontos de taxa-distorção, o problema de controle de taxa consistente — onde um único modelo entrega a taxa desejada para qualquer imagem — é pouco explorado. Neste trabalho, propomos uma solução eficiente para realizar controle de taxa em um único modelo baseado em aprendizado. Nossa abordagem modifica a função de perda do autoencoder durante o treinamento. Além disso, inspirados pela área de aprendizado por reforço, adicionamos uma estratégia temporal-adaptativa, que incorpora ajustes dinâmicos ao longo do tempo, tornando o treinamento mais robusto. Os resultados em bases de dados como Kodak e JPEG AI mostram que nossos modelos atingem controle de taxa com perdas mínimas nas métricas Índice Estrutural de Similaridade (SSIM, do inglês Structural Similarity Index) e Índice Estrutural de Similaridade Multi-Escala (MS-SSIM, do inglês Multi-Scale Structural Similarity Index). Observamos uma leve redução na Razão Pico-Sinal-Ruído (PSNR, do inglês Peak Signal-to-Noise Ratio) em comparação com modelos variacionais tradicionais que otimizam como perda a função de taxa-distorção.

Palavras-chave: codificação de imagens; redes neurais; taxa-distorção; controle de taxa; compressão neural; aprendizado por reforço; parâmetros adaptativos de perda

Contents

List of Figures

List of Tables

Acronyms

BPG Better Portable Graphics.

bpp bits per pixel.

CABAC context-adaptive binary arithmetic coding.

CDF cumulative distribution function.

CODEC Coder-decoder.

DC direct current.

DCT discrete cosine transform.

DPCM differential pulse code modulation.

EBCOT Embedded Block Coding with Optimal Truncation.

ELBO evidence lower bound.

FLIF Free Lossless Image Format.

GDN generalized divisive normalization.

GRU gated recurrent unit.

HD high definition.

HDR high dynamic range.

HEVC High Efficiency Video Coding.

iid independently and identically distributed.

JPEG Joint Photographic Experts Group.

KL Kullback-Leibler.

LSTM long short-term memory.

MAP maximum a posteriori.

ML maximum likelihood.

MLFF Multi-layer Feedforward.

MS-SSIM Multi-scale Structural Similarity Index Measure.

nat natural unit of information.

PDF probability density function.

PMF probability mass function.

PSNR peak signal-to-noise ratio.

RL Reinforcement Learning.

SGD stochastic gradient descent.

SSIM Structural Similarity Index Measure.

UHD ultra high definition.

VAE variational autoencoder.

VTM VVC Test Model.

VVC Versatile Video Coding.

Glossary

C_T Transform Coefficients.

D Distortion Measure.

F Cumulative Distribution Function.

I Activation function.

J Cost Function.

P Probability Measure.

Q Quantization operator.

R^{est} Entropy-estimated Rate.

R^{param} Rate parameter.

R_t^{c1} Generalized parabola coefficients at time t.

R_t^{c2} Generalized parabola coefficients at time t.

R_t^{d1} Generalized parabola coefficients at time t.

R_t^{d2} Generalized parabola coefficients at time t.

R_t^{ema} EMA Smoothed Rate parameter at time t.

R_t^{est} Entropy-estimated Rate at time t.

R_t^{param} Rate parameter at time t.

R_t^{shift} Shift estimation of Rate Parameter at time t.

$ReLU$ Rectified Linear Unit.

S Sample Space.

$SiLU$ Sigmoid Linear Unit.

T Transfer function.

T_M Transform Matrix.

V Original Pixel Values.

X Random Variable.

Y Random Variable.

Δ Parameter Variation or difference.

α Parameter or constant in functions.

β Parameter or constant in functions.

$\emptyset$ Impossible Event.

η Sigmoid Function.

exp Exponential Function.

γ Parameter or constant in functions.

$\hat{V}$ Predicted Pixel Value.

$\hat{\theta}_{ML}$ Maximum Likelihood.

$\hat{x}$ Quantized neural network variable.

$\hat{y}$ Quantized neural network variable.

λ Lagrange Multiplier.

λ' Generalized Lagrange Multiplier.

$\mathbb{E}$ Expectation.

$\mathbb{I}$ Identity Matrix.

$\mathbb{N}$ Set of Natural Numbers.

$\mathbb{R}$ Set of Real Numbers.

$\mathcal{A}$ Alphabet.

$\mathcal{F}$ Event Class.

$\mathcal{H}$ Shannon Entropy.

$\mathcal{I}$ Mutual Information.

$\mathcal{KL}$ Kullback-Leibler Divergence.

$\mathcal{L}$ Evidence Lower Bound.

$\mathcal{N}$ Gaussian Distribution.

$\mathcal{Q}$ Density Family.

$\mathcal{R}$ Set of Target Rates.

$\mathcal{T}$ Indexing Set.

ϕ Parameter.

σ^2 Variance.

θ Parameter.

θ^{rpar} Abbreviated set of parameters passed to r_t^p.

$\tilde{x}$ Noisy neural network variable.

$\tilde{y}$ Noisy neural network variable.

code Encoded Number within a specific numerical range in arithmetic coding.

ema_t Exponential moving average at time t.

f Function.

g Function.

h Hidden Layer/Representation.

high Track the upper bound of the current interval in arithmetic coder.

low Track the lower bound of the current interval in arithmetic coder.

p Distribution.

q Distribution.

r^{max} Maximum rate for the exponential decay.

r^{target} Target Rate.

r_t^p Desired-rate function parameter at time t, which controls exponential decay.

seq A sequence of symbols in arhtimetic coder.

t^{iter} Number of iterations for the exponential decay.

tanh Hyperbolic Tangent Function.

Chapter 1

Introduction

Image compression is crucial for digital communication, transmission, and storage [1]. Techniques based on transformations, context predictions, quantization methods, and entropy encoding are commonly used. These techniques have been successfully incorporated into image compression methods and utilized in various coding standards.

The initial methods for image compression used entropy compression techniques to reduce the statistical redundancy of images. The most well-known techniques include Huffman encoding, Golomb codes, and arithmetic coding [2, 3, 4]. Later, transform coding was proposed using Fourier and Hadamard transforms [5]. The Discrete Cosine Transform (DCT), proposed by Ahmed et al. [6], was a significant advancement because it compacts image energy much more efficiently than previous transforms [7].

Prediction and quantization techniques were proposed to reduce the spatial redundancy of images. For instance, JPEG-1, a DCT-based CODEC, has dominated lossy image compression since its introduction in 1992, employing both prediction and transformation techniques [8]. The CODEC initially divides the image into blocks and applies the DCT to these blocks. Differential Pulse Code Modulation (DPCM), a form of context prediction, is applied to the DC components of the transform, with encoding adopted only for the residuals. Additionally, a special quantization table is used to discard more details than the principal components of the image [7].

The standard evolved with its successor, JPEG 2000 [9]. This transform-based encoder has been used in cinema, medical imaging, and other areas. It uses a two-dimensional wavelet transform and an efficient arithmetic coding method, Embedded Block Coding with Optimal Truncation (EBCOT), to reduce the statistical redundancy of wavelet coefficients [10, 7].

The evolution of coding standards includes BPG (Better Portable Graphics) and JPEG XR [11, 12]. These standards aim to achieve better compression rates than JPEG-1

with a slight increase in algorithmic complexity. However, they have become obsolete, particularly with limitations for HDR images. To overcome these limitations, JPEG XT was launched to support HDR image compression and is compatible with previous standards [13]. More recently, the JPEG committee standardized JPEG XL, aiming for good rate-distortion performance for HD and UHD images, including functionalities for web distribution and various applications [14].

The brief history of the evolution of coding standards demonstrates the community's significant interest in improving image compression. Artificial neural networks are being considered for the task of compression. Recently, neural networks have been the state of the art in semantic tasks such as image classification and object detection. They have also shown high performance in low-level tasks, such as super-resolution and compression artifact reduction. The idea behind these approaches is to explore the hierarchical correlation between neighboring values using cascading operations [7].

The recent success of neural networks in image processing and computer vision has drawn the attention of the compression community. These algorithms are seen as potential enhancements for compression performance. These approaches aim to either improve components of classic standards using neural networks or completely replace the encoding process with neural networks. This work focuses on methods that use neural networks to replace the encoding process, known as end-to-end approaches.

Neural networks were expected to perform well in image compression. However, until recently, there was little evidence that training a competitive neural network for different images at different rates was possible. One of the first relevant approaches to using neural networks in image coding is based on a cascade of autoencoders [15]. In this strategy, each autoencoder compresses and reconstructs its input. The reconstruction error is then used as input for the next autoencoder. This process repeats until a specified number of autoencoders are reached.

This patch-based approach has served as a foundation for several proposals. These include adopting memory layers for progressive encoding, incorporating quality objectives for each image patch, and using neural networks for context prediction [16, 17, 18, 19]. Johnston et al. proposed optimized training strategies to enhance the memory capacity of recurrent networks [20].

The works described thus far optimize distortion during training and adopt various strategies to control the rate. These works have the advantage of controlling the rate produced by the network for each image patch, and the code space is progressive. Thus, a single neural network produces progressively better reconstructions at multiple controlled rates. However, using neural networks with memory proved to be very costly computationally. Moreover, the performance of these approaches only surpasses JPEG-1 and rivals

JPEG-2000, falling significantly behind other more powerful standards, such as BPG. Researchers began to adopt an explicit entropy model in the neural architecture to optimize distortion and the rate during neural network training, seeking to improve the end-to-end compression results described in the previous paragraph.

One of the first approaches to consider this optimization delivered state-of-the-art results at the time [21]. Beyond improving rate-distortion performance, the authors demonstrated that rate-distortion optimization, following certain modeling choices, is mathematically equivalent to the optimization performed in variational autoencoders [22], which conduct variational Bayesian inference. Therefore, even though the coding approaches do not have the same architecture as the work that proposed neural networks for Bayesian inference, mathematically, this is accomplished by the neural network. Consequently, by adopting variational autoencoders, the work in question [21] introduced a different line of research from what was previously presented [15].

Approaches based on variational autoencoders mainly represent results surpassing BPG and other more recent classic standards. Two enhancements proposed for the original work [21] are noteworthy. The first adopts a more robust entropy modeling by introducing a variational autoencoder that represents the hyperprior over the code space, improving the previously obtained results [23]. The second improves the architecture with the hyperprior by introducing context analysis of the latent space with autoregressive neural modeling [24].

The works based on variational inference presented so far have been enhanced in several ways. Some approaches bring improvements to the entropy model, optimizations of computational performance, models that produce multiple rates, the adoption of more robust neural operations for performance improvement, intelligent bit allocation and the use of attention modules, post-processing modules, and generative compression for encoding at very low rates [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44].

Advantages of these approaches include more flexible latent space modeling, the ability to compress the entire image, and better rate-distortion performance. Disadvantages include the loss of progressive encoding, beneficial in various scenarios, and the lack of control over the rate produced by each model.

1.1 Justification and Motivation

Despite progress in neural image compression, these methods have not been widely adopted in practical applications. Challenges include distrust in the training methods and metrics, the time needed to establish and incorporate new standards, and the computational de-

mands of these approaches. Additionally, rate control for reconstructing arbitrary images is a significantly underexplored problem in the literature.

Approaches based on Toderici's work [16] allow obtaining target rates, but achieving the desired rate requires encoding in multiple steps, making it complex for practical use. Ballé's approaches generally require training multiple networks to obtain rate-distortion operating points [45, 23]. Each trained model has considerable variance in rate and distortion for different images, and even multi-rate approaches struggle to achieve consistent target rates.

Minor advancements have been made in target-constrained training, such as Rozendaal's work, which derived a loss function for a target quality [46]. They use a search heuristic to fine-tune the hyperparameters of the rate-distortion loss. Recently, Zhang introduced a Rate Controllable Variational Autoencoder (RC-VAE) for image compression, which adapts to variable target rates [47]. This approach requires a complex, multi-stage training procedure, and the difficulty of the task is evident with the rapid saturation of quality reconstruction results at low bitrates. These challenges highlight the complexity of optimizing target rate restrictions.

1.2 General Objective and Specific Objectives

The overall objective of this document is to present end-to-end neural image compression architectures that can compress any image at a specific bitrate. The specific objectives are:

- Review concepts related to Information Theory, Bayesian Inference, Image Compression fundamentals, and Autoencoders, which are naturally suited for compression purposes;
- Conduct a literature review to find approaches related to optimizing under rate restrictions;
- Formulate training strategies and modify architectures to encode images under rate constraints;
- Implement and train the architectures, generate results, and validate/refine hypotheses;
- Perform empirical and theoretical analyses of the models and results to refine the proposed approaches;
- Publish results in reputable conferences and journals.

This document presents the results of some cumulative strategies for obtaining rate-oriented neural networks, whose foundation lies in modifying the variational loss function inspired by Lagrangian relaxation. Later, ideas of Reinforcement Learning are applied to vastly generalize the robustness of the initial target-rate loss. The results, presented in Chapter 5 and Chapter 6, demonstrate that it is possible to obtain models that encode at specific rates with a particular variance without significant deterioration of results compared to reference models without rate constraint.

1.3 Contributions

This document contributes to neural image compression, particularly in bitrate compression control. The first contribution is the idea of modifying a variational rate-distortion loss into a constrained target bitrate loss using Lagrangian relaxation. This method shows that even though these losses are not directly derived from variational Bayesian inference, their hyperparameters retain some features of the variational approach. These analyses and results were published in **Signal Processing: Image Communication** as **"Rate-constrained learning-based image compression"** [48], detailed in Chapter 4 and Chapter 5. Additionally, the analysis of the quantization mismatch problem, termed the mean rate shift problem, caused by using additive uniform noise as a proxy for non-differential quantization, is another contribution. A search heuristic was proposed to address this issue.

The second set of contributions improves the heuristic to overcome the mean rate shift problem. A solution to balance the mismatch caused by additive uniform noise is proposed using reinforcement learning ideas. The training process is seen as a time-evolution stochastic process, allowing temporal information to fine-tune neural network training. This idea is detailed in Chapter 6 and published in **IEEE Signal Processing Letters** as **"Learning-Based Image Compression with Parameter-Adaptive Rate-Constrained Loss"** [49]. Additionally, this thesis presents a range of analyses, potential extensions, and deeper investigations, highlighting future steps and contributing to the field.

1.4 Document Structure

This document is organized as follows:

- Chapter 2 covers fundamental concepts central to this work, focusing on image compression, neural networks and Bayesian inference, with a particular emphasis on variational autoencoders;

- Chapter 3 presents a literature review discussing recurrent network-based approaches, variational approaches, and works related to the main improvement directions;
- Chapter 4 details seminal architectures, which will be deeply studied in subsequent chapters. It also clearly states the main global problem this thesis addresses;
- Chapter 5 introduces a proposal for acquiring networks with rate constraints. It describes and analyzes the proposed modifications and presents results, fostering improvements in the approaches;
- Chapter 6 introduces an improved approach for rate-constrained coding in neural networks, building on earlier work. It presents a generalization of the previous proposal, now using reinforcement learning ideas to make the model auto-correct mismatches;
- Chapter 7 aggregates conclusions from the above chapters into a general perspective, pointing to future directions.

Chapter 2

Theoretical Framework

This chapter delves into the fundamental concepts central to the proposals in this work, focusing on two technical domains: image compression and neural networks. Sections 2.1 and 2.2 explore these domains, respectively. These sections build a robust theoretical framework to support the arguments and findings in later chapters. They also aim to equip the reader with the knowledge to appreciate the nuanced approaches and methodologies proposed in this thesis.

Bayesian Inference, presented in Section 2.3, covers crucial topics for the rest of this thesis, including variational Bayesian inference and variational autoencoders as a neural model for variational inference. Understanding these concepts is essential to comprehend the approaches discussed in this thesis.

The reader is assumed to be familiar with information theory concepts such as entropy, mutual information, and Kullback-Leibler divergence, detailed in Appendix I. If the reader is not well-versed in these topics, it is advisable to review this appendix before proceeding.

2.1 Image Coding

The core of this thesis explores image coding, focusing on its application within neural networks for image compression. Since pixels in images and videos are highly correlated, achieving efficient compression using only entropy CODECs is challenging [50]. Due to the complexity of pixels, which are typically adjacent and have similar values, entropy encoders require some degree of independence among data to function optimally. This correlation leads to extensive spatial redundancy, which can be exploited by models [50].

Models often leverage subjective redundancy by exploiting the human visual system's sensitivity to various image and video characteristics. To address these redundancies, many techniques have emerged, with Differential Pulse Code Modulation (DPCM) and transform coding being prominent strategies, which will be discussed in detail.

Given this general idea, the image coding domain has developed numerous compression strategies. This section highlights some of the critical concepts and techniques used. Figure 2.1 shows the overall scheme of techniques in image coding. “Preprocessing” refers to any procedure applied before image transformation. The “Transform” step projects the original data into another domain, allowing for the selection of valuable properties. After transformation, a quantization technique is applied to the coefficients, which are then encoded using entropy coding. Inverse processes are used to retrieve the data from the compressed format.

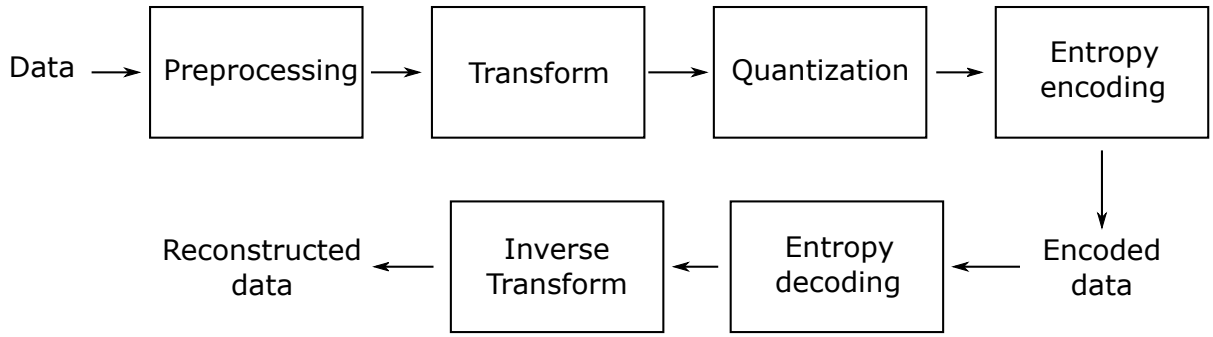


Figure 2.1: A prevalent scheme in image coding. While specific CODECs may incorporate additional components, this flowchart elucidates the general concept of the methods.

Based on this scheme, the following sections will explain common image processing techniques in detail. These include Differential Pulse Code Modulation (Section 2.1.1), Transform Coding (Section 2.1.2), and Entropy Coding with Arithmetic Coding (Section 2.1.3). These techniques are widely used in "Hybrid DPCM/DCT Mode CODECs," whose historical trajectory will be explained later in this section (Section 2.1.4).

The concepts explored in this section are deeply connected to Information Theory, a branch of applied mathematics and electrical engineering that quantifies information. It is strongly recommended to read Appendix I first, as it provides a solid grounding in the key concepts and principles of Information Theory. This will enhance your understanding of the technical depth and complexity of the subsequent discussions.

2.1.1 Differential Pulse-Code Modulation (DPCM)

Differential Pulse-Code Modulation (DPCM) is based on a predictive framework where each sample is predicted based on its predecessors, reducing redundancy and aiding in compression. This involves predicting a pixel value from previous pixel values and transmitting the difference (prediction error) instead of the exact pixel value. The spatial correlation in image data typically results in prediction errors that can be more efficiently encoded, achieving compression [50].

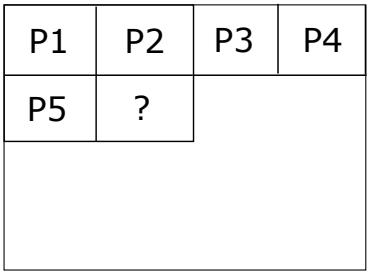


Figure 2.2: Schematic illustration of DPCM strategy. Given transmitted values $P1, P2, P3, P4$, and $P5$, subsequent values (represented by the question mark) can be predicted using $P5$ or a composite of preceding values. Adapted from [50]

Quantization enhances compression but introduces lossiness because it is irreversible. It maps a range of input values to a single quantized value, reducing the number of bits required for representation at the cost of fidelity. Consider a simple example to explain the DPCM methodology. Suppose the original pixel values in grayscale range from 0 to 255 as follows:

$$V = [120, 123, 125, 128, 130, 133, 135]$$

In DPCM, the initial pixel value (120) is transmitted as it is. Subsequent pixel values are predicted based on their predecessors. Using a simple prediction strategy where the next pixel value is predicted to be the same as the current one ($\hat{V_{i+1}} = V_i$), the prediction error ΔV_i is:

$$\Delta V_i = V_{i+1} - \hat{V}_{i+1} \tag{2.1}$$

By calculating and transmitting these difference values (ΔV), a series that potentially requires fewer bits for encoding can be obtained, thus compressing the data:

$$\Delta V = [120, 3, 2, 3, 2, 3, 2]$$

Further compression can be achieved by quantizing these difference values, mapping ranges of differences to single values, though this comes at the expense of perfect reconstruction of the original data. Understanding these concepts is fundamental for grasping how modern CODECs and neural network-based techniques refine and build upon these strategies to deliver enhanced fidelity and compression, as discussed in the following sections.

2.1.2 Transform Coding

Transform coding, a fundamental methodology in image compression, converts image samples into another domain, producing transform coefficients. The main objective is

to minimize the correlation between coefficients, isolating a few with significant visual importance. After transformation, a quantization process targets the less significant coefficients for compression. Combined with entropy coding, transform coding is a backbone for many image and video coding methods [50].

Consider an example to illustrate the mechanics of transform coding. Assume a 2×2 block of pixel values from a grayscale image:

$$I_F = \begin{bmatrix} 100 & 102 \\ 98 & 95 \end{bmatrix}$$

The transformation phase aims to represent these pixel values in a new domain, often the frequency domain, for more efficient compression. A common transform in image compression is the Discrete Cosine Transform (DCT). For simplicity, consider using a hypothetical transform matrix T_M for a 2×2 block:

$$T_M = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix}$$

We obtain the transform coefficients C_T by multiplying the transform matrix T_M with the image block I_F:

$$C_T = T_M \times I_F \times {T_M}^T = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix} \times \begin{bmatrix} 100 & 102 \\ 98 & 95 \end{bmatrix} \times \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

The coefficients matrix C_T represents the transformed domain of the original image block I_F. After transformation, most of the image's energy is typically compacted into a few coefficients, making the others prime candidates for quantization and compression without significantly compromising the perceptual quality of the reconstructed image. This example captures the essence of transform coding: moving to an alternative domain to simplify later compression stages by reducing inter-coefficient correlations and focusing energy on fewer coefficients. In real-world applications, the careful choice of transform types and tailored quantization strategies address a wide range of use cases and content types, which will be further detailed in upcoming sections.

2.1.3 Arithmetic Coder

General-purpose CODECs are designed to encode and compress data, especially data with statistical redundancy. This concept means a signal rich in information can be compressed. Compression is related to entropy, a measure of randomness or disorder in information. This section explores the entropy CODEC, focusing on the arithmetic coder, a widely

adopted algorithm for representing a signal according to its entropy [50]. Arithmetic coding maps a sequence of symbols to fractional numbers, which are then converted into binary for transmission. It offers superior compression compared to Huffman coding by representing each symbol with a fractional number of bits, allowing for a more statistically efficient allocation of data [50].

To differentiate symbol sequences effectively, a unique identifier is essential. These identifiers come from real numbers within the unit interval $[0, 1)$. Creating the identifier involves narrowing the interval as more sequence elements appear [51]. Given an alphabet $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$ and a random variable X such that $X(a_i) = i$ and $a_i \in \mathcal{A}$, with a probability model $p_{\mathcal{A}}$ for the data source, the probability density function for X is $p_X(i) = p_{\mathcal{A}}(a_i)$. The cumulative distribution function F_X is defined as [51]:

$$F_X(i) = \sum_{k=1}^{i} p_X(k) \tag{2.2}$$

The arithmetic coder starts by dividing the unit interval $[0, 1)$ into sub-intervals, each corresponding to a symbol in $\mathcal{A}$. For example, the symbol a_i is mapped to the interval $[F_X(i-1), F_X(i)]$. Assume a sequence of symbols $\{a_k, a_j, \ldots\}$ is to be encoded. Encoding the first symbol, a_k, refines the coding interval to $[low_{a_k}, high_{a_k}] = [F_X(k-1), F_X(k)]$. The arithmetic coder processes one symbol at a time, iteratively refining the interval. After encoding a_k, suppose the next symbol to encode is a_j. The interval for a_j following a_k is derived from the previous interval, becoming [51]:

$$[low_{a_j}, high_{a_j}] = [low_{a_k} + F_X(j-1) \cdot (high_{a_k} - low_{a_k}), \tag{2.3}$$

$$low_{a_k} + F_X(j) \cdot (high_{a_k} - low_{a_k})] \tag{2.4}$$

The encoding continues, progressively refining the interval for each subsequent symbol, $a_l, a_m, \ldots$, in the sequence.

Pseudo-Code for Arithmetic Encoding

The arithmetic encoding procedure compresses a sequence of symbols $\{a_k, a_j, a_l, a_m, \ldots\}$ by iteratively narrowing a numerical interval based on the cumulative probability function F_X. The pseudo-code below explains this step-by-step method.
where

- *seq*: A sequence of symbols to be encoded.
- F_X: The cumulative probability function for each symbol.
- *low* and *high*: Variables tracking the lower and upper bounds of the current interval.

Algorithm 1: Arithmetic Encoding

Result: Return the interval $[low, high)$
Data: *seq*: a sequence of symbols, F_X: cumulative probability function
Result: $[low, high)$: the final interval that represents the encoded sequence

$low \leftarrow 0$
$high \leftarrow 1$
for $i \leftarrow 1$ **to** $length(seq)$ **do**
    $range \leftarrow high - low$
    $high \leftarrow low + range \cdot F_X(i)$
    $low \leftarrow low + range \cdot F_X(i-1)$
return $[low, high)$

Illustrative Example of Encoding a Sequence

Consider an example with an alphabet $\mathcal{A} = \{a, b, c\}$ assigned the probabilities $p(a) = 0.7, p(b) = 0.2, p(c) = 0.1$. The corresponding cumulative function is:

$$F_X(0) = 0, \quad F_X(1) = 0.7, \quad F_X(2) = 0.9, \quad F_X(3) = 1 \tag{2.5}$$

Suppose the sequence to be encoded is $\{a, b, c\}$, with respective symbol indexes $\{1, 2, 3\}$. The procedure is as follows:

- Initially: $[low, high) = [0, 1)$
- After encoding a: $[low_1, high_1) = [F_X(0), F_X(1)) = [0, 0.7)$
- After encoding b, calculate:

$$low_2 = low_1 + F_X(1) \cdot (high_1 - low_1) = 0.49,$$
$$high_2 = low_1 + F_X(2) \cdot (high_1 - low_1) = 0.63,$$

  resulting in $[low, high) = [0.49, 0.63)$.

- After encoding c, refine the interval to:

$$low_3 = low_2 + F_X(2) \cdot (high_2 - low_2) = 0.616,$$
$$high_3 = low_2 + F_X(3) \cdot (high_2 - low_2) = 0.63,$$

  resulting in $[low, high) = [0.616, 0.63)$.

Any number within the interval $[0.616, 0.63)$ can accurately represent the encoded sequence $\{1, 2, 3\}$.

Decoding Process

In arithmetic coding, decoding is the inverse operation that transforms encoded numerical data back into its original symbolic form. Given an encoded number within a specific numerical range and the cumulative probability function F_X, the decoder sequentially reveals the original symbols by navigating nested numerical intervals.

Initially, the decoder identifies which interval $[F_X(j-1), F_X(j)]$ contains the encoded number, outputs the corresponding symbol a_j, and updates the encoded number by normalizing it to the relevant interval. This process continues until all symbols in the sequence are decoded.

Algorithm 2: Arithmetic Decoding

Data: *code*: the encoded number, F_X: cumulative probability function, n: length of the original sequence
Result: The original sequence of symbols

```
for i ← 1 to n do
    Find the smallest j such that code < F_X(j)
    Output symbol a_j as the next symbol in the decoded sequence
    code ← (code − F_X(j − 1))/(F_X(j) − F_X(j − 1))
```

Illustrative Example of Decoding a Sequence

Consider the earlier encoding example with the sequence $\{a, b, c\}$ and the cumulative function:

$$F_X(0) = 0, \quad F_X(1) = 0.7, \quad F_X(2) = 0.9, \quad F_X(3) = 1 \tag{2.6}$$

Assume the number 0.62 (which resides within the final encoding interval $[0.616, 0.63)$) is chosen as the encoded number. The decoding process progresses as follows:

1. Find that $j = 1$ is the smallest index such that $F_X(1) > 0.62$. Hence, the output $a_1 = a$, and update *code* can be given as follows:

$$\text{code} = \frac{0.62 - F_X(0)}{F_X(1) - F_X(0)} = \frac{0.62 - 0}{0.7 - 0} = 0.8857$$

2. With the updated code, $j = 2$ satisfies $F_X(2) > 0.8857$. So, $a_2 = b$ and *code* will be updated again:

$$\text{code} = \frac{0.8857 - F_X(1)}{F_X(2) - F_X(1)} = \frac{0.8857 - 0.7}{0.9 - 0.7} = 0.9285$$

3. Lastly, $j = 3$ satisfies $F_X(3) > 0.9285$. So, $a_3 = c$ is set and *code* is updated once more:

$$\text{code} = \frac{0.9285 - F_X(2)}{F_X(3) - F_X(2)} = \frac{0.9285 - 0.9}{1 - 0.9} = 0.285$$

Hence, the sequence is decoded as $\{a, b, c\}$ from the number 0.62, demonstrating the arithmetic decoding process.

Essentially, the arithmetic coder uses symbol probabilities to create partitions and iteratively forms sub-partitions based on these. This mechanism can approximate entropy and the theoretical compression limit with enhanced precision. However, it is crucial to note that longer symbol sequences require higher precision to accurately represent the fractional number [50].

2.1.4 Trajectory of Hybrid DPCM/DCT Mode CODECS

The evolution of image compression, crucial for advancements in multimedia and telecommunication applications like digital transmission CODECs and teleconferencing, has progressed through various phases, incorporating multiple techniques to reduce image data efficiently [1]. These techniques, including transforms, predictions, and scalar quantization, have been integrated into various image compression standards over time.

Early image compression methods used entropy coding to reduce statistical redundancy in image data, utilizing Huffman coding [2], Golomb codes [3], and arithmetic coding [4]. Transform coding then emerged, employing Fourier and Hadamard transforms before the introduction of the DCT (discrete cosine transform) [6]. The DCT became prominent for efficiently compacting image energy, surpassing earlier transforms [7].

JPEG-1, a DCT-based CODEC, is a significant example of integrating prediction and quantization techniques to reduce spatial redundancy in images [8]. It combines prediction and transform techniques by dividing images into blocks, applying the DCT to each, and then using DPCM on the DC (direct current) components of the transform. Compression is applied to the residuals using a specialized quantization table to discard less important details while preserving the principal image components [7].

JPEG 2000 [9], a successor using a transform based on 2D wavelets, found applications in fields like medical imaging and cinema. It uses an effective arithmetic coding method, EBCOT [10], to reduce statistical redundancy in wavelet coefficients [7]. Subsequent coding standards, such as BPG [11] and JPEG XR [12], aimed to improve compression rates compared to JPEG-1, with an increase in algorithmic complexity. However, these CODECs faced performance limitations with HDR (high dynamic range) images, leading to the introduction of JPEG XT [13], which supports HDR image compression while maintaining compatibility with earlier standards.

The introduction of JPEG XL [14] by the JPEG committee represents the ongoing quest for next-generation image compression systems. This standard aims to deliver robust rate-distortion performance for HD (high definition) and UHD (ultra high definition) images while integrating features for web distribution across various applications.

In this historical context, contemporary classic CODECs continue to evolve, striving for improved compression. Given the success of neural networks in image processing, these techniques have gained significant research interest as promising approaches to enhance compression outcomes. The following chapters will review recent studies using neural networks as holistic solutions for image coding. It is important to note that the class of CODECs described in this Section will be referred to as "classical CODECs" to differentiate them from end-to-end neural-based CODECs.

2.2 Machine Learning and Neural Networks

Machine learning, a subset of artificial intelligence, uses algorithms to enhance computational abilities by learning from data, often referred to as "experience" [52]. This paradigm is closely tied to statistical analysis, focusing on identifying patterns and making data-driven decisions. In image compression, machine learning is valuable for its ability to distill, assimilate, and encode significant features of image data, streamlining storage and transmission.

2.2.1 Algorithms and Tasks in Machine Learning

Machine learning comprises several core tasks, each offering a unique approach to data analysis and manipulation:

- **Classification:** Assigning elements to predefined groups.
- **Regression:** Estimating a continuous value from input features.
- **Ranking:** Ordering elements based on certain criteria.
- **Clustering:** Grouping data based on similar characteristics.
- **Dimensionality Reduction:** Simplifying data's feature space while retaining key information.

In image compression, *dimensionality reduction* is crucial as it aims to maintain significant image attributes while representing the data in a condensed form. For example, if $I \in \mathbb{R}^{m \times n}$ represents an image, the goal of dimensionality reduction is to find a function $f : \mathbb{R}^{m \times n} \to \mathbb{R}^{p \times q}$, where $p < m$ and $q < n$, that minimizes information loss while reducing data size.

Among various machine learning algorithms, such as decision trees, support vector machines, and k-nearest neighbors, artificial neural networks (ANNs) stand out as a critical tool, particularly in applications that require deriving intricate patterns from data, such as image compression. ANNs excel in learning and representing non-linear mappings, which is essential for handling the complexity and high dimensionality of image data.

Unlike other machine learning algorithms that struggle with the intricate, multidimensional structures of image data, ANNs leverage the correlations within this data. Through training that involves forward propagation of input data and backpropagation of errors, ANNs iteratively refine their internal weights to optimize a predefined objective function. This optimization minimizes the loss between the network's output and the actual data, ensuring that the compressed representations retain maximal information relevant to the original data.

2.2.2 Artificial Neural Networks and Multi-layer Percepton Networks

Artificial Neural Networks (ANNs) significantly advance image compression due to their ability to learn intricate patterns and data representations. Originating from the concept of the artificial neuron in the 1950s and 1960s, ANNs have evolved significantly, building on the foundational logic of perceptrons but integrating more advanced neuron models. Figure 2.3 illustrates the basic structure of an artificial neuron.

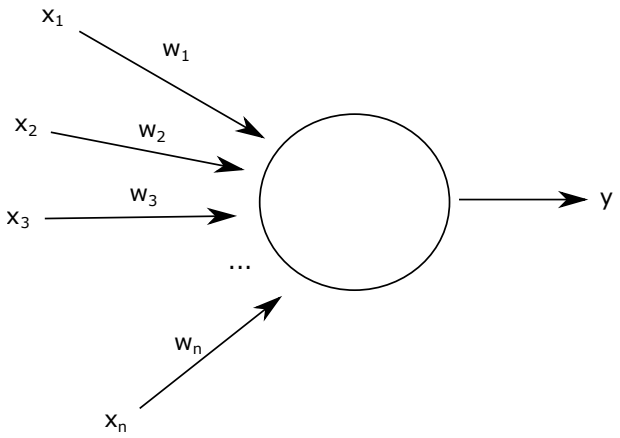


Figure 2.3: An artificial neuron comprising inputs $\boldsymbol{x} = \{x_1, x_2, ..., x_n\}$ and corresponding weights $\boldsymbol{w} = \{w_1, w_2, ..., w_n\}$, producing output y. Adapted from [53].

A perceptron processes multiple binary inputs $\boldsymbol{x} = \{x_1, x_2, ..., x_n\}$ and produces a single binary output based on a simple rule. The weights, represented as $\boldsymbol{w} = \{w_1, w_2, ..., w_n\}$, are real numbers indicating the significance of the corresponding input to the output. A

threshold determines the output as follows [53]:

$$y = \begin{cases} 0 \text{ if } \sum_j w_j x_j \leq \text{threshold} \\ 1 \text{ if } \sum_j w_j x_j > \text{threshold} \end{cases} \tag{2.7}$$

The threshold emulates neuronal firing characteristics, allowing for diverse decision models by adjusting the weights and threshold value. Although a perceptron simplifies the decision-making model, a network of perceptrons can theoretically formulate nuanced decisions [53]. Figure 2.4 shows an ensemble of such connected neurons.

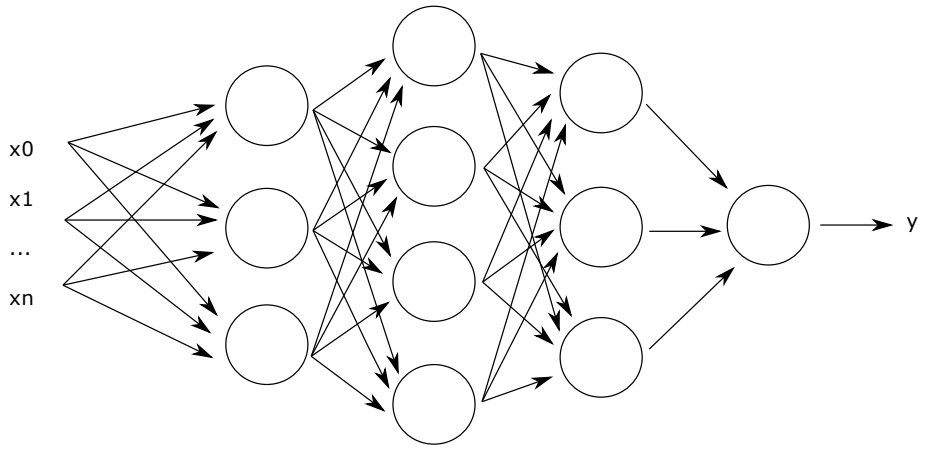


Figure 2.4: Neural network demonstrating collective decision-making. Each neuron layer amalgamates inputs, generating outputs utilized by subsequent layers, thereby facilitating complex decision derivations. Adapted from [53].

In a multilayer perceptron (MLP), the initial layers make simple decisions, and subsequent layers produce increasingly complex determinations based on the outputs of previous layers [53]. ANNs improve their efficiency through collaborative neuron interaction, where subtle weight changes can modify outputs. However, perceptrons have limitations: minor parameter changes can lead to drastic output shifts during threshold transitions [53]. Therefore, sigmoid neurons use the sigmoid function [53]. While retaining the perceptron structure, they produce a smoother output across the $[0, 1]$ interval:

$$\eta(\boldsymbol{w}\boldsymbol{x} + b) = \frac{1}{1 + e^{-\left(\sum_j w_j x_j\right) - b}} \tag{2.8}$$

The continuous transition facilitated by the sigmoid function ensures that small parameter adjustments lead to incremental output changes [53]:

$$\Delta y \approx \sum_j \frac{\partial y}{\partial w_j} \Delta w_j + \frac{\partial y}{\partial b} \Delta b \tag{2.9}$$

While individual neurons can make decisions, their limited flexibility for intricate cases underscores the necessity of combining neurons into ANNs, comprising multiple layers. The depth and structure of the network critically impact learning efficiency, predictive accuracy, and problem-solving capabilities, warranting an exploration of diverse neural network architectures and their merits [54].

2.2.3 Feedforward Neural Networks

Figure 2.4 shows that feedforward neural networks are foundational in deep learning and image compression models. Herein, *feedforward* implies the unidirectional flow of information from the input $\boldsymbol{x}$, through intermediate layers determining network behavior, to the output $\boldsymbol{y}$ [55].

Formally, a feedforward network aims to approximate a function f^*. In a classification context, $y = f^*(\boldsymbol{x})$ maps an input $\boldsymbol{x}$ to a category y. This mapping, parameterized by θ, represents the set of weights and aims to learn θ such that $y = f(\boldsymbol{x}, \boldsymbol{\theta})$ is a competent approximation [55].

The network layers comprise the input layer (the first layer), hidden layers (intermediate layers), and the output layer (the final layer). The terms "network depth" and "width" refer to the number of layers and the elements within a layer, respectively. Notably, the activation function is employed post-input combination at each neuron [55].

2.2.4 Backpropagation in Neural Network Training

Training a feedforward neural network involves two critical stages: forward propagation and backpropagation, which work together to optimize the network's parameters, $\boldsymbol{\theta}$. Initially, an input $\boldsymbol{x}$ generates an output $\hat{\boldsymbol{y}}$ through *forward propagation*, and the scalar cost $J(\boldsymbol{\theta})$ is calculated by comparing $\hat{\boldsymbol{y}}$ to the true output. Conversely, *backpropagation*, introduced by Rumelhart et al. [56], works backward from the cost to calculate gradients essential for parameter optimization. These gradients are used with algorithms like stochastic gradient descent to update the parameters. While backpropagation is commonly used to compute $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$, it can also be applied to other derivatives, highlighting its versatility in various computational tasks [55].

Computational graphs help clarify the backpropagation process, with nodes representing variables and directed edges representing operations. Variables can take various forms, from scalars to tensors, and operations may range from simple to complex functions on these variables. This formalized computational visualization, as shown in Figure 2.4, is invaluable for understanding calculations in neural networks, especially when dealing with several composed operations or variables with multiple entries [55].

The chain rule of calculus is fundamental for computing derivatives of composite functions and constructing the backpropagation framework. Let $\boldsymbol{x} \in \mathbb{R}^m$ and $\boldsymbol{y} \in \mathbb{R}^n$, and define two functions, $g : \mathbb{R}^m \to \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$. If $\boldsymbol{y} = g(\boldsymbol{x})$ and $z = f(\boldsymbol{y})$, then [55]:

$$\frac{dz}{d\boldsymbol{x}} = \frac{dz}{d\boldsymbol{y}}\frac{d\boldsymbol{y}}{d\boldsymbol{x}} \quad \text{and} \quad \nabla_{\boldsymbol{x}} z = \left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right)^{\top} \nabla_{\boldsymbol{y}} z \tag{2.10}$$

where $\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}$ is the Jacobian matrix of g. Thus, backpropagation fundamentally involves performing a Jacobian-gradient product for each operation within the computational graph, validating its efficiency across scenarios involving vectors and tensors of higher dimensions.

Within ANNs, where $\boldsymbol{\theta}$ represents the weights of the neurons (or, as will be elaborated in convolutional neural networks, the elements of the convoluting kernel), optimizing these parameters (weights) is crucial. During training, the network adjusts $\boldsymbol{\theta}$ to minimize the cost function $J(\boldsymbol{\theta})$, which quantifies the deviation between the network's prediction and the actual data. The derivative of the cost function with respect to $\boldsymbol{\theta}$ can be expressed as:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial J}{\partial \hat{\boldsymbol{y}}}\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{\theta}}, \tag{2.11}$$

where $\frac{\partial J}{\partial \hat{\boldsymbol{y}}}$ represents the derivative of the cost function with respect to the network's output, and $\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{\theta}}$ denotes the derivative of the output with respect to the parameters. Although the actual computations are more intricate when expanding the derivatives, especially in a network with numerous layers, the overarching principle remains consistent: backpropagation propagates the error backward through the network, calculating gradients that adjust the weights and biases to minimize the error function J.

Extending this logic, when the flow involves multiple layers or functions, the derivative of the output with respect to a parameter becomes a product of derivatives across the sequence of functions (or layers) from the output back to the parameter in question. This principle underscores the essence of backpropagation, where derivatives are calculated starting from the output and moving through each layer, exploiting the chain rule to amalgamate local derivatives into global derivatives, thereby facilitating the optimization of parameters even in deep structures. Each neuron's contribution to the error is computed using these gradients, guiding the parameter updates and refining the network's predictions through learning.

2.2.5 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) specialize in processing data with a grid-like topology. They are particularly suited for tasks like image compression and other applications involving structured data, such as time series or pixels [55].

The term "convolutional" originates from the mathematical operation "convolution", a specialized linear operation. A CNN uses convolution instead of matrix multiplication in at least one of its layers, achieving notable success in real-world applications [55]. A simple convolutional neural network, composed of two convolutional and two perceptron layers, is depicted in Figure 2.5. The input layer intakes $2D$ grid-type data, x, which undergoes processing via convolutions, yielding feature maps. Dashed squares indicate the mapping executed by convolutional kernels. In this instance, the convolution operation is paired with a subsampling operation, reducing the feature map size. The final two layers, perceptron layers, transform the $2D$ feature maps into a set of activations, culminating in a single value output, y.

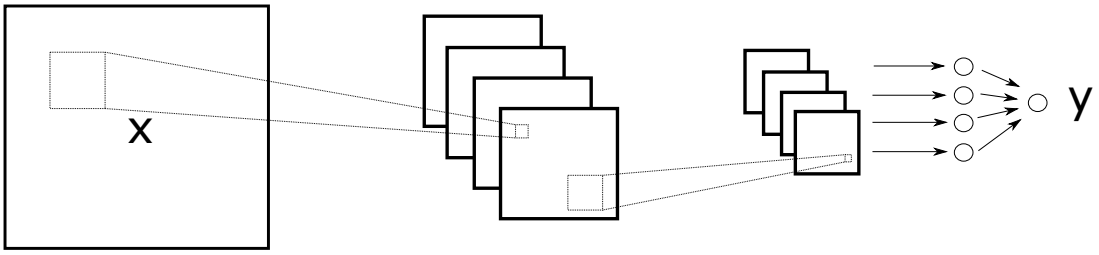


Figure 2.5: An illustrative example of a convolutional neural network.

Let x be an input function and w a weighting function. The convolution operation is defined as:

$$h(t) = (f * g)(t) = \int f(x)g(t-x)dx \tag{2.12}$$

In CNN terminology, f denotes the input, and g the *kernel*, with the output termed the feature map. In neural network contexts, considering the elements are discrete, the convolution translates into its discrete form [55]:

$$h(t) = (f * g)(t) = \sum_{x=-\infty}^{\infty} f(x)g(t-x) \tag{2.13}$$

Although the convolution is expressed in a $1D$ context above, it can be generalized to $2D$ for image processing applications. In a 2D convolution, the kernel is also a 2D function. Thus, the operation can be mathematically formulated as:

$$h(i,j) = (f * g)(i,j) = \sum_{m}\sum_{n} f(m,n)g(i-m, j-n) \tag{2.14}$$

where $f(m,n)$ represents the image and $g(i-m, j-n)$ is the kernel. Both f and g are defined in two dimensions, and the sums are taken over all possible values of m and n. This operation is performed for every pixel location (i,j) in the image, typically resulting in an output image, h, of the same size as the input image, f. In convolutional neural networks, this 2D convolution is crucial for filtering and deriving meaningful features from the input image or feature maps by sliding the kernel across the input in both dimensions (width and height).

In the context of Figure 2.5, the 2D convolutions are visualized abstractly as the movement and application of kernels across the input grid data. While a single kernel is often visualized for simplicity, in practice, a convolutional layer employs numerous kernels simultaneously, each extracting different features, such as edges, textures, or more complex patterns. These resultant feature maps then pass through subsequent layers, enabling the network to progressively learn and make informed decisions based on the hierarchical feature representations.

In image compression, the convolutional layers are pivotal for extracting and learning pertinent features representing the underlying data. These extracted features compress the representation, maintaining essential information while reducing dimensionality, thereby enabling efficient storage and transmission.

2.2.6 Recurrent Neural Networks and their Temporal Dynamics

Recurrent Neural Networks (RNNs) excel at processing sequential data due to their ability to remember past inputs using shared parameters over time steps [55]. RNNs effectively manage dependencies between encoded frames for video compression with variational autoencoders, enhancing compression results.

Unlike traditional feedforward networks, RNNs maintain a memory of past inputs. This memory, stored in their internal state, changes through recurrent computations as depicted in [55]:

$$\boldsymbol{h}(t) = g^{(t)}(\boldsymbol{x}(t), \boldsymbol{x}(t-1), \ldots, \boldsymbol{x}(2), \boldsymbol{x}(1)) \tag{2.15}$$

$$= f(\boldsymbol{h}(t-1), \boldsymbol{x}(t); \boldsymbol{\theta}) \tag{2.16}$$

Here, $\boldsymbol{h}(t)$ denotes the state at time t. The function $g^{(t)}$ takes the entire past sequence as input and yields the current state. The recurrent structure is broken down by repeatedly applying the function f. The RNN dynamics are further elucidated by the following

update equations [55]:

$$\boldsymbol{a}(t) = \boldsymbol{b} + \boldsymbol{W} \cdot \boldsymbol{h}(t-1) + \boldsymbol{U} \cdot \boldsymbol{x}(t) \tag{2.17}$$

$$\boldsymbol{h}(t) = \tanh(\boldsymbol{a}(t)) \tag{2.18}$$

$$\boldsymbol{o}(t) = \boldsymbol{c} + \boldsymbol{V} \cdot \boldsymbol{h}(t) \tag{2.19}$$

$$\hat{\boldsymbol{y}}(t) = \text{softmax}(\boldsymbol{o}(t)) \tag{2.20}$$

RNNs are usually trained via the Backpropagation Through Time (BPTT) approach. The loss function, J, summarizes the losses across all sequence steps. As shown in [55]:

$$J\left(\{\boldsymbol{x}(1), \ldots, \boldsymbol{x}(\tau)\}, \{\boldsymbol{y}(1), \ldots, \boldsymbol{y}(\tau)\}\right) = \sum_{t=1}^{\tau} J^{(t)}, \tag{2.21}$$

where $J^{(t)}$ is the loss at each time step t.

Computing gradients in an RNN, essential for gradient-based optimization, uses the general backpropagation algorithm applied to the unrolled computational graph. To understand how BPTT calculates gradients, consider the nodes representing parameters and sequence-indexed nodes in the computational graph. To compute the gradient $\nabla_N J$ for each node N, it depends on the gradient at the subsequent nodes. This recursion starts with nodes just before the final loss [55]:

$$\frac{\partial J}{\partial J^{(t)}} = 1. \tag{2.22}$$

The gradient $\nabla_{\boldsymbol{o}(t)} J$ for the outputs at time t is:

$$(\nabla_{\boldsymbol{o}(t)} J)_i = \frac{\partial J}{\partial \boldsymbol{o}(t)_i} = \frac{\partial J}{\partial J^{(t)}} \frac{\partial J^{(t)}}{\partial \boldsymbol{o}(t)_i} \tag{2.23}$$

Specialized RNN variants, such as Long Short-Term Memory (LSTM) networks [57] and Gated Recurrent Units (GRUs) [58], address the challenges of traditional RNNs in capturing long-term dependencies. LSTMs have features to carry information across extensive sequences, while GRUs simplify LSTM structures by tweaking information flow.

Convolutional variants of LSTM and GRU, namely Convolutional LSTM (ConvLSTM) [59] and Convolutional GRU (ConvGRU) [60], harness the spatially-local connectivity of convolutional layers for grid data, like images or videos, while preserving their sequential dependency capabilities.

In learned video compression, ConvLSTMs and ConvGRUs process the temporal dependencies between video sequence frames. In a typical VAE, an encoder compresses the input data into a latent space, and a decoder reconstructs it. ConvLSTMs/ConvGRUs

ensure the compression model uses information from surrounding frames for the current frame's compressed representation. More details can be found in many works [59, 60].

2.2.7 Activation Functions and the GDN Function

The terminology related to *activation functions* in the context of artificial neural networks (ANNs) has historically been somewhat inconsistent and has evolved over time. Terms such as *transfer function* and *output function* have been used interchangeably with *activation function* in various contexts and literature. However, distinctions between these terms have been made in some contexts [61]:

- *Activation function* $I(\boldsymbol{x})$: An internal transformation of the input values $\boldsymbol{x}$. Commonly, $I(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}x + b$, where $\boldsymbol{w}$ and b are the neuron parameters (weights) and bias, respectively.
- *Output function* $o(a)$: A function which computes the output value of the neuron using the activation value $a = I(\boldsymbol{x})$, i.e., $o : a \in \mathbb{R} \to o(a) \in \mathbb{R}$.
- *Transfer function* $T(\boldsymbol{x})$: Defined as the composition of the activation and output function, $T(\boldsymbol{x}) = o(I(\boldsymbol{x}))$.

These definitions have not been consistently utilized across literature and practices. In more recent practices, the term *activation function* has also been applied to what was traditionally referred to as the *output function*, blurring the distinction between these terms. Regardless, activation functions are typically characterized as mappings between two subsets of the real numbers, adhering to specific properties to guarantee the universal approximator property of multilayer feedforward (MLFF) networks. These functions are usually non-constant, bounded, and monotonically increasing continuous functions [61].

Fixed-shape Activation Functions

Fixed-shape activation functions, devoid of adjustable parameters during training, have significantly influenced the evolution and understanding of trainable functions, inspiring the development of numerous derivatives and combinations. Despite historical limitations, such as the identity function's inability to approximate continuous functions in early Neural Network (NN) architectures, these functions have paved the way for more refined functions [61]. A detailed exposition of some of these pivotal functions, including the identity function, can be found in Table 2.1.

NN architectures have preferred bounded activation functions like the sigmoid and hyperbolic tangent due to their ability to approximate any continuous function defined on a compact subset. This is true provided the number of hidden neurons is sufficiently large,

and the functions are non-constant, bounded, and monotonically increasing continuous entities (see Table 2.1, Sigmoid and Hyperbolic tangent) [61].

Table 2.1: Some of the most used fixed activation functions [61].

Name	**Expression**	**Range**
Identity	$id(a) = a$	$(-\infty, +\infty)$
Step (Heaviside)	$\mathrm{Th}_{\geq 0}(a) = \begin{cases} 0 & \text{if } a < 0 \\ 1 & \text{otherwise} \end{cases}$	$\{0, 1\}$
Bipolar	$B(a) = \begin{cases} -1 & \text{if } a < 0 \\ +1 & \text{otherwise} \end{cases}$	$\{-1, 1\}$
Sigmoid	$\eta(a) = \frac{1}{1+e^{-a}}$	$(0, 1)$
Bipolar sigmoid	$\eta_B(a) = \frac{1-e^{-a}}{1+e^{-a}}$	$(-1, 1)$
Hyperbolic tangent	$\tanh(a)$	$(-1, 1)$
Hard hyperbolic tangent	$\tanh_H(a) = \max(-1, \min(1, a))$	$[-1, 1]$
Absolute value	$abs(a) = \lvert a \rvert$	$[0, +\infty)$
Cosine	$\cos(a)$	$[-1, 1]$
Softmax	$\mathrm{Softmax}(a)_i = \frac{e^{a_i}}{\sum_j e^{a_j}}$	$(0, 1)$
ReLU	$\mathrm{ReLU}(a) = \max(0, a)$	$[0, +\infty)$
Leaky ReLU	$\mathrm{LeakyReLU}(a) = \max(\alpha a, a)$	$(-\infty, +\infty)$
Softplus	$S_+(a) = \ln(1 + e^a)$	$(0, +\infty)$
ELU	$\mathrm{ELU}(a; \alpha) = \begin{cases} a & \text{if } a > 0 \\ \alpha(e^a - 1) & \text{otherwise} \end{cases}$	$(-\alpha, +\infty)$
SiLU (Swish)	$\mathrm{SiLU}(a) = a\eta(a)$	$(-\infty, +\infty)$

With the advent of the Rectified Linear Unit (ReLU) and its derivatives, a significant shift in NN architecture has been observed. ReLU's unbounded nature in the positive domain and its capacity to encourage sparse activations have notably reduced the propensity for the vanishing gradient problem, creating an environment where only a sparse set of neurons are activated in a given layer. ReLU also confers robustness against minor input perturbations and enhances representational capacity (see Table 2.1, ReLU) [61].

However, ReLU has challenges, such as the "dying" ReLU problem and non-differentiability at zero. These issues are especially pronounced when a neuron has a significant negative bias, causing it to consistently output zero regardless of input. Variations like the Leaky ReLU have been proposed to address these issues. Leaky ReLU introduces a small non-zero gradient when the input is negative, alleviating the mentioned problems (see Table 2.1, Leaky ReLU) [61].

Trainable Activation Functions

Research into trainable activation functions in neural networks has evolved since the early 1990s, witnessing substantial developments in parameterized standard and ensem-

ble methods-based activation functions. With the renewed interest in neural networks, researchers continue to explore whether trainable activation functions could significantly enhance neural network performance. Some examples of these activation functions, including the Sigmoidal Selector and Flexible ReLU, are detailed in Table 2.2 [61].

Table 2.2: Examples of trainable activation functions. Note that exp represents the exponential function e^x [61].

Name	**Expression**	**Trainable Parameters**	**Range**
AGSig	$\text{AGSig}(a;\alpha,\beta) = \frac{\alpha}{1+\exp(-\beta a)}$	α,β	$(0,\alpha)$
AGTanh	$\text{AGTanh}(a;\alpha,\beta) = \frac{\alpha(1-\exp(-\beta a))}{1+\exp(-\beta a)}$	α,β	$(-\alpha,\alpha)$
PReLU	$\text{PReLU}(a;\alpha) = \begin{cases} a & \text{if } a > 0 \\ \alpha a & \text{otherwise} \end{cases}$	α	$(-\infty,+\infty)$
PELU	$\text{PELU}(a;\beta,\gamma) = \begin{cases} \frac{\beta}{\gamma}a & \text{if } a \geq 0 \\ \beta(\exp(\frac{a}{\gamma}) - 1) & \text{otherwise} \end{cases}$	β,γ	$(-\infty,+\infty)$
Swish	$\text{Swish}(a;\alpha) = a \cdot \sigma(\alpha a)$	α	$(-\infty,+\infty)$
Sigmoidal Selector	$\text{S}_k(a;k) = \left(\frac{1}{1+\exp(-a)}\right)^k$	k	$(0,1)$
Flexible ReLU	$\text{frelu}(a;\beta) = \text{ReLU}(a) + \beta$	β	$(-\infty,+\infty)$

Trainable activation functions, particularly those derived from parameterized standard activation functions, usually maintain a shape similar to their non-trainable counterparts, achieving only modest enhancements in expressiveness. For example, AGSig and AGTanh largely retain the shapes of sigmoid and Tanh functions, respectively, with modified smoothness and amplitude modulated by parameters α and β (see Table 2.2). Similarly, Swish can be considered a parameterized variant of SiLU/ReLU, learning its final shape to balance between the two. However, its general shape remains closely related to the primary function from which it was developed [61].

The Sigmoidal Selector represents a class of sigmoidal functions parameterized by a value k in the interval $(0,+\infty)$, allowing for the selection of a practical function during the learning process via gradient descent and backpropagation algorithms. Moreover, the Flexible ReLU (frelu) introduced by Qiu, Xu, and Cai (2018) captures the negative information lost with the classic ReLU function and provides the zero-like property while having its parameters α and β learned by the data, offering a more adaptable activation function [61].

A substantial portion of these functions, essentially a weighted output from the respective weighted input of a fixed activation function, can be modeled via a shallow neural subnetwork composed of a few neurons. These trainable activation functions highlight the

potential versatility within neural network architectures, warranting further investigation into their capabilities and limitations [61].

Generalized Divisive Normalization

The Generalized Divisive Normalization (GDN) is a notable activation function, deserving detailed discussion due to its pivotal role in the baseline architectures explored in this thesis [21, 45, 23]. GDN was initially envisioned as a mechanism for optimizing transforms to acquire specific statistical properties in a transformed space, a concept fundamental to efficient sensory coding theories in neurobiology. It has also found utility in the design of cascaded operations, including deep neural networks [21].

The core of cascaded operations lies in seeking transformations that engender marginal "directions" with minimized similarity to Gaussians, applying Gaussianization to these directions through non-parametric non-linear scalar transforms [21]. While such transformations can converge to diverse data densities, their broad applicability can render models susceptible to errors, requiring substantial data and potentially slow, data-dependent convergence, primarily as they operate solely on marginals [21].

Divisive normalization operates as a form of gain control, dividing responses by a weighted activity of neighbors, and has been widely adopted to elucidate the non-linear properties of sensory neurons [21]. A typical representation of this transformation is:

$$y_i = \gamma \cdot \frac{x_i^\alpha}{\beta^\alpha + \sum_j x_j^\alpha} \tag{2.24}$$

with parameters $\boldsymbol{\theta} = \{\alpha, \beta, \gamma\}$. This transformation adapts responses to an intended operating range while preserving relative values. A variant with $\alpha = 2$ has demonstrated its capacity to yield roughly Gaussian responses with notably diminished dependencies [62], though its efficacy is predominantly observed when applied to groups of spatially local responses [21].

GDN, a generalization featuring enhanced Gaussianization capabilities, extends applicability to more distant responses and those obtained via different filters [21]. GDN is defined by a density transformation $\boldsymbol{y} = g(\boldsymbol{x}; \boldsymbol{\theta})$, wherein:

$$y_i = \frac{z_i}{(\beta_i + \sum_j \gamma_{ij} |z_j|^{\alpha_{ij}})^{\epsilon_i}} \tag{2.25}$$

$$\boldsymbol{z} = \boldsymbol{T} \cdot \boldsymbol{x} \tag{2.26}$$

with parameter vector $\boldsymbol{\theta}$ composed of vectors $\boldsymbol{\beta}$, $\boldsymbol{\epsilon}$, and matrices $\boldsymbol{T}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\gamma}$.

An efficient inversion of this transformation is possible using fixed-point iteration [21]:

$$z_i^{(0)} = \mathrm{sgn}(y_i)(\gamma_{ii}^{\epsilon_i}|y_j|)^{\frac{1}{1-\alpha_{ii}\epsilon_i}} \tag{2.27}$$

$$z_i^{(n+1)} = (\beta_i + \sum_j \gamma_{ij}|z_j^{(n)}|^{\alpha_{ij}})^{\epsilon_i} y_i \tag{2.28}$$

The originators illustrated that GDN adeptly "Gaussianizes" data via parameter optimization, achieved by minimizing the Kullback-Leibler divergence of the distribution of transformed data relative to a Gaussian [21]. The details of the use of GDN will be explored in subsequent chapters. For the definition of the Kullback-Leibler divergence, please refer to Appendix I. As a crucial component in comprehending the reference architectures, further insights into GDN can be garnered from the proposed article [21].

2.2.8 Reinforcement Learning

The emergence of AlphaGo, a revolutionary AI program developed by Google DeepMind, marked a watershed moment in artificial intelligence, defeating top human players in the ancient game of Go in 2016 and 2017. Powered by advancements in reinforcement learning (RL), AlphaGo showcased the potential of machine learning across diverse domains, including self-driving cars and data center optimization. Decision-making is crucial for resource management across hierarchical time scales in process operations. While mathematical programming (MP) techniques like model predictive control are prevalent, they encounter challenges in addressing large-scale stochastic problems [63].

Reinforcement learning (RL) offers a promising solution by deriving optimal decision policies through interaction with the environment and feedback mechanisms. RL enables data-driven dynamic programming and provides computationally feasible solutions using stochastic simulations. Despite its potential, challenges remain, such as the complexities of formulating Markov decision processes (MDPs) for multi-stage decision problems in process operations [63].

This section will briefly discuss the main idea of reinforcement learning. The intent is to provide the reader with a foundational understanding to extend the philosophy of the approaches in this thesis [48].

The general idea of Reinforcement Learning

Machine Learning is fundamentally divided into three classes of algorithms: Supervised Learning (SL), Unsupervised Learning (UL), and Reinforcement Learning (RL). In supervised learning, an agent learns to map from labeled examples to predict the values of new inputs. For instance, given pictures of animals with corresponding labels, the agent learns

to identify images containing specific types of animals. Unsupervised Learning involves unlabeled data and aims to understand the data distribution. For example, the agent may group pictures based on the kinds of animals they contain. The unsupervised paradigm is strongly related to the idea of autoencoders. In contrast, Reinforcement Learning focuses on learning the best way to accomplish tasks through interactions with the environment. In this paradigm, the agent evaluates the long-term value of its actions [63].

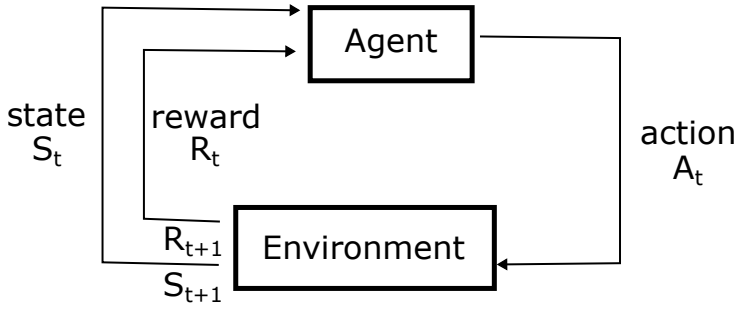


Figure 2.6: The basic idea of reinforcement learning. Adapted from [63]

The primary setting for RL, derived from Animal Psychology, is illustrated in Figure 2.6. It involves two entities: the Agent and the Environment. At each time step, the Agent takes an action, affecting the environment and transitioning it from state S_t to S_{t+1}. This action generates an immediate reward R_{t+1}. The Agent utilizes the state information S_{t+1} and immediate reward R_{t+1} to determine the following action A_{t+1}, perpetuating the cycle. The objective of the Agent is to learn a policy $\pi(A_t = a|S_t = s)$ that maximizes a long-term sum of future rewards, known as the value function $v_\pi(s)$:

$$v_\pi(s) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s]$$

where $\gamma \in [0, 1]$ is a discount factor that determines the importance of the future for the decisions. RL algorithms can be considered iterative updates that improve the policy so that the associated value function increases for all states [63].

2.3 Bayesian Inference

This section elucidates pivotal concepts intrinsic to Bayesian inference, establishing a theoretical foundation that will seamlessly integrate with the advanced coding proposals and methodologies explored in subsequent sections. The initial focus will be on unraveling the intricacies of Bayes' theorem, followed by a discussion on the inherent challenges posed by intractability in Bayesian inference, and subsequently exploring its variational formulation.

The final topic covers variational autoencoders. Understanding these models is crucial for comprehending the intricacies of this thesis. The choice to present variational autoencoders under Bayesian inference is to highlight these sophisticated neural architectures as models for applying Bayesian mathematics, which is the focus of this thesis. Therefore, this section is essential for understanding the core concepts of the proposals presented in this document.

2.3.1 Bayes' Theorem

Bayes' theorem offers a methodical approach for updating probabilities or beliefs in light of new data, especially within the realm of random variables. Let the random variables X and Y symbolize specific events within a probabilistic framework. When we state $X = \boldsymbol{x}$, it implies that the random variable X assumes the vector value $\boldsymbol{x}$ [64].

The random variable type - discrete or continuous - dictates the function employed to describe its probability distribution. If X is a continuous random variable, $p_X(\boldsymbol{x})$ denotes the probability density function (PDF) of X at $\boldsymbol{x}$. The PDF represents the likelihood of X falling within a certain range of values. Conversely, if X is a discrete random variable, its distribution is described using a probability mass function (PMF), similarly denoted by $p_X(\boldsymbol{x})$.

Both the PDF and PMF define the distribution of a random variable for continuous and discrete random variables, respectively. In Bayesian statistics, understanding how observing $X = \boldsymbol{x}$ alters the probability associated with Y is crucial. The conditional probability $p_{Y|X}(\boldsymbol{y}|\boldsymbol{x})$ represents the probability of $\boldsymbol{y}$ given that $X = \boldsymbol{x}$ and is defined as follows [64]:

$$p_{Y|X}(\boldsymbol{y}|\boldsymbol{x}) = \frac{p_{X,Y}(\boldsymbol{x}, \boldsymbol{y})}{p_X(\boldsymbol{x})} \tag{2.29}$$

The multiplication rule of probability breaks down Equation 2.29 as:

$$p_{X,Y}(\boldsymbol{x}, \boldsymbol{y}) = p_{X|Y}(\boldsymbol{x}|\boldsymbol{y})p_Y(\boldsymbol{y}) \tag{2.30}$$

Furthermore, the denominator in Equation 2.29, $p_X(\boldsymbol{x})$, can be found using the law of total probability. In a continuous scenario, it can be written as:

$$p_X(\boldsymbol{x}) = \int p_{X|Y}(\boldsymbol{x}|\boldsymbol{y})p_Y(\boldsymbol{y})d\boldsymbol{y} \tag{2.31}$$

Substituting Equations 2.30 and 2.31 into 2.29, we obtain [64]:

$$p_{Y|X}(\boldsymbol{y}|\boldsymbol{x}) = \frac{p_{X|Y}(\boldsymbol{x}|\boldsymbol{y})p_Y(\boldsymbol{y})}{\int p_{X|Y}(\boldsymbol{x}|\boldsymbol{y})p_Y(\boldsymbol{y})d\boldsymbol{y}} \tag{2.32}$$

With $p_{X|Y}(\boldsymbol{x}|\boldsymbol{y})$ representing the likelihood and the integral $\int p_{X|Y}(\boldsymbol{x}|\boldsymbol{y})p_Y(\boldsymbol{y})d\boldsymbol{y}$ or its discrete equivalent representing the marginal likelihood, these elements ensure consistency across statistical and probabilistic theories, enabling Bayes' theorem to systematically enable the updating of beliefs upon receiving new data, thereby supporting robust methodologies in statistical inference and probabilistic data analysis [64].

2.3.2 Inference through Bayes' Theorem

In Bayesian statistics, inference plays a critical role in updating the probabilities of hypotheses, denoted as Y, upon observing new data, symbolized as X. By anchoring our discussion in Bayes' Theorem, we explore the practice of inferential statistics where data X and hypothesis Y engage in a sophisticated interaction [65].

Bayes' theorem, adapted to our notation X and Y, is articulated as:

$$p_{Y|X}(\boldsymbol{y}|\boldsymbol{x}) = \frac{p_Y(\boldsymbol{y})p_{X|Y}(\boldsymbol{x}|\boldsymbol{y})}{p_X(\boldsymbol{x})} \tag{2.33}$$

Dissecting the components of the equation [65]:

- $p_{Y|X}(\boldsymbol{y}|\boldsymbol{x})$: The *posterior probability*, quantifying the belief in hypothesis Y after observing $X = \boldsymbol{x}$.
- $p_Y(\boldsymbol{y})$: The *prior probability*, depicting the initial belief in Y without the new data.
- $p_{X|Y}(\boldsymbol{x}|\boldsymbol{y})$: The *likelihood*, denoting the probability of observing $X = \boldsymbol{x}$ given $Y = \boldsymbol{y}$ is true.
- $p_X(\boldsymbol{x})$: The *marginal likelihood*, illustrating the probability of observing $X = \boldsymbol{x}$ without considering the validity of Y.

In the case of discrete random variables, the notation $X = \boldsymbol{x}$ denotes that the random variable X takes the specific value $\boldsymbol{x}$, using a probability mass function (PMF). For continuous random variables, wherein a probability density function (PDF) is utilized, $P(X = \boldsymbol{x}) = 0$ for any specific $\boldsymbol{x}$ due to the infinite possibilities X might assume. However, the notation $X = \boldsymbol{x}$ is still commonly employed for simplicity, even in the continuous case, understanding that it typically denotes an interval around $\boldsymbol{x}$.

Deepening our understanding of concepts vital to inference, wherein data (X) and hypothesis (Y) are entwined [66]:

- **Parameter**: A population is thought to be represented by a probability distribution parameterized by $\boldsymbol{\theta} = \{\theta_1, \theta_2, ..., \theta_k\}$.

- **Observation**: Prior to data acquisition, future observations are categorized as **random variables** $\boldsymbol{X} = \{X_1, X_2, ..., X_n\}$, complying with the population distributions. A specific realization, x_k, is referred to as the **realized value** of X_k.
- **Statistic**: Within an inferential context, **data statistics**, or data summarizations, are vital. A set of statistics is deemed sufficient if it encompasses all the information necessary for inference concerning $\boldsymbol{\theta}$.
- **Estimator**: Denoted by $\hat{\boldsymbol{\theta}}$, an estimator furnishes an approximation of the parameter $\boldsymbol{\theta}$.

When $\boldsymbol{x}$ represents observed data and $\boldsymbol{\theta}$ signifies unobserved parameters, **Estimation** relates to approximating a parameter's value amidst the inherent variability and uncertainty enveloping incomplete and noisy data. Conversely, **Inference** intertwines prior knowledge and observations to compute the posterior, adhering to Equation 2.33. Bayes' theorem facilitates a pathway for inference, empowering statistical methodologies to judiciously interpret data and amalgamating prior knowledge and newly observed data into a coherent probabilistic framework.

A last observation is related to the parameter vector $\boldsymbol{\theta}$. Given a function $p_X(\boldsymbol{x})$, the parameters relate to the distribution of a random variable X according to $X \sim \text{dist}(\boldsymbol{\theta})$, where the notation represents X following a distribution parameterized by $\boldsymbol{\theta}$. For instance, if X follows a normal (Gaussian) distribution, denoted as $X \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, $p_X(\boldsymbol{x}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$, where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ are the mean and variance parameters, respectively. To simplify our discussion and maintain cohesiveness in our exposition, we will occasionally use the notation $p_X(\boldsymbol{x})$ with the implicit understanding that it refers to $p_X(\boldsymbol{x}; \boldsymbol{\theta})$ in the ensuing text.

2.3.3 The Intractability Problem in Bayesian Inference

In Bayesian inference, parameter estimation and model fitting typically involve probability distribution functions (or densities) that depend on unknown parameters. For a random variable X following a distribution parameterized by $\boldsymbol{\theta}$, we can represent this as $X \sim \text{dist}(\boldsymbol{\theta})$, with the corresponding probability function denoted as $p_X(\boldsymbol{x}; \boldsymbol{\theta})$ or implicitly as $p_X(\boldsymbol{x})$. Conducting inference on the parameter vector $\boldsymbol{\theta}$ involves interpreting it through the lens of data represented by random variables X. This process entails identifying probable values of $\boldsymbol{\theta}$ and evaluating plausible values of its components while accounting for the inherent uncertainty of such assessments [67].

In parameter estimation, the Maximum Likelihood (ML) estimation approach is widely adopted. It is mathematically expressed as follows [68]:

$$\hat{\boldsymbol{\theta}}_{ML} = \arg\max_{\boldsymbol{\theta}} p_X(\boldsymbol{x}; \boldsymbol{\theta}), \tag{2.34}$$

where $p_X(\boldsymbol{x}; \boldsymbol{\theta})$ elucidates the probabilistic relationship between the observations $\boldsymbol{x}$ and the assumed model underlying these observations. Here, $\boldsymbol{\theta}$ specifies the parameters of the distribution under investigation.

In numerous scenarios, directly computing the function $p_X(\boldsymbol{x}; \boldsymbol{\theta})$ morphs into a complex and formidable optimization problem. This computational challenge can be alleviated by incorporating **hidden variables**, or **latent variables**, denoted by Y. While these variables are not directly observable, they establish a connection between observations and parameters via Bayes' rule. They must encompass sufficient information to render $p_{X|Y}(\boldsymbol{x}|\boldsymbol{y})$ computationally manageable [68].

After defining these latent variables and their prior probability, $p_Y(\boldsymbol{y})$, the marginal likelihood can be derived via [68]:

$$p_X(\boldsymbol{x}) = \int p_{XY}(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} = \int p_{X|Y}(\boldsymbol{x}|\boldsymbol{y}) p_Y(\boldsymbol{y}) d\boldsymbol{y} \tag{2.35}$$

Although seemingly straightforward, the integration within Equation 2.35 is pivotal from a Bayesian perspective, facilitating the derivation of the likelihood function and, via Bayes' rule, the computation of the posterior of latent variables:

$$p_{Y|X}(\boldsymbol{y}|\boldsymbol{x}) = \frac{p_{X|Y}(\boldsymbol{x}|\boldsymbol{y}) p_Y(\boldsymbol{y})}{p_X(\boldsymbol{x})} \tag{2.36}$$

After obtaining the posterior, inference concerning the latent variables becomes feasible. Nonetheless, in many instances, the integral within Equation 2.35 is either computationally infeasible or practically elusive to evaluate [68, 67].

In Bayesian inference, the essence lies in methodologies that allow for the bypassing or approximation of this integral. Two primary categories of techniques emerge for this purpose. The first includes numerical sampling methods, commonly known as Monte Carlo techniques, and the second comprises deterministic approximation techniques [68]. The latter will be explored in Section 2.3.4.

2.3.4 Variational Inference

Variational Inference (VI) represents a systematic strategy to approximate complex conditional densities through an optimization paradigm, distinctly different from previously

discussed sampling methodologies. The essence of VI lies in approximating the conditional density of latent variables given observed variables by exploiting a prescribed family of densities and using optimization strategies. This approach uses a family of densities parameterized by "variational parameters", creating a framework where the conditional density is approximated by minimizing the Kullback-Leibler (KL) divergence [69].

To describe this mathematically, consider a family of densities, denoted as $\mathcal{Q}$, over latent variables Y. Each member, denoted by $q_Y(\boldsymbol{y})$ within $\mathcal{Q}$, serves as an approximating candidate for the exact conditional density. Thus, the VI problem, defined through the lens of KL divergence, is expressed as follows [69]:

$$q_Y^*(\boldsymbol{y}) = \underset{q_Y(\boldsymbol{y})\in\mathcal{Q}}{\arg\min}\, \mathcal{KL}[q_Y(\boldsymbol{y})||p_{Y|X}(\boldsymbol{y}|\boldsymbol{x})], \tag{2.37}$$

where $q_Y^*(\boldsymbol{y})$ represents the optimal approximation of $p_{Y|X}(\boldsymbol{y}|\boldsymbol{x})$ from within the family $\mathcal{Q}$.

By using KL divergence as a measure, the optimization process aims for the closest approximation within $\mathcal{Q}$, balancing the trade-off between approximation accuracy and computational cost. The complexity of the optimization process correlates with the complexity of the family $\mathcal{Q}$.

A computational bottleneck arises from the necessity of evaluating the objective function in Equation 2.37, which requires calculating the log evidence, $\log p_X(\boldsymbol{x})$, known for its computational difficulty. This computational challenge is addressed through the KL divergence expansion [69]:

$$\begin{aligned}\mathcal{KL}[q_Y(\boldsymbol{y})||p_{Y|X}(\boldsymbol{y}|\boldsymbol{x})] &= \mathbb{E}_{q_Y}[\log q_Y(\boldsymbol{y})] - \mathbb{E}_{q_Y}[\log p_{Y|X}(\boldsymbol{y}|\boldsymbol{x})] \\ &= \mathbb{E}_{q_Y}[\log q_Y(\boldsymbol{y})] - \mathbb{E}_{q_Y}[\log p_{YX}(\boldsymbol{y},\boldsymbol{x})] + \log p_X(\boldsymbol{x})\end{aligned} \tag{2.38}$$

where $\mathbb{E}_{q_Y}$ represents the expected value under the distribution $q_Y(\boldsymbol{y})$.

The Evidence Lower Bound (ELBO) is habitually employed as an effective surrogate to alleviate this computational challenge. Given a data model $p_X(\boldsymbol{x})$, representing the marginal likelihood or evidence, and a data distribution involving a variable $\boldsymbol{y}$, represented as $p_{YX}(\boldsymbol{y},\boldsymbol{x})$, the ELBO is invoked when a distribution of the latent variable $\boldsymbol{y}$, denoted as $q_Y(\boldsymbol{y})$, is known or assumed [70].

The interplay between the latent variables y and observed data $\boldsymbol{x}$ constructs a scaffold that facilitates this endeavor. A key aspect is to comprehend the marginal likelihood, expressed as $\ln p_X(\boldsymbol{x})$, which quantifies the log-probability of observing the data given the

model p_X. This can be expressed and manipulated as follows [71]:

$$\ln p_X(\boldsymbol{x}) = \ln \sum_{\boldsymbol{y}} p_{YX}(\boldsymbol{y}, \boldsymbol{x}) \tag{2.39}$$

$$= \ln \sum_{\boldsymbol{y}} q_Y(\boldsymbol{y}) \frac{p_{YX}(\boldsymbol{y}, \boldsymbol{x})}{q_Y(\boldsymbol{y})} \tag{2.40}$$

$$= \ln \mathbb{E}_{q_Y} \left[\frac{p_{YX}(\boldsymbol{y}, \boldsymbol{x})}{q_Y(\boldsymbol{y})} \right] \tag{2.41}$$

Navigating through the principles of variational inference, Jensen's Inequality illuminates a path to establish a lower bound. It connects the logarithm of the marginal likelihood with the expectation across an approximated model [71]. To lay this foundation, let us recall Jensen's Inequality:

$$\ln p_X(\boldsymbol{x}) \geq \mathbb{E}_{q_Y} \left[\ln \frac{p_{YX}(\boldsymbol{y}, \boldsymbol{x})}{q_Y(\boldsymbol{y})} \right] \tag{2.42}$$

Here, $p_{YX}(\boldsymbol{y}, \boldsymbol{x})$ denotes the joint distribution of the observed and latent variables, and $q_Y(\boldsymbol{y})$ signifies an approximating distribution aligned to the true posterior of the latent variables Y. Let us focus on the Evidence Lower Bound (ELBO), symbolized by $\mathcal{L}_{q_Y}$:

$$\mathcal{L}_{q_Y}(\boldsymbol{y}, \boldsymbol{x}) = \mathbb{E}_{q_Y} \left[\ln \frac{p_{YX}(\boldsymbol{y}, \boldsymbol{x})}{q_Y(\boldsymbol{y})} \right] \tag{2.43}$$

Inspecting Equation 2.42, it becomes clear that the gap creating the inequality is the Kullback-Leibler (KL) divergence, which measures the dissonance between two probability distributions.

$$\ln p_X(\boldsymbol{x}) = \mathcal{L}_{q_Y}(\boldsymbol{y}, \boldsymbol{x}) + \mathcal{KL}[q_Y(\boldsymbol{y}) || p_{Y|X}(\boldsymbol{y}|\boldsymbol{x})] \tag{2.44}$$

Delving into the KL divergence, $\mathcal{KL}[q_Y(\boldsymbol{y}) || p_{Y|X}(\boldsymbol{y}|\boldsymbol{x})]$, provides insights into the variance between the ELBO and the marginal likelihood, indicating the tightness of the bound. Minimizing this divergence implies that $q_Y(\boldsymbol{y})$ is an apt approximation of the actual posterior, thus optimizing $\ln p_X(\boldsymbol{x})$ and, concurrently, enhancing the approximation $q_Y(\boldsymbol{y})$ [72].

The crux of the matter revolves around maximizing the ELBO, which is equivalent to refining the KL divergence, excluding the constant term $\log p(\boldsymbol{x})$ [69]. The ELBO conveys

its mathematical essence as:

$$\begin{aligned}\mathcal{L}_{q_Y}(\boldsymbol{y}, \boldsymbol{x}) &= \mathbb{E}_{q_Y}[\log p_{YX}(\boldsymbol{y}, \boldsymbol{x})] - \mathbb{E}_{q_Y}[\log q_Y(\boldsymbol{y})] \\ &= \mathbb{E}_{q_Y}[\log p_Y(\boldsymbol{y})] + \mathbb{E}_{q_Y}[\log p_{X|Y}(\boldsymbol{x}|\boldsymbol{y})] - \mathbb{E}_{q_Y}[\log q_Y(\boldsymbol{y})] \\ &= \mathbb{E}_{q_Y}[\log p_{X|Y}(\boldsymbol{x}|\boldsymbol{y})] - \mathcal{KL}[q_Y(\boldsymbol{y})||p_Y(\boldsymbol{y})]\end{aligned} \tag{2.45}$$

In this intricate process, ELBO optimization balances two crucial facets: (1) amplifying the expected log-likelihood of the observed data and (2) minimizing the divergence between the approximate and true posterior distributions, illustrating a harmony between likelihood and prior [69]. This variational framework lays the foundation for advanced methodologies, like variational autoencoders, within neural data encoding, fostering a delicate balance between data adherence and model complexity.

2.3.5 Variational Autoencoders

Variational Autoencoders (VAEs), introduced by Kingma and Welling, pioneered a paradigm shift in deep latent variable models by combining deep learning and Bayesian inference. This formed a novel methodology for model training and inference through stochastic gradient descent [72]. In understanding VAEs, distinguishing between generative and discriminative modeling is imperative. While discriminative models predict dependent variables given certain features, generative models, such as VAEs, aim to understand the joint distribution of observed and latent variables, providing a fuller, generative representation of the data.

VAEs are a cornerstone within probabilistic modeling, leveraging variational inference to streamline the optimization of complex, high-dimensional data. Instead of directly defining the data distribution $p_X(\boldsymbol{x}; \boldsymbol{\theta})$, the focus is on sculpting a model, denoted as $p_{Y|X}(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta})$, that accurately mimics the actual distribution $p_*(\boldsymbol{y}|\boldsymbol{x})$ via latent variables $\boldsymbol{y}$ and is conditioned on observable variables $\boldsymbol{x}$, parameterized by $\boldsymbol{\theta}$ [72]. The optimization attempts to ensure:

$$p_{Y|X}(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) \approx p_*(\boldsymbol{y}|\boldsymbol{x}) \tag{2.46}$$

The strategic utilization of neural networks becomes pivotal here due to their substantial expressiveness in representing these distributions. Despite the possible simplicity of the conditional distribution $p_{Y|X}(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta})$, the marginal distribution $p_X(\boldsymbol{x}; \boldsymbol{\theta})$ can encapsulate a rich complexity, introducing arbitrary dependencies among variables [72]. However, even with the proficient capabilities of deep models, the intractability problem, specifically regarding the computation of $p_X(\boldsymbol{x}; \boldsymbol{\theta})$, remains a challenge.

A committed approach to navigate the intractability problem, inspired by Variational Inference, is to introduce a parametric model, symbolized as $q_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\phi})$ and referred to as the encoder, as proposed by [22]. Given:

$$q_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\phi}) \approx p_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}) \tag{2.47}$$

the optimization is conducted via SGD, detailing the VAE components and their respective mappings in Figure 2.7. Here, the encoder, expressed as $q_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\phi})$, approximates the true posterior $p_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$, which is inherently intractable. Therefore, the encoder operates as a stochastic mapping, connecting the observed variable $\boldsymbol{x}$ to the latent space $\boldsymbol{y}$, with its distribution representing the prior of the generative model, symbolized as $p_Y(\boldsymbol{y};\boldsymbol{\theta})$. Concurrently, the decoder $p_{X|Y}(\boldsymbol{x}|\boldsymbol{y};\boldsymbol{\theta})$ performs a stochastic mapping from the latent space $\boldsymbol{y}$ to an observed space region.

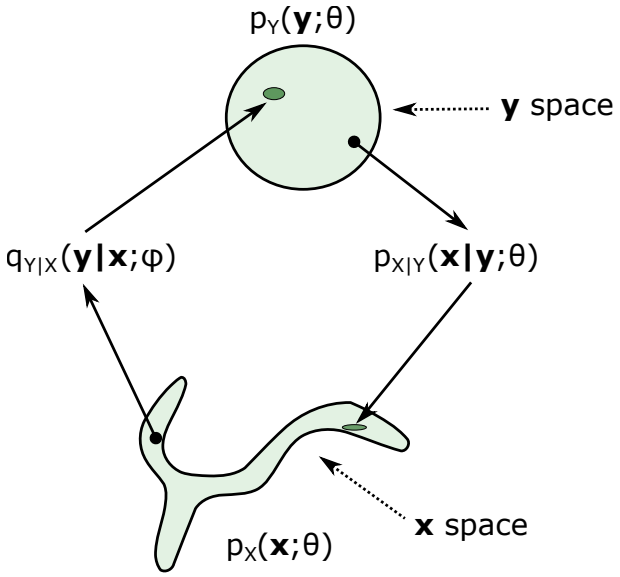


Figure 2.7: Visualizing VAE's stochastic mappings between observed and latent spaces. VAEs establish a network of probabilistic mappings between the observed space $\boldsymbol{x}$ and the latent space $\boldsymbol{y}$, with respective distributions $p_X(\boldsymbol{x};\boldsymbol{\theta})$ and $p_Y(\boldsymbol{y};\boldsymbol{\theta})$. [72].

Efficiency in the VAE's optimization lies in its ability to leverage the latent space $\boldsymbol{y}$, which embodies a simpler, typically spherical distribution, as a powerful tool for capturing and replicating the complexities and variances within the observed data $\boldsymbol{x}$. Here, the encoder functions as a probabilistic function that maps observed variables to a latent space, which is further exploited by the decoder to generate new data samples, feeding into the generative model capabilities of VAEs [72].

The VAE optimization aims not only to model but also to generate new data by drawing from this latent space, ensuring it adheres as closely as possible to the original data distribution $p_X(\boldsymbol{x};\boldsymbol{\theta})$. It holds potential for productive applications through proficient

data generation and approximation. Concurrently, the model, via the optimization of parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, attempts to minimize the divergence between the encoder's approximation $q_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\phi})$ and the true posterior $p_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$, creating a coherently organized latent space and fostering enhanced generative capabilities. The objective function of VAEs, similar to other variational methods, revolves around the ELBO. Optimizing the ELBO coincides with optimizing the KL divergence. The relationship between the ELBO and the KL divergence can be established based on the modeling context of this section [22, 72]:

$$\begin{aligned}
\log p_X(\boldsymbol{x};\boldsymbol{\theta}) &= \mathbb{E}_{q_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\phi})}[\log p_X(\boldsymbol{x};\boldsymbol{\theta})] \\
&= \mathbb{E}_{q_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\phi})}\left[\log\left[\frac{p_{XY}(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta})}{p_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})}\right]\right] \\
&= \mathbb{E}_{q_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\phi})}\left[\log\left[\frac{p_{XY}(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta})}{q_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\phi})}\cdot\frac{q_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\phi})}{p_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})}\right]\right] \\
&= \mathbb{E}_{q_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\phi})}\left[\log\left[\frac{p_{XY}(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta})}{q_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\phi})}\right]\right] + \mathbb{E}_{q_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\phi})}\left[\log\left[\frac{q_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\phi})}{p_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})}\right]\right] \\
\log p_X(\boldsymbol{x};\boldsymbol{\theta}) &= \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}) + \mathcal{KL}(q_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\phi})||p_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}))
\end{aligned} \tag{2.48}$$

From the derivation showcased in Equation 2.48, and considering the properties of $\mathcal{KL}$, it is clear that maximizing $\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x})$ with respect to parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ optimizes two critical aspects. The first pertains to the maximization of the marginal likelihood $p_X(\boldsymbol{x};\boldsymbol{\theta})$, indicating enhancements in the generative model. Simultaneously, it minimizes the KL divergence of the distribution $q_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\phi})$ with respect to the true posterior $p_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$, thereby fine-tuning $q_{Y|X}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\phi})$ to be a more accurate approximation.

2.4 Summary

In conclusion, this chapter serves as a cornerstone in laying the theoretical groundwork for the research endeavors outlined in this doctoral thesis. By delving into the intricacies of image compression, neural networks, and Bayesian inference, the chapter provides a comprehensive overview of key concepts essential for understanding the proposed approaches. In exploring image compression, fundamental techniques such as arithmetic coding and transform coding are elucidated. It also offers readers a deep understanding of the historical trajectory and evolution of classic CODECs. Integrating machine learning and neural networks into image compression techniques signifies a paradigm shift, promising enhanced performance and adaptability in multimedia and telecommunication applications.

The discussion on neural networks covers various aspects, including feedforward neural networks, convolutional neural networks, recurrent neural networks (RNNs), and activation functions. RNNs' ability to process sequential data and capture temporal dependencies opens avenues for applications in natural language processing, time-series analysis, and more. Trainable activation functions represent advancements in neural network design, offering flexibility, expressiveness, and robustness in model representation and inference.

The chapter also emphasizes the importance of foundational concepts from information theory and variational Bayesian inference, providing readers with the necessary background to effectively engage with the proposed research. The appendix detailing information theory serves as a valuable resource for readers seeking further clarification and understanding. Additionally, the section on Bayesian inference provides a deep understanding of the cornerstone model in this thesis, the Variational Autoencoders (VAEs).

As the chapter concludes, its significance becomes evident as a foundational pillar for subsequent chapters. The theoretical framework established here not only supports the thesis's arguments and findings but also equips readers with a discerning lens to appreciate the approaches and methodologies explored and proposed in the following chapters.

Chapter 3

Literature Review

Neural networks are known to perform well in image compression. Until recently, there was little evidence that training a competitive neural network for various images and rates was possible. Nevertheless, several studies now demonstrate the power of neural networks in compressing multiple data types.

This chapter reviews the most significant works in neural image coding. Non-recurrent neural networks are discussed in Section 3.1, while recurrent neural network techniques are covered in Section 3.2. Approaches based on variational autoencoders, the main methods used in the field, are detailed in Section 3.3, which is further subdivided into topics such as entropy model techniques, computational performance improvements, variable rate methods, and attention-based techniques, among others.

3.1 Non-recurrent Neural Networks

Neural networks have recently become state of the art in semantic tasks such as image classification and object detection. They also achieve high performance in low-level tasks like super-resolution and compression artifact reduction by exploring the hierarchical correlation between neighboring values using cascaded operations [7].

One of the first significant approaches to using neural networks in image coding is based on a cascade of autoencoders [15]. In this strategy, each autoencoder compresses and reconstructs its input, with the reconstruction error propagated as input to the next autoencoder, repeating until a specified number of autoencoders. This approach, shown in Figure 3.1, results in each autoencoder specializing in compressing and reconstructing residuals. The strategy can be summarized in the following equations, where $\mathbf{x}$ is the

input image, t is the index of the autoencoders, and $\mathbf{r}_t$ is the t-th generated residual:

$$\mathbf{r}_0 = \mathbf{x} \tag{3.1}$$

$$h_t(\mathbf{r}_{t-1}) = g_t(b(f_t(\mathbf{r}_{t-1}))) \tag{3.2}$$

$$\mathbf{r}_t = h_t(\mathbf{r}_{t-1}) - \mathbf{r}_{t-1} \tag{3.3}$$

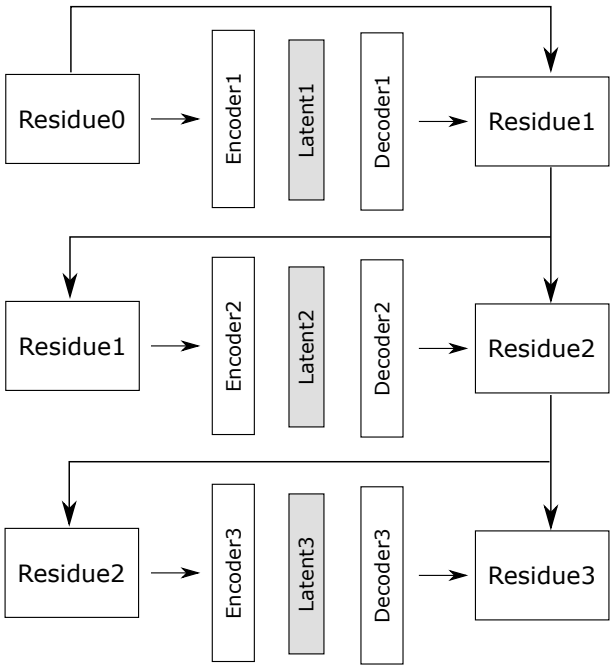


Figure 3.1: Multiple encoder-decoder pairs progressively refine residuals.
An architecture where several pairs of encoder and decoder progressively operate on the residuals. The original block is the level 0 residual. From the reconstruction of this residual, the error is calculated and propagated as a level 1 residual, and so on [15].

In this case, the pair (f_t, g_t) represents the autoencoder for the t-th level of residuals, where f_t is the encoder and g_t is the decoder. Given the input residual for level $t-1$, the residual at level t is defined by the reconstruction error between the input $\mathbf{r}_{t-1}$ and the reconstruction $f_t(\mathbf{r}_{t-1})$. The term b represents the binarizer. Since the binarization step is not differentiable, the authors proposed a training strategy where the output of the last layer of the encoder, with hyperbolic tangent activation, was probabilistically binarized. They considered the derivative of the operation in a statistical sense through expectation.

The authors used 16 levels of residuals in this work, with an input size of 32×32. An advantage of this strategy is that the coding is progressive, with each level encoding the residuals of the previous level, incrementally increasing the rate. The dataset for training consisted of 216 million random images collected from the internet, with 90% used for training and 10% for testing.

In the same work, the authors analyzed different variations of neural network architectures with fully connected and convolutional layers [15]. This work is significant as it was one of the first to propose using neural networks for compression tasks. Subsequent improvements based on this proposal have been suggested and will be presented later.

3.2 Recurrent Neural Networks

Unlike non-recurrent approaches that rely on fully connected or convolutional layers, recurrent models use recurrent layers. A recurrent neural network uses memory to store information about past inputs, with memory units having connections to themselves. This temporal information alters the layer's behavior concerning the current input [7].

Various layer implementations have been proposed to maintain temporal information in neural networks. One of the earliest effective approaches is the use of LSTM (Long Short-Term Memory). Subsequent improvements introduced GRU (Gated Recurrent Unit), which simplifies the recurrent component while maintaining the performance of LSTM in some scenarios [7].

The work proposed by Toderici et al. [16] introduces an improvement over their previous work [15], specifically related to memory components. Instead of using multiple autoencoders, this approach utilizes a single network with memory components. The reconstruction residue feeds back into the network, which uses this memory to attempt a higher quality reconstruction of the image. Figure 3.2 illustrates this strategy. In this approach, the latent layer is progressively composed. The authors analyzed the performance of several layers, including LSTM and GRU, with GRU yielding the best results.

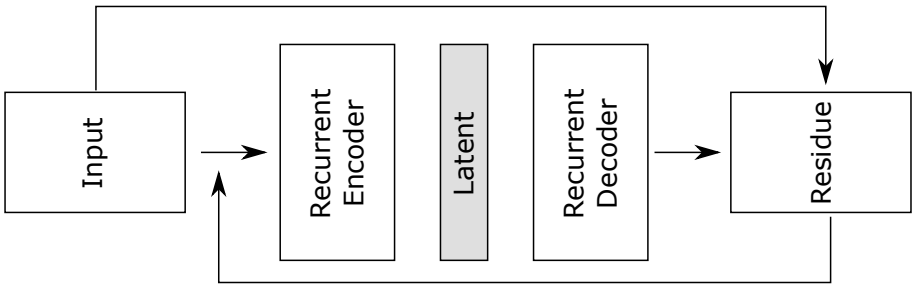


Figure 3.2: The architecture consists of a single autoencoder with memory layers. The input is either the original image or the reconstruction residue [16].

Additional improvements included the introduction of a simultaneously trained network and scaling of the input residues to ensure different levels of residues were within an expected range to facilitate network convergence. Moreover, the binarization process was enhanced by adopting an LSTM layer to model a context passed to an arithmetic encoder.

Thus, the approach also introduces an architecture capable of performing compression at variable rates. However, a significant issue is the lack of flexibility and the complexity in training a memory model that performs multiple iterations over the data and residues [16].

The architecture proposed by Toderici et al. [16] has been enhanced in several works. Covell et al. addressed the rigidity issue in bit allocation [17]. The original proposal had no criterion for bit allocation. However, allocating more bits to complex regions and fewer bits to simple image regions can be beneficial. The authors propose this allocation based on a desired quality level. A mask is used in the binarization layer before code transmission to signal the following situations:

- The quality objective has been achieved;
- The encoder output is accidentally composed solely of zeros;
- A mask was applied to the code in the previous iteration.

In other words, it controls when additional iterations are no longer necessary to improve the quality of the given block. The authors employ strategies so that the network recognizes this pattern of latent components composed only of zeros as a signal that no further refinement is needed, given the image's quality level.

A concept inspired by traditional coding strategies, based on the architecture proposed by Toderici et al., is presented by Minnen et al. In this approach, DPCM is simulated using a neural network [18]. Here, the autoencoder processes only the residues obtained in the initial stage instead of directly encoding the image. Adjacent blocks are used as context for prediction, as shown in Figure 3.3.

Johnston et al. [20] modify the training strategy to enhance the neural network's generalization by better exploring the hidden states of the recurrent layers. This strategy, termed "priming", involves running several "fake" iterations before generating binary codes (encoder) or a reconstructed image (decoder) [16]. These fake iterations are not considered in the final flow but are performed before iterations to improve the initialization of the hidden states of memory layers. Figure 3.4 shows an example of "priming" with one "fake" iteration.

A generalization of the priming strategy is also proposed, termed "diffusion". In this case, for each real iteration for a block, "fake" iterations are interspersed, further refining the hidden state of the recurrent layers. Although the diffusion strategy is much more computationally intensive, it yields the best-reported results. The main drawback of this approach is that it introduces even more complexity to the training of an architecture that already requires extensive training.

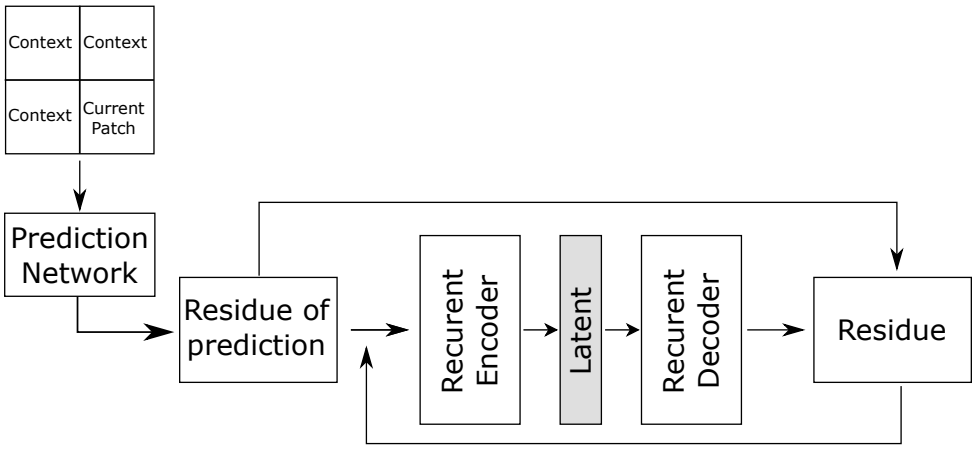


Figure 3.3: The architecture adds a context prediction network, which also follows the architecture of an autoencoder. In this case, the input to the encoding network is no longer an image block but the residue of context prediction as proposed by Minnen et al. [18].

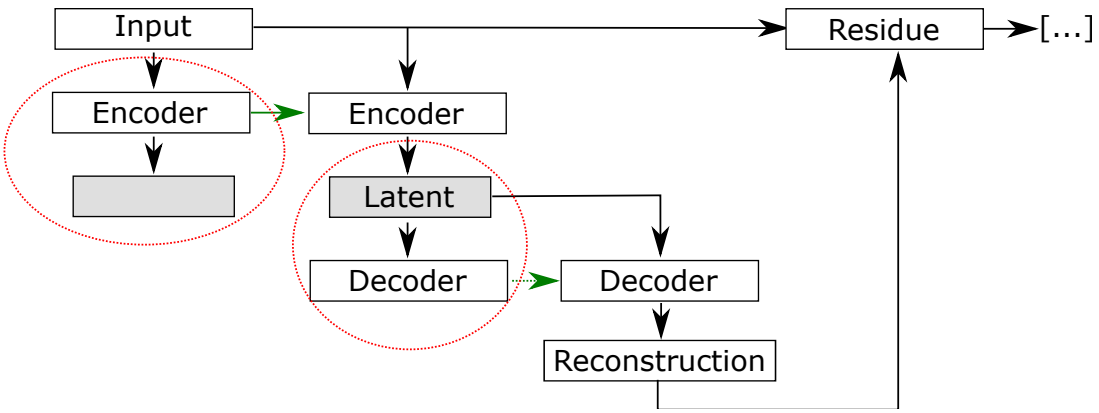


Figure 3.4: The priming strategy encodes the input once, and the latent representation is discarded. Only the encoder's state is propagated forward, and the image is fed again. The latent representation feeds the decoder twice, with the state of the first being propagated to the second [20].

3.3 Variational Autoencoders

Variational Autoencoders (VAEs) are neural networks that implement an approximate model of the *a posteriori* in the context of variational Bayesian inference [22]. Since their inception, VAEs have been extensively used as generative data models. However, they have also proved to be very useful for encoding tasks. In encoding, these *a posteriori* models are not used to generate new data but to model the density of the code space.

VAEs were adopted in the context of encoding as specific modeling choices led the original loss function to become equivalent to the rate-distortion function [45]. The code space model is obtained by relaxing the quantization operation. Since this operation is non-differentiable, it is replaced by additive uniform noise of unit width during training. Moreover, like in Kingma's work [22], the likelihood is modeled as a Gaussian with constant variance and mean represented by the decoder. A flowchart of this approach is described in Figure 3.5.

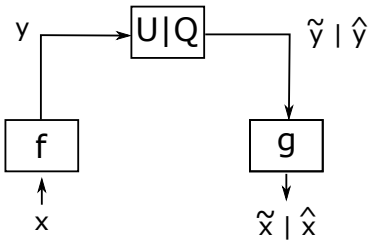


Figure 3.5: Function f represents the encoder, and g represents the decoder. In this case, the model for $\mathbf{y}$ is a non-parametric distribution. The components U and Q represent the quantization at training and testing times, where uniform noise is added to $\mathbf{y}$ during training. This notation leads to the variables with the tilde accent ($\sim$) and the circumflex accent ($\wedge$) representing the values obtained at training time and testing time, respectively [23].

A factorized distribution models the code space. When this restriction is adopted in variational inference, the approach is termed the *mean-field approximation*. It is the most common type of variational inference as it is conceptually simple, easy to implement, and particularly feasible for problems with a large number of latent variables [73].

Various models with different λ values are trained on 6507 images from ImageNet [74]. The entropy encoder is an implementation inspired by CABAC (context-adaptive binary arithmetic coding), and the evaluation is performed on the Kodak dataset [75]. With the proposed equivalence, the authors introduce variational autoencoders in the context of encoding, optimizing both the latent space and the reconstruction. This approach has become the basis for many improvements and shows superior results compared to methods based on Toderici's approach [15].

Ballé et. al. proposed an improvement in a later work [23], where their original idea is enhanced with the introduction of a hyper *a priori*. The authors model the latent space using a Gaussian with a mean of 0 and parametrized variance. To estimate the variance, a new VAE serves as an *a priori* over the *a prior* (hence the name hyper *a priori*). The output of this component is the variance vector of the data's entropy model. The block diagram of this approach is displayed in Figure 3.6. As the hyper *a priori* is a VAE, it also has its own latent space that adopts the non-parametric modeling of their earlier approach [45]. This information is considered additional (side information) to the main latent.

In this case, training was conducted on a 1 million JPEG web image database, sized 3000×5000. The images were sub-sampled to the size of 640×1200, from which random crops of size 256×256 were taken. This sub-sampling was adopted to mitigate artifacts produced by the pre-encoding JPEG of the images. This architecture, which employs the hyper *a priori*, is the most adopted in subsequent works. The variational approaches described in this chapter will be presented in detail in the following chapter since they are the reference architectures used in this work. It is worth mentioning that the hyper *a priori* approach [23] is the basis of a wide range of proposals.

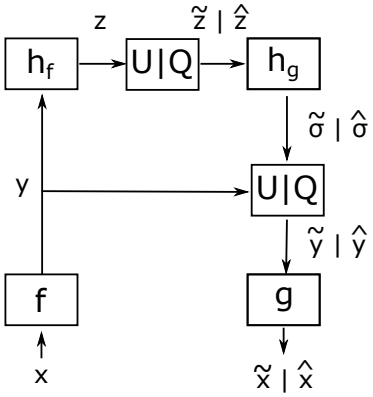


Figure 3.6: The hyperprior architecture proposed in [23].

Minne's work [24] improves over Ballé's hyperprior approach [23] including an autoregressive component. It was the first work to surpass HEVC (High Efficiency Video Coding) in intra-prediction mode for PSNR (peak signal-to-noise ratio). The hyperprior combined with the autoregressive component produces both parameters of the Gaussian modeling of the central latent's entropy: mean and variance. An entropy parameter network receives both the output of the hyperprior and the output of the autoregressive context component to produce the parameter vectors. A significant weakness of this work is its low feasibility in real scenarios with large images due to the autoregressive component. Furthermore, the results reported by the authors are estimated according to Shannon entropy, not by actually performing the inference process.

The role of non-linear transforms in VAEs is further studied in another paper of Ballé [76]. This work comprehensively reviews non-linear transform coding (NTC) methods, showcasing their competitiveness against leading linear transform codecs for image compression. The empirical assessment of NTC's rate-distortion performance employs simple source examples, facilitating optimal quantizer performance estimation. The study introduces an innovative variant of entropy-constrained vector quantization and analyzes stochastic optimization techniques for NTC models. Architectures of artificial neural network-based transforms and learned entropy models are reviewed. The paper also directly compares various methods for parameterizing the rate-distortion trade-off of non-linear transforms. The variational approaches mentioned in this section will be explained in greater detail in the following chapters, as they are the backbone architectures used in this work.

3.3.1 Approaches Focusing on the Entropy Model

One of the main lines of improvement applied to the previously mentioned works is the adoption of different entropy models, as it is the case of Minnen's approach [25]. The standard approach introduced by Ballé [45] is to learn a fixed entropy model through non-linear transforms optimizing the rate-distortion. However, this global model is not entirely suitable for all images. Side information is passed to specify specific entropy models for each image to make the architecture more flexible. Among the local entropy models generated a priori, the one that minimizes latent entropy is chosen.

Li et. al. [26], based on Minnen's work [24], propose different modeling strategies that address the inefficiency of the standard autoregressive component. Independence conditions are adopted between groups of latent elements, and a zig-zag analysis is conducted to parallelize some calculations. Another introduced change is that the code is three-dimensional.

A multiscale autoregressive approach, based on Minnen's work [24], is presented by Zhou et al. [27]. The authors use causal masks of sizes 3×3, 5×5, and 7×7. The outputs of this multiscale autoregressive model feed a parameter network, along with the output of the hyperprior. Another difference is that the input to the hyperprior is a composition of feature maps from various encoder layers. These multiscale approaches have been quite common in neural networks. Another interesting idea presented in the paper is to have the arithmetic encoder ignore maps almost entirely composed of zeros, saving bits.

Jiang et al. [77] introduce advancements in learned image compression, focusing on the role of entropy models in enhancing rate-distortion performance. Existing entropy models often capture correlations in a single dimension, neglecting channel-wise, local, and global spatial correlations in latent representations. To address this limitation, the

authors propose multi-reference entropy models (MEM and MEM+) capable of capturing these various contexts. The latent representation is divided into slices, and during decoding, previously decoded slices serve as contexts. An attention map of the preceding slice predicts global correlations in the current slice, addressing local and global spatial contexts.

Duan et al. [78] explore the intersection of generative modeling and lossy image compression, drawing inspiration from the theoretical connection between variational autoencoders (VAEs) and rate-distortion theory. Building upon ResNet VAEs, initially designed for data distribution modeling, the authors redesign the latent variable model with a quantization-aware posterior and prior. The QRes-VAE (Quantized ResNet VAE) model adopts a hierarchical architecture for entropy modeling, encoding, and decoding coarse-to-fine images.

3.3.2 Improvements Regarding Computational Performance

Computational cost is one of the major hindrances to applying highly complex neural architectures. As mentioned earlier, the approach proposed by Minnen [24], which applies an autoregressive component, has restricted applicability due to the non-parallelism of this component. Various works have focused on improving the efficiency of some models, either by changing the entropy modeling [26] or by optimizing other features of the modeling.

The need to perform complete training for multiple models with different λ's can be problematic. In Theis' work [28], the authors propose computationally efficient ways to train models. They train networks with different optimization hyperparameters and adopt a second hyperparameter to refine the rate-distortion learned by the reference model. Only one refinement training is performed on the previous model with this new hyperparameter, generating multiple models from each reference model. Although the standard optimization is superior, this strategy allows training more models with minor performance losses.

A work focusing on improving computational efficiency through changes in the entropy model can be seen in another Minnen's approach [29]. The authors propose independence between the channels of the latent representation, allowing all components of the same channel to be processed in parallel. The autoregressive component is applied channel by channel, introducing a higher level of parallelism than the element-by-element autoregressive approach.

Hu et al. [30] abandon the autoregressive component and replace it with hierarchical modeling based on a set of hyperpriors, as shown in Figure 3.7. The components make increasingly finer data analyses in this hierarchy of priors. Additionally, the hyperdecoders not only predict the parameters of the distributions but also feed the primary

decoder to assist in the reconstruction, better utilizing the information passing through these elements.

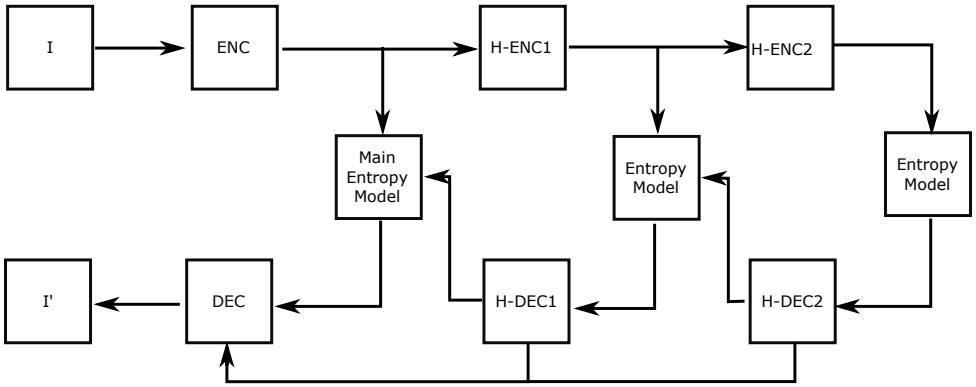


Figure 3.7: The approach adopts a hierarchy of autoencoders. The process reports the input I and the reconstruction I'. In this case, the functions $H - ENC$ and $H - DEC$ portray multiple hierarchical hyper *a priori*. Each processes the latent information of the preceding one. These hyper-decoders produce the parameters for the component hierarchically below and feed the main decoder to aid in the reconstruction [30].

An approach focused on optimizing network layers with hyper *a priori* [23] is presented by Johnston et al. [31]. The authors argue the choice of the number of layers and filters is arbitrary. They focus on improving the decoders' performance using neural network pruning techniques, achieving different network configurations. Specifically, Lasso regularization is adopted to identify the adequate number of filters. However, the process could not be fully automated as excessive restrictions sometimes occurred in predicting the hyper-decoder.

Luo et al. [79] address some applicability challenges of Learned Image Compression (LIC), particularly the high memory cost, which hinders their application on various devices, especially portable and edge devices. They identify the over-parameterization of the hyperprior module and the redundancy in its latent representation as contributors to this memory cost. To address this, they propose a pruning method called Enhanced Resrep on Hyper Path (ERHP), aimed at reducing the memory consumption of hyperprior modules while enhancing network performance.

3.3.3 Variable Rate Models

The approaches presented earlier [45], [23], and [24] yield arbitrary compression rates with different images. Moreover, this compression rate is fixed for a given image, making it impossible to vary the rate. Several works have been proposed to overcome this limitation and obtain architectures that produce variable compression rates. This class of approaches is closely related to the proposals presented in this thesis.

Conditional autoencoders are used for conditional data generation [80, 81]. Conditioning variables, such as labels and attributes, feed the model. Choin et al. [32] use the Lagrangian λ as the conditioning variable to vary the rate produced. Another change is the adoption of universal quantization, which receives a parameter Δ controlling the precision of the rounding operation. The network is trained with different Δs, and at inference time, this parameter is also used to vary the rate.

Cai et al. adopt a multiscale residual decomposition, composing different latent representations as more scales are used [33]. An allocation mechanism determines the number of scales to achieve a target compression rate. Cui et al. [34] propose a G-VAE (Gained Variational Autoencoder) architecture where a gain unit scales the produced latent. This allows rates to be adjusted by modifying these gain vectors, achieving continuous rate adjustment.

A sophisticated strategy to obtain multiple rates with a single model is presented by Yang et al. [35]. The authors change the entropy encoder, enabling the strategy in any variational autoencoder with mean field distributions. They modify the arithmetic encoder to handle continuous values. The Bayesian arithmetic encoder uses ideas from the standard encoder, such as the cumulative distribution function and an "uncertainty region". Instead of a rigid uncertainty region, the goal is to find a point that identifies the latent variables with high probability according to the variational distribution, minimizing the number of bits.

Cui et. al. [82] address continuous rate adaptation by proposing the Asymmetric Gained Variational Autoencoder (AG-VAE). AG-VAE introduces gain units for discrete rate adaptation and employs an exponent interpolation formula for continuous rate adaptation without extra training. It also uses an asymmetric Gaussian entropy model for accurate entropy estimation. This framework generalizes to various VAE-based compression methods and improves entropy estimation.

The use of progressive encoding to achieve variable bitrate is studied in Lu's work [83]. This paper introduces PLONQ, a progressive neural image compression scheme that enables quality-scalable coding with a single bitstream. Unlike existing solutions, PLONQ simplifies rate control and reduces storage requirements by allowing truncation of a single bitstream for different qualities. It progressively refines latent representations from coarse to fine quantization levels by leveraging latent scaling and introducing nested quantization.

The approach of Baldassarre [84] enhances vector-quantized autoencoders for image compression, addressing their limitation in allocating variable bits based on semantic content or local saliency. The proposed method combines product quantization (PQ) with structured dropout, enabling some degree of rate control for different image regions.

The PQ-autoencoder is trained end-to-end with structured dropout, selectively masking a variable number of codes at each location. This mechanism forces the decoder to reconstruct the image based on partial information, allowing local rate control.

Many other approaches have been proposed regarding variable bitrate. For example, Kao et al. [85] adopt Region-of-Interest control to vary the bitrate compression. Liang's work [86] introduces bitrate flexibility using spatial importance through a gain unit that applies an importance mask. Tong et al. [87] use a quantization error regulator vector aligned with predefined Lagrange multipliers to achieve variable rate. Duan et al. [88] employs a hierarchical quantization-aware ResNet VAE architecture with an adaptive layer normalization operation for variable-rate compression. Cai's approach [89] introduces an Invertible Activation Transformation (IAT) module on a single-rate Invertible Neural Network (INN) based model, feeding the quality level (QLevel) into the IAT to generate scaling and bias tensors. Lee's work [90] introduces Selective Compression of Representations (SCR) for variable-rate image compression using neural networks, which selectively compresses latent representations using a 3D importance quality-adjusted map.

3.3.4 Approaches with Different Neural Operations

One of the leading introductions of modified neural operations comes from octave convolutions, proposed in Chen's work [36]. Natural images usually treat different frequency components distinctly: high frequencies are associated with fine details, and low frequencies with global structure. The authors factorize the convolution activation maps, separating operations into two frequency bands. This approach improves recognition task results and reduces computational cost. Figure 3.8 depicts a flowchart of this new convolution operation.

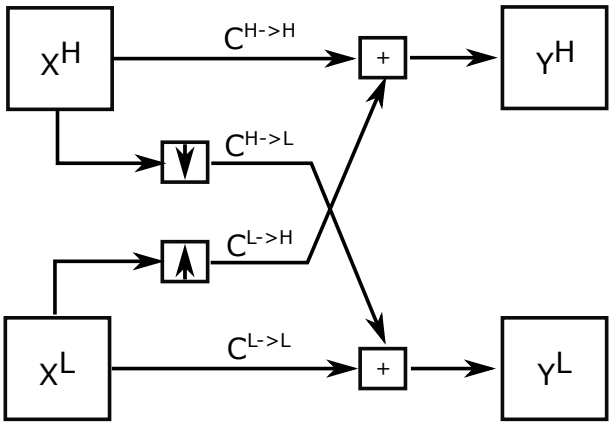


Figure 3.8: The approach introduces high-frequency components H and low-frequency components L. A pooling operation and a supersampling operation allow the exchange of information between the two components [36].

The applicability of octave operations in encoding was first addressed in Akbari's work [37]. The hyper *a priori* architecture from Minnen's approach [24] was adopted for analyses. However, the authors concluded that standard octaves do not outperform regular convolution. The pooling and supersampling operations discard information relevant to encoding. Therefore, the authors propose a generalization of the octaves where convolutions replace subsampling and supersampling operations. This variation achieved superior results, competing with newer encoding standards like the VTM.

Lin et al. [38] propose an extension of the generalized octaves from Akbari's work [37]. The authors add a network to scale the convolution outputs with a parameter λ, naming it modulated octave convolution. These operations preserve more spatial structure. The architecture used to validate the results is derived from Ballé's hyperprior approach [23], and the performance surpasses the VTM (VVC Test Model) in some scenarios.

3.3.5 Approaches with Bit Allocation and Attention Modules

Neural image compression methods are limited in modeling and exploring spatial variation and dependence of image contents. The invariant bit allocation is generally adopted, while image content is spatially variant. More complex regions are usually essential for constituting an image [39]. Given this, several proposals aim to develop strategies to improve bit allocation. Another approach closely related to this idea is using attention modules in neural networks.

In Zhou's work [40], the authors propose an image compression approach for low compression rates, inspired by Minnen's autoregressive idea [24]. A PixelCnn++ represents the autoregressive component [91]. Non-local residual attention modules [92] are used to capture global data correlations. These modules can improve compression by allocating more bits to important areas unsupervised. This architecture achieved the best result at the CLIC 2019 competition.

A similar strategy with non-local attention modules is adopted in Chen's and Liu's works [41, 42]. These modules enhance the capture of global and local correlations and generate masks relevant to the images. These masks weigh the features, producing an implicit bit allocation effect based on the element's importance.

An approach with a similar idea that does not use variational autoencoders is presented in Li's work [39]. Although it is not within the scope of VAEs, it is worth mentioning due to its relevance. The authors propose an encoding network, a decoding network, and a network of importance maps. These maps are applied to the quantized code to truncate elements based on their importance.

Jeny et al. [93] address the challenge of retaining local redundancies in non-repetitive patterns. The proposed method utilizes an autoencoder-style network with three blocks:

the adjacent attention block (AAB), the Gaussian merge block (GMB), and the decoded image refinement block (DIRB). The AAB captures spatial correlations vertically and horizontally, removing unnecessary information and improving entropy coding efficiency. The GMB simulates the distribution of latent representations, enhancing rate-distortion optimization performance. To mitigate compression artifacts, the DIRB leverages global information to improve the quality of reconstructed images.

3.3.6 Post-Processing Based Approaches

Another approach in neural image coding involves adding a post-processing component to the autoencoders' reconstruction. A joint optimization scheme for coding and quality enhancement, termed JointIQ-Net, is proposed. This scheme is the first to surpass VTM [43]. The authors suggest strategies for context modeling, dividing them into local and non-local contexts. The reported results were achieved with up to 3 million training iterations on a CLIC database.

3.3.7 Generative Compression

A generative adversarial model approach is adopted by Agustsson et al. to produce pleasing images at low rates [44]. The authors propose two types of generative compression: global data generation and selective data generation. In the latter, the network input is the image and its semantic segmentation. An encoder generates a bitstream, which is processed by a generator/discriminator pair. The generator receives this bitstream and the semantic map, learning to replicate the encoder's data or create new data based on a binary mask applied to the latent. This mask indicates the regions for compression and generation, influencing the loss function.

A similar idea of generating data at low compression rates is adopted in Huang's work [94]. This approach uses an autoencoder architecture that represents data at three different scales. The authors couple the output of this autoencoder with a discriminator to enhance the autoencoder's reconstruction at lower rates.

3.3.8 Target-based approaches

The core approaches explored in this thesis focus on fine-tuning neural models for specific target bitrate compression. This problem has not been well studied in the literature, with few examples available. One example introduces distortion-constrained optimization (D-CO) as an alternative to λ-VAE for training end-to-end learned models in lossy compression [46]. The proposed D-CO uses constrained optimization to achieve a rate

subject to a distortion constraint. A drawback of this method is the need to search the λ space to fine-tune the model's loss.

A unique approach to target-bitrate constrained optimization is presented by Zhang et al. [47]. This paper introduces a Rate Controllable Variational Autoencoder (RC-VAE) for image compression, designed to adapt to specific target rates. Unlike models in Section 3.3.3, which have multiple arbitrary operation points, this approach targets specific bitrates in a variable rate model. The proposed model includes a Rate-Feature-Level (RFL) neural component that outputs quantization levels to control the primary model's rates. This approach requires a complex, multi-stage training procedure. The task's difficulty is evident in the rapid saturation of quality reconstruction results at low bitrates, highlighting the challenges of rate-constrained optimization, especially with a single model.

3.3.9 Studies of quantization

One of the main problems in neural-based compression is the non-differentiability of the quantization operation. This thesis explores some of these issues. Various works have examined the impact of differential approximation of quantization from different perspectives. Guo et al. [95] address quantization challenges in neural image compression, analyzing three methods: additive uniform noise, straight-through estimator (STE), and soft-to-hard annealing. They identify that while additive uniform noise is advantageous for learning expressive latent spaces, it suffers from train-test mismatch. This mismatch is highlighted in this thesis. On the other hand, STE and annealing methods avoid this mismatch but compromise latent representation ability. To reconcile these issues, the authors propose a soft-then-hard quantization strategy. This involves a two-stage training process: first using additive uniform noise to train a powerful encoder softly, followed by hard quantization for decoder tuning.

Pan et al. [96] focus on challenges in quantization approximation for learned lossy image compression, particularly the non-differentiability of quantization and its impact on neural network training. While prior research has proposed different approximation methods, there is a lack of theoretical understanding of the mechanisms behind their success. The authors highlight three critical gaps: Discrete Gap, Entropy Estimation Gap, and Local Smoothness Gap. These gaps characterize quantization approximation. Visualizations demonstrate the significant influence of different quantization methods on latent distribution and compression performance. To mitigate these gaps, the authors propose a similar approach as Guo [95] with adaptations to address the gaps studied, aiming for a more meaningful entropy estimation during training.

3.3.10 Parameter-adaptative approaches

The area of Reinforcement Learning (RL) involves an agent interacting with an environment, where the agent takes actions resulting in immediate rewards. The objective is for the agent to learn a policy that maximizes cumulative future rewards. The discount factor determines the importance of future rewards. RL algorithms iteratively update the policy to enhance the value function [97].

Although RL might seem distant from unsupervised learning approaches for compression, one approach proposed in this thesis draws inspiration from RL's general setup. At some point, the neural network training process will be viewed as a temporal evolution, and an adaptive training strategy will be adopted, continuously adjusting hyperparameters to achieve training goals. Known RL techniques will not be directly adopted, only the "environment adaptation" inspiration, related to the automatic change of loss hyperparameters, which are usually constant values under optimization.

Zhe et al. [98] studied an idea of loss hyperparameters adaptation. This paper proposes an enhanced training method for L2-normalized softmax in CNNs. L2-normalized softmax has shown promise in improving CNN accuracy, especially in tasks like face recognition and person re-identification. However, existing methods add extra parameters as class centers without addressing how to learn them, limiting further improvement. The paper addresses this issue by treating CNN training as a time series and introducing a learning algorithm that combines gradient descent with exponential moving averages (EMA) to update class centers. This approach illustrates adaptive training since softmax is usually a fixed operation, not a mutable optimization target.

Zhang et al. [99] examine face recognition tasks using loss functions with angular margin, which have been successful for this task. However, the authors tackle the instability problem caused by the sensitivity of the hyper-parameters setting. To handle this, the authors propose an adaptive parameter softmax loss with different scale parameters for target and non-target logits, while dynamically adapting margin parameters. The sensitivity of training to loss hyper-parameters will also be addressed in this thesis.

Adaptive training ideas are also applied in semi-supervised learning to reduce dependence on fully labeled datasets. In some scenarios, non-labeled targets are updated through temporal process analysis, a form of meta-learning [100]. This approach views neural network training as a temporal adaptive process. Semi-supervision can also be applied adaptively through multiple models, for example, using a soft teacher to label the unlabeled data, aiding in training the main neural network [101]. While these approaches are not directly related to the ideas developed in this thesis, they resemble the core concepts explored in later chapters.

3.4 Other Approaches

Generative adversarial models have also been applied in image coding [102]. Although it was one of the first proposed works, it represented the best results in the field for a long time. The authors attempted to simulate the behavior of a wavelet transform with a neural structure that uses a pyramidal decomposition, as shown in Figure 3.9. Another interesting aspect of this work is the adversarial training applied to the autoencoder to produce more realistic reconstructions. The generator is an autoencoder inspired by the wavelet transform, while the discriminator is a standard neural network.

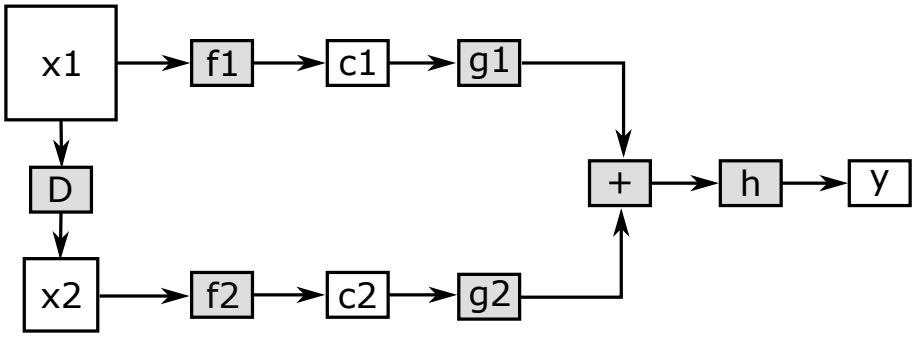


Figure 3.9: The pyramidal approach where the inputs are obtained by a subsampling operation D, which generates the representations x_1 and x_2. These inputs are processed by the functions f that generate different coefficients. The functions g align the size of these coefficients, which are summed and processed one last time by the function h to generate the latent y [102].

A different approach can be seen in Akbari's work [103], which involves several computer vision concepts. The authors utilize the classic encoders FLIF (Free Lossless Image Format [104]) and BPG (Better Portable Graphics, based on a subset of the HEVC - High Efficient Video Coding - open video compression standard [105]). Using segmentation maps, they propose a reconstruction network that receives a subsampled image version. The network then attempts to reconstruct the image. The residual difference is compressed with BPG. The subsampled version and the segmentation map are losslessly compressed using FLIF. Their results surpass BPG, except at high rates.

Although the focus of this work is not lossless neural compression, models from this line can also contribute ideas to the field of lossy compression. An example of this idea can be seen in Mentzer's work [106], which presents hierarchical modeling. A sequence of feature extractors is applied hierarchically to the non-quantized outputs of previous extractors. In this manner, the i-th level extractor uses the output of the $(i-1)$-th level extractor.

The generated maps are quantized and passed to autoregressive predictors, which output parameters to characterize the image distribution at multiple scales. These predictors

are autoregressive because the i-th level predictor requires the output of the $(i+1)$-th level predictor as well as the quantized feature map of its level. The flowchart of this architecture is shown in Figure 3.10. The authors report better results compared to PNG, WebP, and JPEG2000.

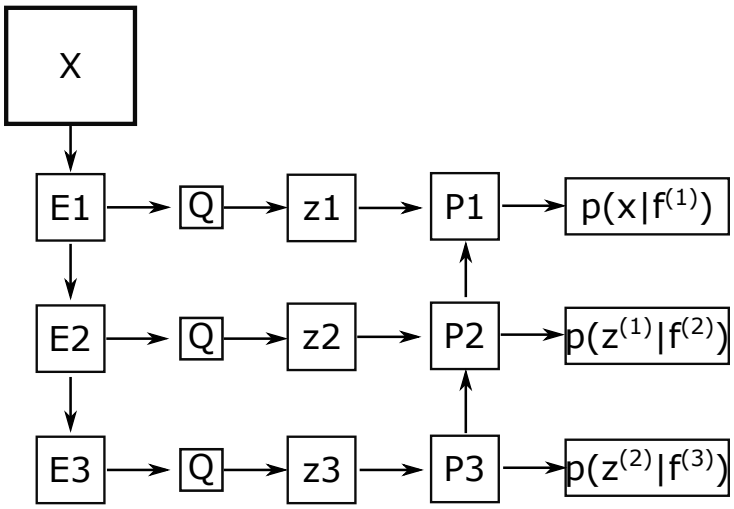


Figure 3.10: The approach uses a set of feature extractors E that establish a hierarchy. The predictors P are fed both the quantized maps and the output of the lower-level predictor. The symbol Q represents quantization, and the values z are the quantized features [106].

Another work following this approach can be seen in Mentzer's paper [107]. The authors compress an image using BPG to obtain the residual. They use a neural model that learns the distribution of the residual conditioned on the reconstruction obtained with BPG. This distribution is combined with entropy coding to compress the residual. They report better results compared to PNG, WebP, and JPEG2000.

3.5 End-to-end Neural Approaches versus Classical CODECs

Initial attempts at end-to-end neural compression using common autoencoders struggled to match or surpass JPEG in terms of SSIM, MS-SSIM, and PSNR metrics [15, 16, 17, 18]. Later approaches, incorporating LSTM-based layers and Generative Adversarial Networks, achieved MS-SSIM results comparable to BPG [20, 102]. However, these autoencoder-based strategies faced challenges in consistently outperforming the best classical compression methods, even when incorporating auxiliary methods like FLIF, BPG [103], relevance maps [39], or similar techniques.

The breakthrough that enabled neural methods to surpass classical techniques came with the advent of variational autoencoder-based approaches [45, 23]. These methods began to outperform JPEG-2000 and BPG in specific metrics and scenarios. Adopting the hyperprior mechanism [23] became prevalent in subsequent approaches, consistently surpassing BPG and even VTM intra-encoders. Minnen's work [108] notably was the first to consistently outperform BPG across all metrics and scenarios. Other works [30] showed comparable results to BPG, and some strategies [43, 37] began surpassing VTM results in select scenarios.

It is crucial to emphasize that this thesis primarily focuses on target bitrate control of neural networks, specifically related to variable rate models (detailed in Section 3.3.3), quantization approximations (explored in Section 3.3.9), and other target-based approaches (discussed in Section 3.3.8). The objective of these approaches is not always to surpass existing neural methods but sometimes to introduce additional features to the neural compression paradigm or enhance the understanding of these models.

Despite the significant advancements mentioned above, recent literature has presented many results that surpass these achievements, including those that have already outperformed some of the best classical CODECs. Notable examples of these developments include [86, 76, 93, 78, 89, 77, 34].

3.6 Conclusions

This chapter reviewed significant works in neural image coding, covering non-recurrent neural networks, recurrent neural networks, variational autoencoders (VAEs), and various methodologies within each category. It detailed approaches focusing on variable rate models, different neural operations, bit allocation and attention modules, post-processing techniques, generative compression, parameter-adaptive strategies, and quantization approximation issues.

The evolution of neural image coding techniques highlights the growing potential of neural networks for efficient and effective image compression. The field has progressed remarkably from early struggles to match classical codecs to recent advancements that surpass them in many scenarios. Techniques like variational autoencoders and recurrent neural networks have shown promise in optimizing the rate-distortion trade-off while preserving image content.

One main characteristic of this review is the almost universal use of variational models derived from the original proposals of Ballé. Both parametric and non-parametric approaches have been seminal for neural image compression. Despite various initiatives to obtain greater flexibility with these models, such as incorporating the possibility of

compression with multiple rate-distortion points, the problem of rate control—where a model achieves a specific rate for all images it compresses—remains underexplored and poorly understood.

Assimilating the concepts highlighted in this literature review is crucial for the subsequent chapters, which will delve deeper into specific problems and techniques. By narrowing the focus and detailing these approaches, the objective is to present proposals that build upon the foundations established in this chapter. In summary, the knowledge gleaned from this review sets the stage for the detailed exploration of neural image compression techniques and problem-solving strategies in the following chapters.

Chapter 4

Problem Statement

This chapter serves as a foundation for understanding the rationale and methodologies employed in the subsequent approaches proposed in this thesis. It explores the fundamental concepts of rate control in compression and autoencoder modeling, focusing on variational autoencoder (VAE) paradigms and variational Bayesian inference.

Section 4.1 delves into the intricate world of rate control in compression, grounded in rate-distortion theory. This framework underpins the optimization modeling for effective compression with controlled rates. Following this, Section 4.1.1 draws parallels between rate-distortion optimization and seminal architectures, particularly those incorporating variational Bayesian inference in neural network structures. These insights lay the groundwork for understanding the architecture details presented in Section 4.2 and its subsections, where two seminal compression architectures and their entropy modeling intricacies are dissected.

A thorough understanding of the concepts outlined here requires prior knowledge from Section 2.3 and Appendix I, which will be extensively referenced throughout the thesis. It is strongly recommended that the reader fully understands the concepts presented in those sections.

4.1 Rate Control in Coding

Efficient use of communication channel bandwidth is crucial in multimedia transmission. Both rate control and bit allocation are needed to manage the encoding rate to meet channel bandwidth constraints or memory requirements. JPEG-1, a popular image standard, lacked rate control and only offered a gradation of compression quality levels [8]. Its successor, JPEG-2000, introduced rate control mechanisms [9]. The efficiency of this rate control in JPEG-2000 has been studied [109]. Similar studies exist for classic video CODECs [110, 111].

Rate control allows designers to create feasible schemes for specific applications more flexibly. While trying to achieve a target rate, the control mechanism must consider challenging issues like distortion [111]. According to rate-distortion theory, distortion D decreases as rate R increases. Thus, the fundamental problem in rate control can be formulated as follows [111]:

$$\begin{aligned} &\arg\min D \\ &\text{such that } R \leq R_{max} \end{aligned} \tag{4.1}$$

where R_{max} is the maximum allowed rate. The goal is to achieve the highest quality within the R_{max} limit. Encoders adopt a Lagrangian rate-distortion cost function:

$$J = D + \lambda \cdot R \tag{4.2}$$

where D is a measure of distortion, R is the rate, and λ is the Lagrange multiplier. The rate-distortion space is characterized by the convex hull, shown in Figure 4.1. The convex hull represents the best possible distortion for a given rate. Typically, points on the convex hull cannot be achieved, so different encoders may reach different points when compressing images at specific rates.

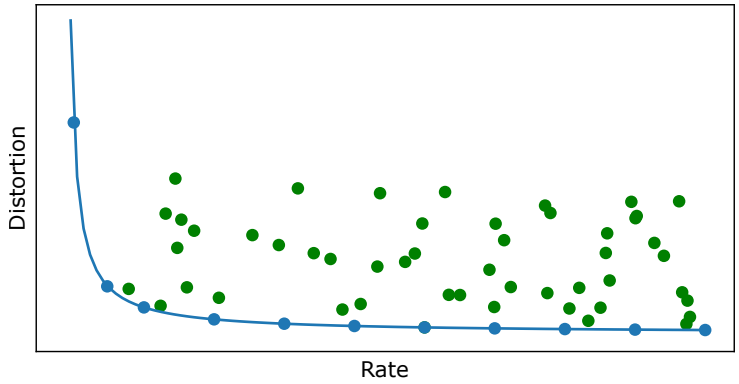


Figure 4.1: The rate-distortion space characterized by the convex hull is denoted by blue points, representing the optimal distortion values for a given rate. The green points represent the performance of encoders, as it is not always possible to reach an optimal point.

Theoretically, a value for λ can be optimized to encode an image at a target rate optimally. However, λ is usually determined by other parameters, such as the quantization step, to avoid the need for multiple iterations. Therefore, rate control tools are commonly used when rate-distortion optimality cannot be guaranteed, yet the rate constraint is met.

4.1.1 Behavior of the Rate in Variational Approaches

Although there are several approaches for rate control in classic CODECs, this remains an underexplored problem in the context of neural coding, which motivates the discussed approach. The rate control issue arises from the strategy of modifying λ, which cannot be applied to variational neural models after training.

ELBO is the function typically used for optimization in Bayesian inference, as detailed in Section 2.3.4. Its adoption is due to the KL divergence, denoted by $\mathcal{KL}$, which is not directly tractable because it depends on the true *a posteriori* [112]. A VAE [22] is a neural network approximating the Bayesian inference *a posteriori*. The connection with the encoding with VAEs is established in [45], leading to the following equation:

$$J = \lambda \cdot D + R \tag{4.3}$$

Further details about this equation will be provided in Section 4.2. Notably, the above equation is equivalent to Equation 4.2, but in variational inference derivation, λ must specifically accompany the distortion term D.

Using this approach, the optimization algorithm finds parameters that minimize the weighted sum of distortion and rate. Evaluating the distortion D involves assessing the original and reconstructed image using a distance measure. The rate R is estimated using the entropy of the model's latent representation. Thus, by using different values for λ, various models operating at different points on the convex hull can be obtained, as shown in Figure 4.1.

For instance, when a neural network is trained using a function with specific parameters, such as $J = \lambda_0 D + R$, its weights align to perform the task considering the trade-off defined by λ_0. Rate-distortion minimization means that during training, the model learns to extract and refine features to optimize the compression task under these conditions. Using λ_0 during training results in a neural model whose weights are tied to this hyperparameter: $\lambda_0 \implies \theta_{\lambda_0}$. This weight vector, also known as the model parameter vector, parameterizes the neural network.

Since the parameter vector θ_{λ_0} is only modified during training, choosing a λ_1, where $\lambda_0 \neq \lambda_1$, would not significantly alter the established trade-off during inference. To achieve a reconstruction with a trade-off defined by λ_1, new training with $\lambda_1 \implies \theta_{\lambda_1}$ is necessary, and $\theta_{\lambda_0} \neq \theta_{\lambda_1}$. This implies that when using the loss function from variational Bayesian inference [22, 45], each neural model is tied to its respective λ_i. Therefore, each model is represented by the pair $(\lambda_i, \theta_{\lambda_i}), i \in \mathbb{N}$. This approach is used in most proposals, following reference architectures in the literature [45, 23].

It is worth noting that there are multiple-rate architectures, as mentioned in Chapter

3, specifically in Section 3.3.3. These approaches do not aim to achieve specific rates for image compression. Using the adopted notation, these models are represented by the pair $(\lambda, \theta_\lambda)$, where $\lambda = \{\lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_i\}, i \in \mathbb{N}$. In other words, these models generalize the models proposed by Ballé's works [45, 23] using different strategies and architectures.

However, there is a problem with these rate-distortion Lagrangian-based approaches. For a trained model $(\lambda_i, \theta_{\lambda_i}), i \in \mathbb{N}$, the operating point on the convex hull depends on the image being compressed. Thus, if a specific rate r_i is desired, there is no direct link with λ_i: $\lambda_i \not\Longrightarrow r_i$.

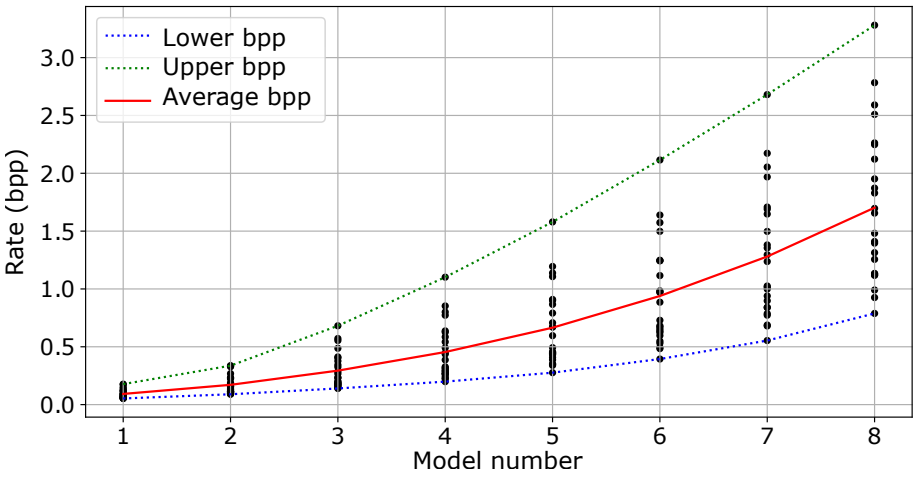


Figure 4.2: Results of models available on GitHub [113] by the authors of the VAE proposals for compression [45, 23]. The points represent the rate for each image from the Kodak database for each available model. The models were not trained to achieve specific rates, showing that obtaining specific rates with the standard approach is not feasible.

This fact is illustrated in Figure 4.2, which shows the results of 8 pre-trained models [113] with increasing λ, made available by the authors [45] on the Kodak database. These models, not trained for specific rates, produce reconstructions at a wide range of rates depending on the image. Each model shows significant variance in both rate and quality. For example, Model 1 has a bpp (bits per pixel) ranging from 0.053 to 0.177. Model 5 has rates ranging from 0.276 to 1.579.

From Figure 4.2, we can also infer the behavior of multiple-rate models, denoted by $(\lambda, \theta_\lambda)$, where $\lambda = \{\lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_i\}, i \in \mathbb{N}$. If a model is trained with multiple values of λ using rate distortion, similar high-variance behavior in the encoding rate will be observed.

In standard rate-distortion approaches, to achieve a specific rate, one could encode an image with multiple neural models and select the representation with the closest rate. By ordering models by λ, a binary search could be used to avoid testing all models. However, this method has several flaws, as many models would be necessary to have a high probability of achieving the desired rate. Moreover, even if the rate is achieved,

both the encoder and decoder would need to store a large number of trained models, taking up significant space and making the CODEC unfeasible for specific applications. Considering these aspects of standard rate-distortion optimization, the proposal is to train rate-oriented models. Section 4.2 will detail the VAEs used as references and their features.

4.2 Baseline Architectures

This section presents details concerning the architectures used as a reference. The central focus is the modeling adopted in autoencoders for compression [45] and their equivalence with the variational optimization in VAEs [22]. This formalization provides insights regarding the architecture and the impact of modifications.

To make the reading more concise, the section revisits the ideas of generalized divisive normalization and the formulation of VAEs based on variational Bayesian inference. These ideas are closely linked to the modeling of the architectures, which will be detailed. It is highly advised that the reader review Section 2.3.4. A deep understanding of the baseline architectures is essential to grasp the analyses and proposals of this thesis.

4.2.1 Activation Function of the architectures

A necessary component to describe the reference architectures is the GDN (generalized divisive normalization) [21], detailed in Section 2.2.7. The authors optimized transforms to achieve specific statistical properties in the transformed space. When adjusted as cascaded operations, each stage produces marginal "directions" that are less similar to Gaussians and then Gaussianize these directions using non-parametric non-linear scalar transforms [21].

The generalized divisive normalization, a general approach of the divisive normalization, was proposed by Ballé [21] and shown to work well as an activation function on the baseline architectures [45, 23]. The authors demonstrated that the transformation adeptly "Gaussianizes" the data by optimizing parameters to minimize the KL divergence of the distribution of transformed data. They also proposed an ablation study comparing the impact of the GDN in a separate work [114]. These non-linearities are distinct features of the baseline architectures.

4.2.2 Optimization in VAEs

VAEs are probabilistic models that perform variational inference [72]. Uncertainty is specified through conditional probability distributions. The goal is not to model $p_{\mathbf{x}}(\mathbf{x};\theta)$ but

to model $p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\theta)$, parameterized by θ, that approximates the true distribution. Here, the latent variables $\mathbf{y}$ are conditioned on the observed variables $\mathbf{x}$ [72]. The maximum likelihood optimization in these models can be enhanced from a Bayesian standpoint using Bayes' Theorem, introduced in Section 2.3.1, through MAP (maximum a posteriori) [72]:

$$p_{\mathbf{x}}(\mathbf{x};\theta) = \int p_{\mathbf{x},\mathbf{y}}(\mathbf{x},\mathbf{y};\theta)d\mathbf{y} = \int p_{\mathbf{y}}(\mathbf{y};\theta)\cdot p_{\mathbf{x}|\mathbf{y}}(\mathbf{x}|\mathbf{y};\theta)d\mathbf{y} \tag{4.4}$$

where the model is derived from the conditional probability rule.

Neural networks are advantageous due to their expressiveness in representing distributions. Even if the conditional distribution $p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\theta)$ is simple, the marginal distribution $p_{\mathbf{x}}(\mathbf{x};\theta)$ can be complex, containing arbitrary dependencies between variables [72]. However, even in deep models, the intractability problem related to calculating $p_{\mathbf{x}}(\mathbf{x};\theta)$, presented in Section 2.3.3, still exists. The workaround is to introduce a parametric model $q_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\phi)$, termed an encoder [22]:

$$q_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\phi) \approx p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\theta) \tag{4.5}$$

The optimization is carried out via SGD. In this case, the encoder serves as a stochastic mapping between the observed variable $\mathbf{x}$ and the latent space $\mathbf{y}$, with its distribution representing the prior of the generative model, denoted by $p_{\mathbf{y}}(\mathbf{y};\theta)$. The decoder $p_{\mathbf{x}|\mathbf{y}}(\mathbf{x}|\mathbf{y};\theta)$ is the stochastic mapping from the latent space $\mathbf{y}$ to a region in the observed space.

The objective function of VAEs is based on the ELBO (Equation 2.45). Optimizing the ELBO is equivalent to optimizing the KL divergence, denoted by $\mathcal{KL}$. The relationship between the ELBO and the KL divergence can be derived as follows:

$$\begin{aligned}
\log p_{\mathbf{x}}(\mathbf{x};\theta) &= \mathbb{E}_{q_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\phi)}[\log p_{\mathbf{x}}(\mathbf{x};\theta)] \\
&= \mathbb{E}_{q_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\phi)}\left[\log\left[\frac{p_{\mathbf{x},\mathbf{y}}(\mathbf{x},\mathbf{y};\theta)}{p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\theta)}\right]\right] \\
&= \mathbb{E}_{q_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\phi)}\left[\log\left[\frac{p_{\mathbf{x},\mathbf{y}}(\mathbf{x},\mathbf{y};\theta)}{q_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\phi)}\cdot\frac{q_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\phi)}{p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\theta)}\right]\right] \\
&= \mathbb{E}_{q_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\phi)}\left[\log\left[\frac{p_{\mathbf{x},\mathbf{y}}(\mathbf{x},\mathbf{y};\theta)}{q_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\phi)}\right]\right] + \mathbb{E}_{q_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\phi)}\left[\log\left[\frac{q_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\phi)}{p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\theta)}\right]\right] \\
\log p_{\mathbf{x}}(\mathbf{x};\theta) &= \mathcal{L}_{\theta,\phi}(\mathbf{x}) + \mathcal{KL}(q_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\phi)||p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\theta))
\end{aligned} \tag{4.6}$$

From the above equation, it is evident that maximizing $\mathcal{L}_{\theta,\phi}(\mathbf{x})$ for parameters θ and ϕ optimizes two major aspects. First, it maximizes the marginal likelihood $p_{\mathbf{x}}(\mathbf{x};\theta)$, improving the generative model. Additionally, it minimizes the KL divergence of the distribution $q_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\phi)$ relative to the true posterior $p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\theta)$, making $q_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\phi)$ a

better approximation. Equation 4.6 is crucial for understanding the link to rate-distortion optimization in subsequent sections, which is why it is derived in detail here.

4.2.3 Non-Parametric Architecture

The architecture presented in this section was the first to consider the trade-off of rate-distortion in its optimization [21], as discussed in Chapter 3. It has an equivalence with the modeling of VAEs (detailed in Section 2.3.5). This architecture is optimized for the objective, using non-linear transforms, as depicted in Figure 4.3.

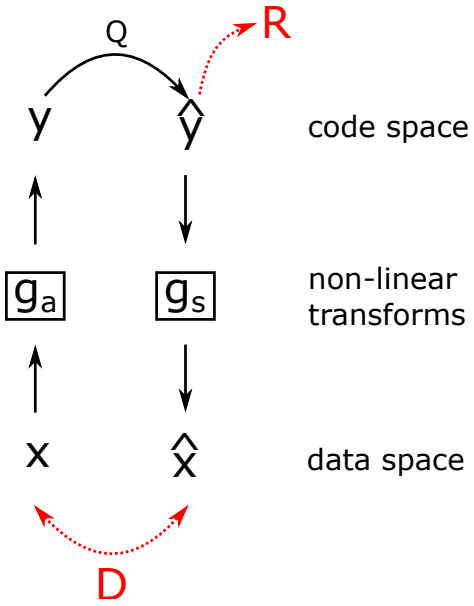


Figure 4.3: An image vector $\mathbf{x} \in \mathbb{R}^n$ is mapped by a transformation called analysis, $\mathbf{y} = g_a(\mathbf{x}; \phi); \mathbf{y} \in \mathbb{R}^m$. This representation is quantized (function Q), generating $\hat{\mathbf{y}}$, from which the rate of this discrete code is calculated (based on Shannon entropy, $R = \mathcal{H}(p_{\hat{\mathbf{y}}})$). For reconstruction, the elements of $\hat{\mathbf{y}}$ are interpreted as continuous vectors, returning to the original domain through synthesis, $\hat{\mathbf{x}} = g_s(\hat{\mathbf{y}}; \theta); \hat{\mathbf{x}} \in \mathbb{R}^n, \hat{\mathbf{y}} \in \mathbb{R}^m$. From the pair $(\mathbf{x}, \hat{\mathbf{x}})$ a distortion measure D is calculated. The parameter vectors ϕ and θ are optimized via a weighted sum of rate and distortion, $R + \lambda \cdot D$.

Most compression methods use orthogonal linear transforms to reduce data correlations and simplify entropy coding [45]. However, the joint statistics of linear filter responses exhibit strong higher-order dependencies, which can be significantly reduced using non-linear gain control operations, such as GDN. The cascade of these operations is more efficient in Gaussianizing the joint statistics of natural image data than cascades of linear transforms followed by point non-linearities [21]. This efficiency is why GDN is adopted as a non-linearity in this architecture.

The analysis transform g_a consists of three stages: convolution, sub-sampling, and generalized divisive normalization. It uses standard convolution operations with GDN

Table 4.1: The analysis transform and synthesis transform are set up as detailed below. In the adopted notation, C-9 $\times$ 9, $\downarrow$2, GDN, 256 signifies a convolution layer with 256 filters, featuring a spatial support of 9 $\times$ 9, a subsampling factor of 2, and utilizing GDN as the activation function. Similarly, TC-5 $\times$ 5, $\uparrow$2, IGDN, 256 refers to a transposed convolution with 256 filters, a supersampling factor of 2, and IGDN as the activation function. When no activation function is shown, it implies that the layer has no non-linear activation.

Analysis	**Synthesis**
C-9 $\times$ 9, $\downarrow$2, GDN, 256	TC-5 $\times$ 5, $\uparrow$2, IGDN, 256
C-5 $\times$ 5, $\downarrow$2, GDN, 256	TC-5 $\times$ 5, $\uparrow$2, IGDN, 256
C-5 $\times$ 5, $\downarrow$2, 256	TC-9 $\times$ 9, $\uparrow$2, 256

as the non-linear activation function, defined in Equation 2.25. The fixed vectors for the GDN are $\alpha_{ij} = 2$ and $\epsilon_i = \frac{1}{2}$, $\forall i, j$. As the operation combined with the GDN is a convolution, the matrix $\mathbf{T}$ in these equations represents the result of the convolution with sub-sampling.

In the synthesis transform g_s, three stages of convolution, up-sampling, and generalized divisive normalization are also performed. Here, the GDN is used in its inverse form, IGDN, obtained through fixed-point iteration as presented in Equation 2.27. The fixed vectors are $\alpha_{ij} = 2$ and $\epsilon_i = \frac{1}{2}$, $\forall i, j$. In both analysis and synthesis, the parameters trained by SGD are β_i and γ_{ij} for GDN, and $\hat{\beta}_i$ and $\hat{\gamma}_{ij}$ for IGDN. The neural network structure is shown in Table 4.1.

The goal is to minimize the weighted sum of rate and distortion, $R + \lambda D$, over the parameters of the analysis and synthesis transforms and the code entropy. The actual rates achieved by a properly designated entropy encoder are slightly higher than the entropy. The objective can be formulated as [45]:

$$J(\mathbf{x}, \hat{\mathbf{x}}) = -\mathbb{E}_{p_{\hat{\mathbf{y}}}}[\log_2 p_{\hat{\mathbf{y}}}] + \lambda \mathbb{E}[d(\mathbf{x}, \hat{\mathbf{x}})] \tag{4.7}$$

where d represents the distortion function, with expectations approximated by batches of the training data. The quantized values are obtained through a rounding function $\hat{y}_i = \text{round}(y_i)$, where i indexes all vector elements, including spatial positions and channels. The marginal density of $\hat{y}_i$ is represented by a PMF defined as [45]:

$$p_{\hat{\mathbf{y}}}(n) = \int_{n-\frac{1}{2}}^{n+\frac{1}{2}} p_{y_i}(t)dt \tag{4.8}$$

with $n \in \mathbb{Z}$. Since the elements of Equation 4.7 depend on quantized values, and quantization is not a differentiable operation, the authors substitute the operation with uniform additive noise of unit width, $\Delta\mathbf{y}$. The density $\tilde{\mathbf{y}} = \mathbf{y} + \Delta\mathbf{y}$ is a continuous approximation

of the PMF of $\hat{\mathbf{y}}$:

$$p_{\tilde{\mathbf{y}}}(n) = p_{\hat{\mathbf{y}}}(n) \tag{4.9}$$

The relation implies that the differential entropy of $\tilde{\mathbf{y}}$ can approximate the discrete entropy of $\hat{\mathbf{y}}$. Uniform noise approximates the quantization error and is often used as a quantization error model [45, 115]. With this continuous approximation for the distribution of quantized coefficients, the loss function is redefined as:

$$J(\phi, \theta) = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[-\sum_i \log_2 p_{\tilde{y}_i}(g_a(\mathbf{x};\phi) + \Delta\mathbf{y}; \psi^{(i)}) + \lambda d(g_s(g_a(\mathbf{x};\phi) + \Delta\mathbf{y};\theta), \tilde{\mathbf{x}})\right] \tag{4.10}$$

where batches of training data approximate the expectation, and $\psi^{(i)}$ represents the parameter vector for the distribution $p_{\tilde{y}_i}$, which will be detailed later.

Rate-Distortion Optimization via Variational Inference

Although the model optimization is derived through rate-distortion optimization, with the continuous relaxation of the quantization operation, these models have a strict relationship with variational models [22] detailed in Section 4.2.2. In this modeling, the goal is to approximate the true posterior $p_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\theta)$, parameterized by θ, with a density $q_{\mathbf{y}|\mathbf{x}}(\mathbf{y}|\mathbf{x};\phi)$, parameterized by ϕ.

To understand the equivalence between the architectures, consider that the generative model is defined according to the following equations [45]:

$$p_{\mathbf{x}|\tilde{\mathbf{y}}}(\mathbf{x}|\tilde{\mathbf{y}};\lambda,\theta) = \mathcal{N}\left(g_s(\tilde{\mathbf{y}};\theta), (2\lambda)^{-1}\mathbf{1}\right)(\mathbf{x}) \tag{4.11}$$

$$p_{\tilde{\mathbf{y}}|\psi}(\tilde{\mathbf{y}}|\psi) = \prod_i \left(p_{y_i|\psi^{(i)}}\left(\psi^{(i)}\right) * \mathcal{U}\left(-\frac{1}{2}, \frac{1}{2}\right)\right)(\tilde{y}_i) \tag{4.12}$$

where $\mathcal{U}(-\frac{1}{2}, \frac{1}{2})$ is uniform noise in the interval $\left[-\frac{1}{2}, \frac{1}{2}\right]$. The term $\mathcal{N}$ is a Gaussian evaluated for $\mathbf{x}$, with $\mu = g_s(\tilde{\mathbf{y}};\theta)$ and $\sigma^2 = (2\lambda)^{-1}\mathbf{1}$. The entropy model of the prior has parameters ψ, which are used to approximate the distribution but do not have a statistical interpretation like the mean and variance of the Gaussian. Finally, the inference model approximates the posterior by adding a uniform density of unit width around each point y_i [45]:

$$q_{\tilde{\mathbf{y}}|\mathbf{x}}(\tilde{\mathbf{y}}|\mathbf{x};\phi) = \prod_i \mathcal{U}\left(y_i - \frac{1}{2}, y_i + \frac{1}{2}\right)(\tilde{y}_i) \tag{4.13}$$

with $\mathbf{y} = g_a(\mathbf{x};\phi)$. This variational interpretation for the rate-distortion modeling allows the derivation of the proposed objective function in terms of the KL divergence between

the inference model and the true posterior:

$$\begin{aligned}
\mathcal{KL}(q_{\tilde{\mathbf{y}}|\mathbf{x}} \| p_{\tilde{\mathbf{y}}|\mathbf{x}}) &= \mathbb{E}_{\tilde{\mathbf{y}}\sim q} \ln q_{\tilde{\mathbf{y}}|\mathbf{x}}(\tilde{\mathbf{y}}|\mathbf{x};\phi) - \mathbb{E}_{\tilde{\mathbf{y}}\sim q} \ln p_{\tilde{\mathbf{y}}|\mathbf{x}}(\tilde{\mathbf{y}}|\mathbf{x}) \\
&= \mathbb{E}_{\tilde{\mathbf{y}}\sim q} \ln q_{\tilde{\mathbf{y}}|\mathbf{x}}(\tilde{\mathbf{y}}|\mathbf{x};\phi) - \mathbb{E}_{\tilde{\mathbf{y}}\sim q} \ln p_{\mathbf{x}|\tilde{\mathbf{y}}}(\mathbf{x}|\tilde{\mathbf{y}}) - \mathbb{E}_{\tilde{\mathbf{y}}\sim q} \ln p_{\tilde{\mathbf{y}}}(\tilde{\mathbf{y}}) + \mathbb{E}_{\tilde{\mathbf{y}}\sim q} \ln p_{\mathbf{x}}(\mathbf{x}) \\
&= -\mathbb{E}_{\tilde{\mathbf{y}}\sim q} \ln \left(\frac{1}{(2\pi)^{n/2}|(2\lambda)^{-1}\mathbb{I}|^{1/2}} e^{\left(-\frac{1}{2}(\mathbf{x}-\tilde{\boldsymbol{x}})^\top ((2\lambda)^{-1}\mathbb{I})^{-1}(\mathbf{x}-\tilde{\boldsymbol{x}})\right)} \right) \\
&\quad - \mathbb{E}_{\tilde{\mathbf{y}}\sim q} \ln p_{\tilde{\mathbf{y}}}(\tilde{\mathbf{y}}) + const \\
&= -\mathbb{E}_{\tilde{\mathbf{y}}\sim q} \ln \left(\frac{1}{(2\pi)^{n/2}(2\lambda)^{-n/2}} e^{-\lambda\|\mathbf{x}-\tilde{\boldsymbol{x}}\|^2} \right) - \mathbb{E}_{\tilde{\mathbf{y}}\sim q} \ln p_{\tilde{\mathbf{y}}}(\tilde{\mathbf{y}}) + const \\
&= \lambda \mathbb{E}_{\tilde{\mathbf{y}}\sim q} \left(\|\mathbf{x}-\tilde{\boldsymbol{x}}\|^2 \right) - \mathbb{E}_{\tilde{\mathbf{y}}\sim q} \ln p_{\tilde{\mathbf{y}}}(\tilde{\mathbf{y}}) + const
\end{aligned} \tag{4.14}$$

where $\tilde{\mathbf{x}} = g_s(\tilde{\mathbf{y}};\theta)$, $\mathbb{I}$ is the identity matrix, $|\cdot|$ is the determinant operator and $\|\cdot\|^2$ is the Euclidean norm operator. The second line of the equation is obtained by applying Bayes' Theorem. Since the distribution $q_{\tilde{\mathbf{y}}|\mathbf{x}}$ is represented as uniform noise of unit width, its entropy is zero. The data distribution $p_{\mathbf{x}}$ is not optimized and remains constant. Furthermore, the entropy is given in nat (natural unit of information). Since the unit only affects the scale of the results, one can consider the entropy in bits for optimization without affecting the outcome. Thus, minimizing the KL divergence between the variational approximation and the true posterior is equivalent to optimizing a rate-distortion.

The Non-Parametric Entropy Model

A significant point regarding the architecture proposed by Ballé [45] is the approximation of the distribution in Equation 4.12. To make the approximation applicable, the approach uses cumulative distribution functions. A density $p : \mathbb{R} \implies \mathbb{R}^+$ is defined through its cumulative distribution function $f : \mathbb{R} \implies [0,1]$. If the cumulative is a composition of functions, the density can be obtained through the chain rule for derivatives:

$$c = f_K \circ f_{K-1} \circ f_{K-2} \cdots f_1 \tag{4.15}$$

$$p = f'_K \cdot f'_{K-1} \cdot f'_{K-2} \cdots f'_1 \tag{4.16}$$

where the derivative of f_k is represented by f'_k. Considering f_k as vector functions, $f_k : \mathbb{R}^{d_k} \implies \mathbb{R}^{r_k}$, the derivatives f'_k are Jacobian matrices. Since these are univariate distributions, we assume $d_1 = r_K = 1$. For $p(x)$ to be a density, f_K must map to the interval $[0,1]$ and $p(x) \geq 0$. This can be ensured by requiring the elements of the Jacobian

to be non-negative. An effective choice for f_k can be given by the following equations [23]:

$$f_k(\mathbf{x}) = g_k(\mathbf{H}^{(k)}\mathbf{x} + \mathbf{b}^{(k)}) \tag{4.17}$$

$$f_K(\mathbf{x}) = \eta(\mathbf{H}^{(K)}\mathbf{x} + \mathbf{b}^{(K)}) \tag{4.18}$$

$$g_k(\mathbf{x}) = \mathbf{x} + \mathbf{a}^{(k)} \odot \tanh \mathbf{x} \tag{4.19}$$

$1 \leq k < K$, where $\mathbf{H}^{(k)}$ represent matrices, $\mathbf{b}^{(k)}$ represent vectors, η is the sigmoid function, and tanh is the hyperbolic tangent function. In this case, $\odot$ denotes element-wise multiplication. The idea behind the last non-linearity is that it allows the space to be contracted or dilated near $x = 0$ through the parameter $\mathbf{a}^{(k)}$ [23].

For the derivatives to be non-negative, it is necessary that $\mathbf{H}^{(k)}$ only have non-negative elements and that the elements of $\mathbf{a}^{(k)}$ have -1 as their lower limit. This property is achieved with a reparameterization [23]:

$$\mathbf{H}^{(k)} = softplus(\hat{\mathbf{H}}^{(k)}) \tag{4.20}$$

$$\mathbf{a}^{(k)} = \tanh(\hat{\mathbf{a}}^{(k)}) \tag{4.21}$$

where the hat variables represent the actual parameters. These elements encompass all the significant points in modeling the non-parametric architecture. Further details can be found in the reference works [45, 23], which inspired the mathematical notations adopted here.

4.2.4 Parametric Architecture

The parametric architecture [23, 24] extends the architecture presented in the previous section [45]. Using a simple factorized distribution is a substantial oversimplification. The quantized latent $\hat{\mathbf{y}}$ tends to produce non-zero responses clustered in regions of high contrast, edges, and areas with textures. These clustered outputs imply a probabilistic coupling between the responses, not represented in models with a fully factorized *a priori* distribution [23].

The enhanced architecture uses side information, a strategy where additional information sent from the encoder to the decoder signals modifications to the entropy model, reducing misalignment. This method is feasible because the marginals for a particular image vary significantly from those for the overall set of images the model is designed to handle. The volume of side information sent is, on average, smaller than the reduction in code length achieved by better adjusting $p(\hat{\mathbf{y}})$ to the marginals of a particular image [23].

Using the formalism developed for VAEs, additional information can be seen as an *a priori* on the parameters of the entropy model. This new *a priori* becomes a hyper *a*

priori of the latent representation. The idea is to train it jointly with the network so that it captures the fact that spatially neighboring elements of the latent tend to have their scales vary in a coupled manner [23].

Introducing latent variables conditioned on another set of objective variables, assumed to be independent, is a standard way of modeling dependencies between these variables [116]. The idea is to introduce a set of random variables $\mathbf{z}$ to capture these spatial dependencies, as shown in Figure 4.4.

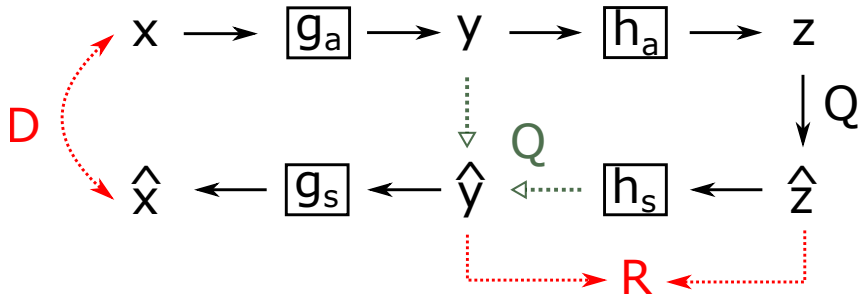


Figure 4.4: The architecture with the addition of the hyper *a priori*. In this case, h_a and h_s represent the additional non-linear transforms, which make up the hyper *a priori*.

The variable $\mathbf{z}$ follows the same approach as the previous work [23], adopting a fully factorized non-parametric model, given that there is no *a priori* distribution on this set of variables [23]:

$$p_{\tilde{\mathbf{z}}|\psi}\left(\tilde{\mathbf{z}}|\psi\right) = \prod_i \left(p_{z_i|\psi^{(i)}}(\psi^{(i)}) * \mathcal{U}\left(-\frac{1}{2}, \frac{1}{2}\right)\right)(\tilde{z}_i) \tag{4.22}$$

In this context, $\tilde{\mathbf{z}}$ is obtained during training through the continuous relaxation of the quantization operation via additive uniform noise. Again, $\psi^{(i)}$ represents the parameters trained to portray the density, as discussed in the previous section. The variable $\tilde{\mathbf{y}}$ is subsequently modeled as a zero-mean Gaussian with its standard deviation σ_i, where these values are predicted by applying a transform h_s to $\tilde{\mathbf{z}}$:

$$p_{\tilde{\mathbf{y}}|\tilde{\mathbf{z}}}(\tilde{\mathbf{y}}|\tilde{\mathbf{z}}; \theta_{\mathbf{h}}) = \prod_i \left(\mathcal{N}\left(0, \tilde{\sigma}_i^2\right) * \mathcal{U}\left(-\frac{1}{2}, \frac{1}{2}\right)\right)(\tilde{y}_i) \tag{4.23}$$

with $\tilde{\sigma}_i = h_s(\tilde{\mathbf{z}}; \theta_h)$ and θ_h being the set of parameters of the generative model for $\mathbf{z}$. The inference model is an extension of the model in Equation 4.13 [23]:

$$q_{\tilde{\mathbf{y}},\tilde{\mathbf{z}}|\mathbf{x}}(\tilde{\mathbf{y}}, \tilde{\mathbf{z}}|\mathbf{x}; \phi_{\mathbf{g}}, \phi_{\mathbf{h}}) = \left(\prod_i \mathcal{U}\left(y_i - \frac{1}{2}, y_i + \frac{1}{2}\right)(\tilde{y}_i)\right) \cdot \left(\prod_j \mathcal{U}\left(z_i - \frac{1}{2}, z_i + \frac{1}{2}\right)(\tilde{z}_i)\right) \tag{4.24}$$

with $\mathbf{y} = g_a(\mathbf{x}; \phi_g)$ and $\mathbf{z} = h_a(\mathbf{y}; \phi_h)$. The parameters ϕ_g and ϕ_h denote the parameters of the inference model with a priori hyperparameters. The loss function for this architecture,

following the variational interpretation, is given by [23]:

$$\mathcal{KL}(q_{\tilde{\mathbf{y}},\tilde{\mathbf{z}}|\mathbf{x}}||p_{\tilde{\mathbf{y}},\tilde{\mathbf{z}}|\mathbf{x}}) = \mathbb{E}_{\tilde{\mathbf{y}},\tilde{\mathbf{z}}\sim q} \ln q_{\tilde{\mathbf{y}},\tilde{\mathbf{z}}|\mathbf{x}}(\tilde{\mathbf{y}},\tilde{\mathbf{z}}|\mathbf{x};\phi_g;\phi_h) - \mathbb{E}_{\tilde{\mathbf{y}},\tilde{\mathbf{z}}\sim q} \ln p_{\tilde{\mathbf{y}},\tilde{\mathbf{z}}|\mathbf{x}}(\tilde{\mathbf{y}},\tilde{\mathbf{z}}|\mathbf{x}) \tag{4.25}$$

The first term, q, is a product of uniform densities with unit widths. Therefore, the entropy of this term is 0. Applying Bayes' theorem, the KL divergence can be expanded as:

$$\mathcal{KL}(q_{\tilde{\mathbf{y}},\tilde{\mathbf{z}}|\mathbf{x}}||p_{\tilde{\mathbf{y}},\tilde{\mathbf{z}}|\mathbf{x}}) = -\mathbb{E}_{\tilde{\mathbf{y}},\tilde{\mathbf{z}}\sim q} \ln p_{\mathbf{x}|\tilde{\mathbf{y}}}(\mathbf{x}|\tilde{\mathbf{y}}) - \mathbb{E}_{\tilde{\mathbf{y}},\tilde{\mathbf{z}}\sim q} \ln p_{\tilde{\mathbf{y}}|\tilde{\mathbf{z}}}(\tilde{\mathbf{y}}|\tilde{\mathbf{z}}) - \mathbb{E}_{\tilde{\mathbf{z}}\sim q} \ln p_{\tilde{\mathbf{z}}}(\tilde{\mathbf{z}}) + const \tag{4.26}$$

Where the first term (the likelihood) encapsulates the distortion and is tied to the decoder, the second and third terms represent the cross-entropy of encoding $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{z}}$, respectively. Analogous to traditional transform coding, the last term can be interpreted as side information.

The parametric architecture [23] is adopted in this chapter with a minor modification in the modeling depicted above. It involves a slightly different version [24], where the entropy model of $\tilde{\mathbf{y}}$ is parameterized by a Gaussian with arbitrary mean and variance, as shown in the equation below:

$$p_{\tilde{\mathbf{y}}|\tilde{\mathbf{z}}}(\tilde{\mathbf{y}}|\tilde{\mathbf{z}};\theta_{\mathbf{h}}) = \prod_i \left(\mathcal{N}\left(\tilde{\mu}_i, \tilde{\sigma}_i^2\right) * \mathcal{U}\left(-\frac{1}{2}, \frac{1}{2}\right)\right)(\tilde{y}_i) \tag{4.27}$$

with $[\tilde{\mu}_i, \tilde{\sigma}_i] = h_s(\tilde{\mathbf{z}};\theta_h)$. In this case, the prior hyper is also responsible for estimating the means for the Gaussians, which generalizes the model presented earlier [23]. However, this alteration does not impose other differences in the modeling shown in the previous equations. The specific values for the supports of the filters, the number of filters, can be seen in Table 4.2, which uses the same notation as Table 4.1.

The details presented in this section constitute the differences between parametric and non-parametric architecture. For further information, the reference works can be consulted [23, 24]. These same works also inspired the mathematical notations adopted here.

4.3 The General Idea of a Bitrate Control Approach

This chapter builds on the concepts proposed by Ballé [45, 23], where the direct optimization of the rate-distortion loss function is pursued. A notable feature of both approaches is the approximation of the quantization operation used during training. These approaches achieve this by adopting additive uniform noise, using the differential entropy of the noisy density function model during training. In contrast, the discrete entropy of the obtained

Table 4.2: Description of layers of the parametric baseline.

The analysis transforms, synthesis, hyper-analysis, and hyper-synthesis have settings established as per the values below, which follow the same notation adopted in Table 4.1.

Encoder	**Decoder**
C-5 × 5, ↓2, GDN, 256	TC-5 × 5, ↑2, IGDN, 256
C-5 × 5, ↓2, GDN, 256	TC-5 × 5, ↑2, IGDN, 256
C-5 × 5, ↓2, GDN, 256	TC-5 × 5, ↑2, IGDN, 256
C-5 × 5, ↓2, 256	TC-5 × 5, ↑2, 256
Hyper Encoder	**Hyper Decoder**
C-3 × 3, ReLU, 256	TC-5 × 5, ↑2, ReLU, 256
C-5 × 5, ↓2, ReLU, 256	TC-5 × 5, ↑2, ReLU, 256
C-5 × 5, ↓2, 256	TC-3 × 3, 256

probability mass function is used during inference. This statistical approximation and its impact on the optimization of variational autoencoders remains an open question, as discussed in Section 3.3.9.

Two aspects link to variational Bayesian approaches, such as variational autoencoders. First, the entropy model is trained by optimizing the differential entropy of the noisy latent space. Second, this modeling specifies the optimization of both distortion and compression rate. The main characteristic is the variability of the operating points for each trained model for different images, leading to high variability in the achieved compression rate. Therefore, it would be necessary to train a large set of models to achieve compression values close to the desired rate.

The exploration of this problem, and similar ones concerning the control of neural network optimization, was presented in Section 3.3.8. The work of Zhang [47] stands out for its more elaborate training strategy. In addition to the primary variational parametric model, an auxiliary neural model receives data information (in the form of the latent space) and the rate parameterization. This strategy involves using universal quantization per arbitrary quantization interval, with the auxiliary neural model predicting the quantization step necessary to achieve the desired rate. This model requires more elaborate training in multiple steps [47] and is a multi-rate approach, similar to the methods in Section 3.3.3. However, the approach suffers from the difficulties of rate-constrained optimization, showing little variability in reconstruction quality, with PSNR saturating at almost constant values, even at low rates.

Another issue is that the problem is solved using a neural network as a black box, transforming it into an optimization problem through non-linear transformation. This approach may overlook the reasons for issues like "highly saturated results". The references cited in the work may not be directly relevant to the thesis-specific focus.

The philosophy behind this work is to control the rate of the models while maintaining the nuances of Ballé's seminal works [45, 23]. The focus on simplicity aims to make the proposal a straightforward plugin for frameworks based on Ballé's works, which dominate the literature. Two possible paths are outlined:

- Modifying the optimization function.
- Utilizing auxiliary models with a simple strategy to maintain the usual optimization function.

Considering the research objectives, a simpler approach is to modify the loss function of variational autoencoders and study the nuances of this modification.

4.4 Discussions

This chapter has elucidated crucial aspects of image compression using neural networks, particularly focusing on optimization methodologies and challenges associated with rate control. A nuanced understanding of these concepts is essential for comprehending the methods proposed in the ensuing chapters, as they form the foundation for the proposed approaches.

Valuable pointers for developing practical solutions can be discerned from the insights gleaned. Optimizing the logarithm of data involves optimizing KL-divergence, as shown in Equation 4.6. This optimization paradigm underscores the essence of variational Bayesian optimization, particularly in autoencoder-based rate-distortion optimization, elucidated in Equation 4.14. The behaviors illustrated in Figure 4.2, discussed in Section 4.1.1, are inherent consequences of the loss functions and the neural network architectures involved.

These discussions naturally lead to exploring potential solutions to these challenges, particularly in achieving effective rate control using variational autoencoder models. One direct solution is modifying the loss function, potentially employing Lagrangian relaxation to establish the constraint into the loss. This idea sets the stage for further investigations into overcoming the challenges outlined, especially in achieving nuanced rate control in neural image compression.

Chapter 5

Rate-Constrained Learning-based Image Compression

This chapter details the adoption of Lagrangian relaxation as a solution to achieve target bitrate control in neural networks and the repercussions of such a proposal. To present this approach, Section 5.2 presents the preliminary concepts regarding the optimization with restrictions. The desired properties of a loss function that makes the neural network converge to a target bitrate are given in Section 5.3, followed by the proposed bitrate loss and its deep analysis in Section 5.4. Lastly, several experiments are performed in Section 5.5 to show the method's performance and some empirical evidence for the theoretical problems pointed out throughout this chapter. One experiment in particular is evidence of a highly pursued objective of this approach, which is to be a simple plug-in solution, showing that the strategy still applies to a completely different architecture, as shown in Section 5.5.7.

5.1 Mathematical Notation Rules

An important point relates to the mathematical notations adopted here, which will become heavier than Chapter 4. Even though it will follow these earlier notations, it is important to highlight that it will expand in many ways. Therefore, it is interesting to make it clear a pattern of new symbols that will appear:

- **Non-constant terms**: these terms will always use the function notation, like $f(x)$;
- **Constant terms**: in this case, the function notation will not be adopted. Whenever a term does not adopt the function notation, it is not a function of anything. Therefore it is constant;

- **Time-dependent terms**: whenever a term is also a function of time, it will be subscribed. For example, $f_t(x)$ would denote a function of x which is also time-dependent;
- **Parametrizations**: whenever a term of function fundamentally dictates the behavior of some function, the parametric function notation will be adopted. In $f_t(x; r)$, there is a function f whose variable is x, which is time-dependent and has a parameter r, which influences different function behaviors.

5.2 General Formulation of a Target Bitrate Loss

Rate-distortion optimization often leads to models with significant variations in reconstruction concerning rate and distortion, as discussed in Section 4.1. To address this, modifying the standard loss function to control the operating point of the rate R^{est} based on a target rate r^{target} may be necessary. A more complex notation for terms will be adopted to align with the notations used in subsequent approaches, where it will be further extended.

It is important to clarify that the rate, originally denoted as R^{est}, is based on estimating entropy in the latent space, as presented in the previous chapter. The input data $\mathbf{x}$ and its reconstructions are represented as $\hat{\mathbf{x}}$ during inference, where quantization is applied, or $\tilde{\mathbf{x}}$ during training, where reconstruction uses uniform noise approximation. Similarly, the latent spaces are denoted as $\mathbf{y}$, $\hat{\mathbf{y}}$, and $\tilde{\mathbf{y}}$. The estimated rate R^{est} is given by:

$$R^{est}(\tilde{\mathbf{y}}) = -\mathbb{E}_{q_{\tilde{\mathbf{y}}}}\left[\log p_{\tilde{\mathbf{y}}}(\tilde{\mathbf{y}})\right] = -\sum_{i}^{n} q_{\tilde{y}_i}(\tilde{y}_i) \log p_{\tilde{y}_i}(\tilde{y}_i) \tag{5.1}$$

where $R^{est}(\tilde{\mathbf{y}})$ is the estimated rate of the noisy latent, $\mathbb{E}_{p_{\tilde{\mathbf{y}}}}$ represents the expectation, and n is the dimensionality of the latent space. Equation 5.1 considers the factorization of the distribution $p_{\tilde{\mathbf{y}}}$, where $\tilde{\mathbf{y}} = [\tilde{y}_i]_{i=1}^{n}$ and $p_{\tilde{\mathbf{y}}} = [p_{\tilde{y}_i}]_{i=1}^{n}$. Analogously, it considers the factorization of the variational distribution $\boldsymbol{q}$. The rate for $\hat{\mathbf{y}}$ can be evaluated as:

$$R^{est}(\hat{\mathbf{y}}) = -\mathbb{E}_{q_{\hat{\mathbf{y}}}}\left[\log p_{\hat{\mathbf{y}}}(\hat{\mathbf{y}})\right] = -\sum_{i}^{n} q_{\hat{y}_i}(\hat{y}_i) \log p_{\hat{y}_i}(\hat{y}_i) \tag{5.2}$$

However, in the context of neural network training, R^{est} is a function of the neural network, estimated from the noisy latent space $R^{est}(\tilde{\mathbf{y}})$.

Equation 4.1 introduced the optimization problem considering an upper limit for the rate. This Lagrangian can be rewritten as a stronger constraint:

$$\min D \tag{5.3}$$

$$\text{such that } R^{est} = r^{target} \tag{5.4}$$

In this case, a rate $R^{est} \leq r^{target}$ is not desired; instead, a rate $R^{est} = r^{target}$ is required. Optimization problems with stronger constraints are usually challenging to solve, and incorporating direct constraint optimization in a neural model is not straightforward. However, a widely used tool in such cases is Lagrangian relaxation [117].

5.2.1 Setup of Lagrangian Relaxation for the Loss Function

As a fundamental technique in optimization, employing bounds is a core strategy [117]. Formally, when given a set $M \in \mathbb{R}^n$ and a simple function g (such as linear or quadratic), the objective is to find a lower bound [117]:

$$\min g(\mathbf{x}) \tag{5.5}$$

Lagrangian relaxation becomes relevant when the set M is defined as $M = N \cap \mathbf{x} : \mathbf{f}(\mathbf{x}) = 0$, where N provides a simplified subspace for optimizing g, while the constraints $\mathbf{f} = (f_1, ..., f_m)$ complicate the problem:

$$\min g(\mathbf{x}), \quad \mathbf{x} \in N, \quad f_j(\mathbf{x}) = 0, \quad j = 1, ..., m \tag{5.6}$$

Upon adopting this technique, a Lagrangian is introduced as a function of $\mathbf{x}$ and an auxiliary vector $\boldsymbol{\alpha} \in \mathbb{R}^m$ [117]:

$$N \times \mathbb{R}^m \ni (\mathbf{x}, \boldsymbol{\alpha}) \implies J(\mathbf{x}, \boldsymbol{\alpha}) = g(\mathbf{x}) - \sum_{j=1}^{m} \alpha_j f_j(\mathbf{x}) = g(\mathbf{x}) + \boldsymbol{\alpha}^T \mathbf{f}(\mathbf{x}) \tag{5.7}$$

where $\mathbf{f}$ represents the m-dimensional constraint vector. Essentially, the Lagrangian substitutes each constraint with a linear "price" to be either paid or received, contingent upon the sign of α_j. The set N can be viewed as the "universe" of the variable $\mathbf{x}$, while $\mathbf{f}$ encapsulates the relaxed constraints [117].

5.2.2 Translating the Formulation to the Bitrate Control Loss

Using this optimization tool, the problem with the strong constraint in Equation 5.3 can be transformed into one where the constraint $R^{est} = r^{target}$ is relaxed. To illustrate this,

consider $\boldsymbol{x}$ as the original data and $\tilde{\boldsymbol{x}}$ as the reconstructed data:

$$J(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = D(\boldsymbol{x}, \tilde{\boldsymbol{x}}) + \beta f(\mathbf{y}; r^{target}) \tag{5.8}$$

where $D(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ is the distortion measure to be minimized, and $f(\mathbf{y}; r^{target})$ represents the rate-constraining function introduced in the Lagrangian. The constraining function generally depends on the latent $\boldsymbol{y}$, which can refer to either the noisy latent $\tilde{\boldsymbol{y}}$ or the quantized latent $\hat{\boldsymbol{y}}$, depending on $\boldsymbol{x}$. To maintain concise notation, consider it simply a function of $\boldsymbol{y}$. These notations will be clearer as the specific loss is constructed for the problem. The constant β represents the Lagrangian "price" paid in terms of distortion to achieve the desired rate. The restriction function f is an arbitrary function that acts as a cost function for the problem, i.e., $f(\mathbf{y}; r^{target}) \to 0$ when minimizing $J(\boldsymbol{x}, \tilde{\boldsymbol{x}})$.

5.3 Properties of a Target Bitrate Loss

The primary aim was to devise a loss suitable for neural networks. Defining some essential properties for such a loss is necessary to achieve this goal.

5.3.1 Differentiability

Neural networks rely on the gradient descent technique, which involves propagating derivatives of the loss for the parameters. Hence, differentiability is a vital property of the loss function. In machine learning, many activations and functions aren't differentiable across their entire domains. For instance, consider the ReLU function (described in Section 2.2.7):

$$ReLU(x) = \max(0, x) \tag{5.9}$$

where max denotes the maximum of the two values. This function isn't differentiable at $x = 0$. Despite this, it's widely used, and approximations for these derivatives are commonly employed in major frameworks. Another example is the absolute value function, occasionally used in machine learning contexts:

$$|x| = \begin{cases} x, & \text{if } x \geq 0 \\ -x, & \text{otherwise} \end{cases} \tag{5.10}$$

This function isn't differentiable at $x = 0$. Nonetheless, it's frequently utilized with gradient approximations at these non-differentiable points.

The objective is to devise the simplest loss function that meets the requirements. For differentiable functions, options include trigonometric functions such as cosine and sine,

and polynomials. In the following sections, we'll narrow down the possibilities for functions possessing ideal properties to serve as loss functions for gradient descent optimization of neural networks.

5.3.2 Unique Global Minimum

The second essential property is having a unique global minimum, crucial when designing a loss function to achieve a target rate. A single global minimum prevents the neural network from converging erroneously or encountering optimization difficulties.

For instance, suppose a target r^{target} is to be reached by a loss function following Equation 5.8. In such a scenario, there is a balance between distortion D and the target function f. Now, consider a function $f(x) = (x-2)(x-3) = x^2-5x+6$. This polynomial has two roots at $x = 2$ and $x = 3$. In a minimization problem, these roots represent points where the loss function achieves a zero value. If penalizing a rate mismatch, it might encourage the neural network to converge towards a rate of 2. However, as distortion typically decreases with a rate increase, convergence towards a rate of 3 might be stimulated. This conflict undermines the objective of attaining a rate of 2.

Similar challenges arise with periodic functions like sine and cosine, which feature multiple minima along the x-axis. While modifications can be made to these functions to align with the objectives, such complexities can impede neural network convergence and diverge from the aim of simplicity. Therefore, this property excludes many elementary functions as potential candidates for the target loss function f.

5.3.3 Bounded Property

Another critical property is that the function yields bounded values across its domain. These restrictions apply to the target function candidate and general loss functions used for optimizing neural networks. Functions being minimized should be bounded below. Unbounded functions pose a problem in minimizing a loss function. For example, consider a linear function $f = x - 3$. Given the objective of controlling deviation from 3, convergence towards 3 minimizes the loss for x values greater than 3. However, converging towards any value less than 3 decreases the total function value as f becomes negative, encouraging convergence to lower rate values. Hence, the function must be bounded below across its domain.

5.3.4 Zero-Value at the Minimum

Consider a polynomial $f(x) = (x-2)^2 + 2 = x^2 - 4x + 6$. This polynomial has no roots, even though it could be a suitable candidate considering the properties outlined. In the

factorized version, the constant term +2 prevents the loss from reaching zero even when the target value is attained. While this is not a problem for neural network convergence, adding extra terms contradicts the aim of designing the simplest possible loss function.

5.3.5 Symmetry around the Minimum

Another ideal property is the symmetry of the function around the global minimum, representing the situation where the neural network reaches the desired rate. A non-symmetric function around the global minimum is undesirable because it would penalize the network more heavily as the rate fluctuates above or below the target rate. While this can be desirable in specific contexts, maintaining simplicity by using symmetric functions for the first approach is preferred. Additionally, asymmetric penalization can make neural network training harder as the rate rebounds between the minima.

5.3.6 Architecture-Independent

A desired property is that the approach can be applied to any neural solution, meaning it works with different architectures and objectives. Following the path of simplicity, the constraint function should be additive over the fundamental objective of the neural network. This approach simplifies the optimization of the component dictating this feature of the model. Therefore, following a Lagrangian relaxation setup would yield this feature.

5.4 The target bitrate loss

Considering the highlights of Section 5.3, it is possible to narrow the view of potential elementary functions that could satisfy the requirements. The simplest one would be the polynomials with multiple roots at the same value, like:

$$f(x) = (x - a)^n \tag{5.11}$$

where n is an even value and a is a positive value. The necessity of n being an even number relies on the restrictions described in the last section. Therefore, the simplest function that has all the desired properties is a quadratic function.

Given this, the loss can be written as:

$$f(\mathbf{y}; r^{target}) = \left(\frac{r^{target} - R^{est}(\tilde{\mathbf{y}})}{r^{target}} \right)^2 \tag{5.12}$$

where $\mathbf{y}$ is the value of the latent and $\tilde{\mathbf{y}}$ is the noisy latent. The function $R^{est}(\tilde{\mathbf{y}})$ stands for the estimated Shannon entropy of the noisy latent based on the factorized entropy model $p_{\tilde{\mathbf{y}}}$:

$$R^{est}(\tilde{\mathbf{y}}) = -\mathbb{E}_{q_{\tilde{\mathbf{y}}}}\left[\log p_{\tilde{\mathbf{y}}}(\tilde{\mathbf{y}})\right] = -\sum_{i}^{n} q_{\tilde{y}_i}(\tilde{y}_i)\log p_{\tilde{y}_i}(\tilde{y}_i) \tag{5.13}$$

where $\mathbb{E}_{p_{\tilde{\mathbf{y}}}}$ represents the expectation, and n is the dimensionality of the latent space. Equation 5.13 is obtained considering the factorization of the distribution $p_{\tilde{\mathbf{y}}}$, where $\tilde{\mathbf{y}} = [\tilde{y}_i]_{i=1}^{n}$ and $p_{\tilde{\mathbf{y}}} = [p_{\tilde{y}_i}]_{i=1}^{n}$, following Balléś approaches [45, 23]. The full loss of Equation 5.8 becomes:

$$J(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = D(\boldsymbol{x}, \tilde{\boldsymbol{x}}) + \beta\left(\frac{r^{target} - R^{est}(\tilde{\mathbf{y}})}{r^{target}}\right)^2 \tag{5.14}$$

Thus, the constraint of the target rate is tied to the minimization of $f(\mathbf{y}; r^{target})$ whose relevance is determined by the constant β. The greater the deviation of $R^{est}(\tilde{\mathbf{y}})$ from r^{target}, the higher the penalty in the cost function. In the extreme case where $R^{est}(\tilde{\mathbf{y}}) = r^{target}$, the rate does not contribute to the cost function, and the model attempts to optimize distortion only.

As explained earlier, the chosen function $gf(\mathbf{y}; r^{target})$ is smooth throughout its support, which is ideal for optimization in neural networks. Furthermore, the quadratic function has the property of decreasing the penalty for values close to r^{target} and increasing the penalty for values far from r^{target}, which can be interesting for optimization. Another characteristic of the loss function is the normalization factor. This normalization proved effective in making the loss function more robust to all target rates. Although the quadratic function has been chosen for the study, any function with desirable properties can be used. Nevertheless, unless a specific behavior is expected in the convergence, the other candidates are not expected to bring noticeable improvements.

5.4.1 Analysis of the Proposed Modification

The proposed loss function can be expressed as follows:

$$J(\mathbf{x}, \tilde{\mathbf{x}}) = D(\mathbf{x}, \tilde{\mathbf{x}}) + \beta\frac{R^{est}(\tilde{\mathbf{y}})^2}{\left(r^{target}\right)^2} - 2\beta\frac{R^{est}(\tilde{\mathbf{y}})}{r^{target}} + \beta \tag{5.15}$$

where y is not shown as a variable of J because it is a function of x. When $R^{est}(\tilde{\mathbf{y}}) = r^{target}$, the loss function reduces to $J = D$. Therefore, only the distortion will be minimized whenever the target rate is met. Following this rationale, the function induces a local minimum at $R^{est}(\tilde{\mathbf{y}}) \to r^{target}$. These local minima are influenced not only by $R^{est}(\tilde{\mathbf{y}}) \to r^{target}$ but are also determined by low values of D at these points.

The rate term, defined by the function $f(\mathbf{y}; r^{target}) = \beta \frac{R^{est}(\tilde{\mathbf{y}})^2}{(r^{target})^2} - 2\beta \frac{R^{est}(\tilde{\mathbf{y}})}{r^{target}} + \beta$, is a parabola with two roots at $R^{est}(\tilde{\mathbf{y}}) \to r^{target}$. The term β is related to the "width" of the parabola: higher values will narrow the parabola, and lower values will make it wider, as shown in Figure 5.1. Thus, higher values of β will penalize deviations from $R^{est}(\tilde{\mathbf{y}}) = r^{target}$ more severely. Consequently, this hyperparameter will act as a control mechanism to manage the variance of the achieved rate. A point worth noting is that excessively high values for β impede the convergence of the solution, as the Lagrangian relaxation approaches the constrained problem without relaxation, where $\beta \to \infty$.

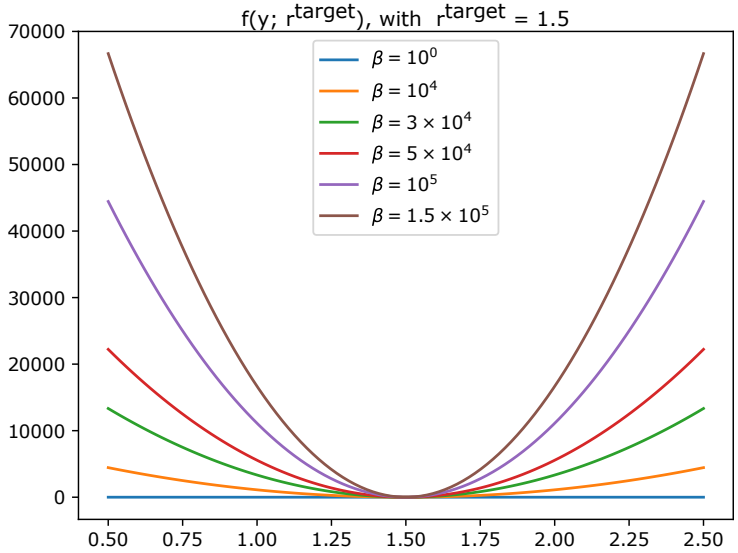


Figure 5.1: This graph depicts the overlay of different parabolas of the form $f(\mathbf{y}; r^{target}) = \left(\frac{r^{target} - R^{est}(\tilde{\mathbf{y}})}{r^{target}}\right)^2$, with $R^{est}(\tilde{\mathbf{y}}) \in [0.5, 2.5]$, $r^{target} = 1.5$ and $\beta = \{10^0, 10^4, 3 \times 10^4, 5 \times 10^4, 10^5, 1.5 \times 10^5\}$. In this case, it is observable that β controls this "width" of the parabola, which in this instance is linked to the range required for the parabola to rise sharply. This derivative with very high values plays a role in influencing the variance of the achieved rates, as deviations from $R^{est}(\tilde{\mathbf{y}}) = r^{target}$ will be penalized more severely. It is interesting to note that, for instance, $\beta = 10^0$ makes the parabola appear flat compared to the parabola with $\beta = 1.5 \times 10^5$. This implies that deviations in this range $R^{est}(\tilde{\mathbf{y}}) \in [0.5, 2.5]$ will be less penalized when $\beta = 10^0$ compared to $\beta = 1.5 \times 10^5$.

It is worth mentioning that the term β is also dependent on another hyperparameter, r^{target}. This dependency is because the relevance of the rate to the optimization will be relative to the relevance of the distortion. Naturally, when using lower values for r^{target}, the distortion D will be larger, which will decrease the significance of the constraint $f(\mathbf{y}; r^{target})$. Therefore, higher values of β are expected to be necessary as the values of r^{target} decrease.

The characteristic of the function can be seen in Figure 5.2 in the context of the convex hull. For this illustration, random points were chosen in the rate-distortion space. The rate parabola can be considered a mechanism to focus on the desired points in the rate-

distortion space since it contributes to the value of the loss function. Ideally, reaching the red point on the convex hull would be desirable. However, as with any rate control mechanism, constraint optimization is challenging.

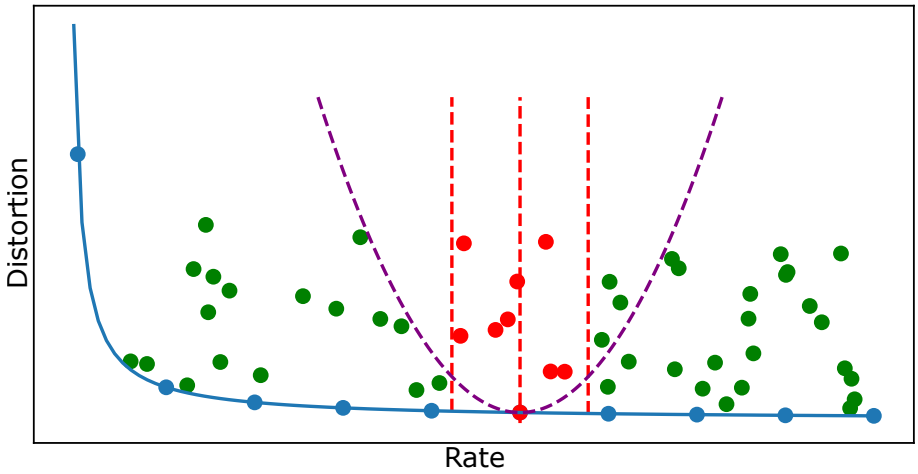


Figure 5.2: The parabola influences the points on the rate-distortion plane since it contributes to the loss function. The dashed purple lines represent the additional values the parabola introduces to the loss function. It establishes a tolerance interval, depicted by the dashed red lines. The dashed red line in the center represents the target rate. This tolerance interval is a consequence of the fact that points outside this interval will have high values for the loss function and, consequently, are less likely to be selected. The neural network can converge to any red points, but it is not necessarily the optimum point on the convex hull (represented by the blue line).

5.4.2 Relation with Variational Inference

The reference architectures, introduced in Section 4.2, implemented for the experiments of this chapter, directly optimize the rate-distortion. They are named VAEs due to the equivalence between them, which can be observed from the derivations of Equations 4.6 and 4.14. The second step of these equations, which defines the calculation of the KL divergence used as a loss function, can be specified as follows:

$$J(\mathbf{x}, \tilde{\mathbf{x}}) = \mathbb{E}_{\tilde{y}\sim q} \log q(\mathbf{y}|\mathbf{x};\phi) - \mathbb{E}_{\tilde{\mathbf{y}}\sim q} \log p_{\mathbf{x}|\tilde{\mathbf{y}}}(\mathbf{x}|\tilde{\mathbf{y}}) - \mathbb{E}_{\tilde{\mathbf{y}}\sim q} \log p_{\tilde{\mathbf{y}}}(\tilde{\mathbf{y}}) + const \tag{5.16}$$

As previously shown, adopting uniform noise of unit width reduces this loss function to a rate-distortion function. For comparison purposes, the function with rate constraint

relaxation shown in Equations 5.8 and 5.12 can be expanded and presented as follows:

$$J(\mathbf{x}, \tilde{\mathbf{x}}) = D(\mathbf{x}, \tilde{\mathbf{x}}) + \beta \frac{(r^{target})^2 - 2r^{target} R^{est}(\tilde{\mathbf{y}}) + R^{est}(\tilde{\mathbf{y}})^2}{(r^{target})^2}$$

$$J(\mathbf{x}, \tilde{\mathbf{x}}) = D(\mathbf{x}, \tilde{\mathbf{x}}) + \beta - \frac{2\beta}{r^{target}} R^{est}(\tilde{\mathbf{y}}) + \frac{\beta}{(r^{target})^2} R^{est}(\tilde{\mathbf{y}})^2 \tag{5.17}$$

This function corresponds to a parabola. Since β and r^{target} are constants, this loss function generically has the following structure:

$$J(\mathbf{x}, \tilde{\mathbf{x}}) = D(\mathbf{x}, \tilde{\mathbf{x}}) + c_1 R^{est}(\tilde{\mathbf{y}}) + c_2 R^{est}(\tilde{\mathbf{y}})^2 + c_3 \tag{5.18}$$

where c_1, c_2, and c_3 are constants and $\mathbf{y}$ is a function of $\mathbf{x}$. The term $J = D + c_1 R^{est}(\tilde{\mathbf{y}})$ has a structure similar to the variational loss function; however, in this case, c_1 is a negative constant. There is also the $R^{est}(\tilde{\mathbf{y}})^2$ term, which balances the negative $R^{est}(\tilde{\mathbf{y}})$, ensuring that the parabola in question never attains a negative value. This quadratic variable does not have a formal correspondence with the derivation of variational inference based on KL divergence, as presented in the equations of Section 4.2. This quadratic term results from the use of a quadratic loss function, whose motivation was discussed in previous topics.

Different types of density and modeling choices would have to be made to equate with the variational derivation presented. However, the implemented architectures retain the modeling of the non-parametric architecture (Section 4.2.3) and parametric architecture (Section 4.2.4). Therefore, considering the analysis from this viewpoint, the loss function in question is not a variational inference function. Nevertheless, similar behaviors of the rate-oriented and variational loss functions have been empirically observed, as will be presented in Section 5.5.4. A formal equivalence, whether arising from different modeling choices or formulations that do not adopt KL divergence, could be the subject of future studies.

Another way to evaluate the function in question could be through the strict view of rate-distortion. As presented in Equation 4.2, the characteristic of rate-distortion optimization is the balance between distortion and rate terms weighted by a constant. For theoretical evaluation purposes, the equation for the rate-oriented function could be

expressed as follows:

$$
\begin{aligned}
J(\mathbf{x}, \tilde{\mathbf{x}}) &= D(\mathbf{x}, \tilde{\mathbf{x}}) + \beta \left(\frac{r^{target} - R^{est}(\tilde{\mathbf{y}})}{r^{target}} \right)^2 \\
&= D(\mathbf{x}, \tilde{\mathbf{x}}) + \beta \left(\frac{r^{target} - R^{est}(\tilde{\mathbf{y}})}{r^{target}} \right)^2 \frac{R^{est}(\tilde{\mathbf{y}})}{R^{est}(\tilde{\mathbf{y}})} \\
&= D(\mathbf{x}, \tilde{\mathbf{x}}) + \beta \left(\frac{r^{target} - R^{est}(\tilde{\mathbf{y}})}{r^{target}\sqrt{R^{est}(\tilde{\mathbf{y}})}} \right)^2 R^{est}(\tilde{\mathbf{y}}) \\
&= D(\mathbf{x}, \tilde{\mathbf{x}}) + h(\mathbf{y}; r^{target}, \beta) R^{est}(\tilde{\mathbf{y}})
\end{aligned} \tag{5.19}
$$

where $h(\mathbf{y}; r^{target}, \beta)$ is a function dependent on $R^{est}(\tilde{\mathbf{y}})$ with parameters β and r^{target}. The components of this equation could represent a generalization of a rate-distortion function, where the relevance of optimizing the rate component is dependent on the hyperparameters and the rate obtained. The balance between the distortion D and the rate $R^{est}(\tilde{\mathbf{y}})$ is given dynamically.

From this perspective, it is as if the relevance of optimizing the rate $R^{est}(\tilde{\mathbf{y}})$ becomes more significant as the rate diverges from r^{target}. One effect of the quadratic function is that if $R^{est}(\tilde{\mathbf{y}})$ is smaller than r^{target}, the relevance of reducing the rate also increases. However, in this case, as the distortion increases, at some point, it becomes more relevant to optimize the gradients to decrease the distortion. This balance would converge the rate to r^{target}.

Equation 5.19 could be multiplied by $h^{-1}(R^{est}(\tilde{\mathbf{y}}); r^{target}, \beta)$, which would maintain the balance between rate and distortion. In this case, the rate-distortion equivalence would be given by:

$$
\begin{aligned}
J(\mathbf{x}, \tilde{\mathbf{x}}) &= h^{-1}(R^{est}(\tilde{\mathbf{y}}); r^{target}, \beta) D + R^{est}(\tilde{\mathbf{y}}) \\
&= \frac{1}{\beta} \left(\frac{r^{target}\sqrt{R^{est}(\tilde{\mathbf{y}})}}{r^{target} - R^{est}(\tilde{\mathbf{y}})} \right)^2 D(\mathbf{x}, \tilde{\mathbf{x}}) + R^{est}(\tilde{\mathbf{y}})
\end{aligned} \tag{5.20}
$$

The balance achieved in this case is that as the rate $R^{est}(\tilde{\mathbf{y}})$ approaches r^{target}, optimizing the rate becomes less interesting, and optimizing the distortion dominates. Again, if the rate $R^{est}(\tilde{\mathbf{y}})$ deviates from r^{target} with smaller values, the relevance of the term accompanying the distortion decreases, but it is balanced again by the increase in the value of the distortion. Conversely, when $R^{est}(\tilde{\mathbf{y}})$ is greater than r^{target}, the relevance of the distortion decreases, as does the distortion itself, but the increase in $R^{est}(\tilde{\mathbf{y}})$ makes it more attractive to optimize the gradients to decrease $R^{est}(\tilde{\mathbf{y}})$.

The parallel analysis raised in Equations 5.19 and 5.20 do not represent a formal

argument regarding a rate-distortion interpretation, but they can help enhance the understanding of the behavior of this loss function. Despite the differences in the modeling of rate-oriented architectures, the other aspects of the modeling presented in Sections 4.2.3 and 4.2.4 were adopted in the proposed method. Notably, modeling and possible formalizations of rate-oriented architectures could be pretty interesting in future works.

5.4.3 The Non-linearity of Target Rate Losses

One crucial aspect emphasized throughout the discussion on loss functions is the necessity of a non-linear complex balance between rate and distortion. This balance is encapsulated in Equation 5.19, mediated by the balance term $h(\mathbf{y}; r^{target}, \beta)$. In reality, even simple modifications to baseline proposals necessitate a non-linear behavior.

The classical rate-distortion balance is given by the following equation:

$$J(\mathbf{x}, \tilde{\mathbf{x}}) = D(\mathbf{x}, \tilde{\mathbf{x}}) + \lambda R^{est}(\tilde{\mathbf{y}}) \tag{5.21}$$

Here, λ can serve as a multiplier for both $D(\mathbf{x}, \tilde{\mathbf{x}})$ or $R^{est}(\tilde{\mathbf{y}})$, highlighting that the interpretation of the term would change in different scenarios. Nevertheless, the optimization principle remains intact in the context of neural networks.

Consider a proposal for $f(\mathbf{y}; r^{target})$ where it is adopted a linear function: $f(\mathbf{y}; r^{target}) = R^{est}(\tilde{\mathbf{y}}) - a$, where $a > 0$.The target rate equation can be expanded to show the following:

$$\begin{aligned} J(\boldsymbol{x}, \tilde{\boldsymbol{x}}) &= D(\boldsymbol{x}, \tilde{\boldsymbol{x}}) + \beta f(\mathbf{y}; r^{target}) \\ &= D(\boldsymbol{x}, \tilde{\boldsymbol{x}}) + \beta \left(R^{est}(\tilde{\mathbf{y}}) - a \right) \\ &= D(\boldsymbol{x}, \tilde{\boldsymbol{x}}) + \beta R^{est}(\tilde{\mathbf{y}}) - \beta a \\ &\approx D(\boldsymbol{x}, \tilde{\boldsymbol{x}}) + \beta R^{est}(\tilde{\mathbf{y}}) \end{aligned} \tag{5.22}$$

In the last line of the equation, the term βa can be disregarded as it is a constant. Replacing β with λ makes it possible to obtain behavioral equivalence to Equation 5.21. However, the problem with Equation 5.21 is that this loss does not converge to a target rate behavior as applied in baseline works.

Instead, one could consider, for example, the absolute value function. However, as defined in Equation 5.10, this function has two different behaviors depending on the value it receives. Consider the expansion of the absolute function using the estimated rate notation:

$$f(\mathbf{y}; r^{target}) = \begin{cases} R^{est}(\tilde{\mathbf{y}}) - a, & \text{if } R^{est}(\tilde{\mathbf{y}}) - a \geq 0 \\ -\left(R^{est}(\tilde{\mathbf{y}}) - a\right), & R^{est}(\tilde{\mathbf{y}}) - a < 0 \end{cases} \tag{5.23}$$

Expanding the case where $R^{est}(\tilde{\mathbf{y}}) - a \geq 0$, there is a behavior equivalence to Equation 5.22. However, considering the case where $R^{est}(\tilde{\mathbf{y}}) - a < 0$, there is some weird behavior, described as follows:

$$
\begin{aligned}
J(\boldsymbol{x}, \tilde{\boldsymbol{x}}) &= D(\boldsymbol{x}, \tilde{\boldsymbol{x}}) + \beta f(\mathbf{y}; r^{target}) \\
J(\boldsymbol{x}, \tilde{\boldsymbol{x}}) &= D(\boldsymbol{x}, \tilde{\boldsymbol{x}}) + \beta |R^{est}(\tilde{\mathbf{y}}) - a| \\
&= D(\boldsymbol{x}, \tilde{\boldsymbol{x}}) + \beta \left(a - R^{est}(\tilde{\mathbf{y}})\right) \\
&= D(\boldsymbol{x}, \tilde{\boldsymbol{x}}) + \beta a - \beta R^{est}(\tilde{\mathbf{y}}) \\
&\approx D(\boldsymbol{x}, \tilde{\boldsymbol{x}}) - \beta R^{est}(\tilde{\mathbf{y}})
\end{aligned}
\tag{5.24}
$$

The last line shows undesirable behavior unless $\beta < 0$. Even though $\beta < 0$ yields classical rate-distortion behavior, it wouldn't work for the case where $R^{est}(\tilde{\mathbf{y}}) - a \geq 0$ since that term is positive. In conclusion, although the absolute function seems suitable for the loss function, except for its differentiability issue at the origin, it doesn't exhibit desirable behavior when its equations are expanded.

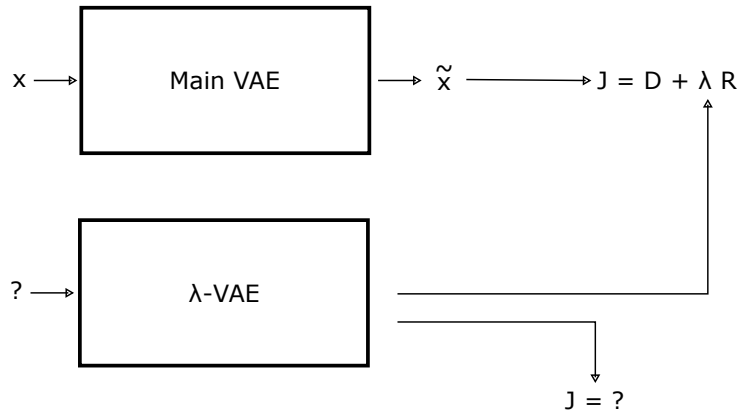


Figure 5.3: Diagram illustrating a general approach employing an auxiliary neural network to assist the main network in optimizing a classical rate-distortion loss. The question mark "?" indicates the unspecified input and loss function required by the neural network to generate a suitable lambda value for the current image, considering the target rate compression.

One potential solution to address the non-linearity issue is incorporating an auxiliary model into the classical rate-distortion loss, as depicted in Figure 5.3. This approach may avoid non-linear functions and relies on a simple classical rate-distortion loss with an auxiliary neural network. However, this is not necessarily the case.

In general, the loss function of the auxiliary model could take various forms, penalizing the model based on the image compression rate, as indicated by the λ output. For instance, any loss satisfying the requirements outlined in Section 5.3 could be employed as the training loss of the auxiliary λ-model. However, additional information is necessary

for this λ model. Considering that the auxiliary model should receive the image itself $\mathbf{x}$ and the target rate r^{target} is reasonable. In this general setup, the auxiliary lambda model outputs:

$$\lambda'(\mathbf{x}, R^{est}(\tilde{\mathbf{y}}); r^{target}) = h(\mathbf{x}, R_t^{est}(\tilde{\mathbf{y}}); \theta) \tag{5.25}$$

where h represents the neural network function, parameterized by θ. The term $R^{est}(\tilde{\mathbf{y}})$ is explicitly included because, even though it is a function of $\mathbf{x}$, it indicates that λ itself is a function of the rate estimation.

The main loss function becomes:

$$\begin{aligned} J(\mathbf{x}, \tilde{\mathbf{x}}) &= D(\mathbf{x}, \tilde{\mathbf{x}}) + \lambda R^{est}(\tilde{\mathbf{y}}) \\ &= D(\mathbf{x}, \tilde{\mathbf{x}}) + \lambda'(\mathbf{x}, R^{est}(\tilde{\mathbf{y}}); r^{target}) R^{est}(\tilde{\mathbf{y}}) \end{aligned} \tag{5.26}$$

The second line of the equation reveals a λ function dependent on the rate $R^{est}(\tilde{\mathbf{y}})$, which is multiplied by the rate $R^{est}(\tilde{\mathbf{y}})$, leading to a non-linear expansion of $R^{est}(\tilde{\mathbf{y}})$. Although this idea is not extensively elaborated, it underscores that all simple sketches of solutions to the problem may result in a non-linear loss. Therefore, the squared function remains one of the simplest functions that satisfies all desired characteristics.

5.4.4 The Mean Rate Shift Problem

Adopting the loss function in Equation 5.8 introduced two new hyperparameters. First, the parameter r^{target}, which is related to the target rate and is connected to the average rate that the network produces considering the training dataset. The second parameter, β, controls the significance of optimizing the rate function in the loss equation. This parameter has a more complex dynamic, given that the relevance of the rate function is relative to the value of distortion. Thus, β does not define an absolute relevance as its weight significance depends on other variables.

The λ in Equation 4.7 has a well-defined interpretation in the context of variational inference. It is tied to the variance of the Gaussian that defines the marginal likelihood (Equation 4.11). Higher or lower values of λ will increase or decrease the reconstruction variance, which implies better or worse reconstruction quality. However, the behavior of the parameters (r^{target}, β) is not fully understood, as introduced in this chapter's proposal. Furthermore, the idea is that this approach should be general and applicable to various neural compression architectures.

Initial experiments with the rate-oriented function in the architectures described in Section 4.2 demonstrated a specific behavior of the loss function. Whenever a model was trained to achieve a rate r^{target}, upon evaluating the model, it would produce a rate

$R^{est}(\hat{\mathbf{y}})$ where $R^{est}(\hat{\mathbf{y}}) \leq r^{target}$. This deviation from the average rate was larger as the target rate r^{target} was smaller. Regarding the hyper-parameter β, it indeed controls the variance of the reconstructions, as will be presented in Section 5.5, but, being a function of $R^{est}(\tilde{\mathbf{y}})$ and r^{target}, it also increased the shift of the obtained average rate. The result was a rate-oriented model with a real rate different from the rate during training.

Theoretical investigations into these preliminary results indicated that one of the possible causes of this issue is related to how quantization is performed during training and that it is more noticeable at lower rates. In the reference architectures, quantization must be replaced by a differentiable approximation to allow gradient back-propagation. Taking the non-parametric architecture as a use case, which introduces the prior hyper, during the inference step, each element y_i of the raw latent $\mathbf{y}$ is scalar-quantized after subtracting the mean μ_{y_i}. Thus, the quantized value $\hat{y}_i$ is obtained after adding the mean:

$$\hat{y}_i = round(y_i - \mu_{y_i}) + \mu_{y_i} \tag{5.27}$$

where *round* rounds to the nearest integer. Conversely, during the training step, quantization is modeled as additive uniform noise:

$$round(y_i - \mu_{y_i}) \approx (y_i - \mu_{y_i}) + \mathcal{U}(-\frac{1}{2}, \frac{1}{2}) \tag{5.28}$$

Combining Equations 5.27 and 5.28 yields the latent at training time:

$$\tilde{y}_i = y_i + \mathcal{U}(-\frac{1}{2}, \frac{1}{2}) \tag{5.29}$$

As the target rate r^{target} shifts towards smaller values, the entropy of the latent representations decreases, equivalent to saying that the uncertainty of the latent $\mathbf{y}$ is reduced. The Gaussian distributions modeling the latent (Equation 4.23) begin to form sharper peaks around their mean, with a smaller scale of variation. In this scenario, the difference between the rate estimated using the noisy latent $\tilde{\mathbf{y}}$ and the rate calculated using the quantized latent $\hat{\mathbf{y}}$ becomes more noticeable.

Consider, without loss of generality, a latent element y_i with a scale parameter $\sigma_i < \frac{1}{2}$ and mean $\mu_i = 0$. The noisy latent element $\tilde{y}_i$ will randomly lie in the interval $[-\frac{1}{2}, \frac{1}{2}]$,

and its likelihood will be calculated as follows:

$$p(\tilde{y}_i|\tilde{\sigma}_i) = \left(\mathcal{N}(0,\tilde{\sigma}_i) * \mathcal{U}\left(-\frac{1}{2},\frac{1}{2}\right)\right)(\tilde{y}_i) \tag{5.30}$$

$$= \int_{\tilde{y}_i-\frac{1}{2}}^{\tilde{y}_i+\frac{1}{2}} \mathcal{N}(0,\tilde{\sigma}_i)(u)du \tag{5.31}$$

$$= f\left(\tilde{y}_i+\frac{1}{2}\right) - f\left(\tilde{y}_i-\frac{1}{2}\right) \tag{5.32}$$

where u is an integration variable and f is the CDF of the Gaussian $\mathcal{N}$. Figure 5.4 schematically shows the likelihood calculated for $\tilde{y}_i$. When $\tilde{y}_i \neq 0$, during training, the likelihood of this element is equivalent to the Gaussian area in the unit interval $\left[\tilde{y}_i - \frac{1}{2}, \tilde{y}_i + \frac{1}{2}\right]$. In contrast, during inference, all values in the interval $\left[-\frac{1}{2}, \frac{1}{2}\right]$ will be rounded to 0, and the likelihood will be the area of this interval.

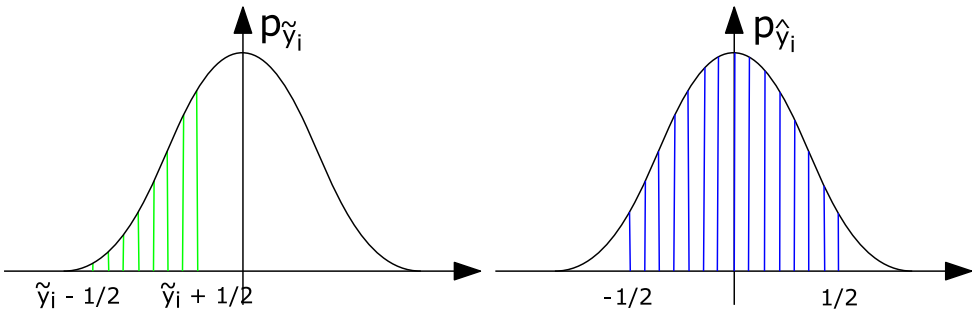


Figure 5.4: The schematics of the probability of the unity interval around a noisy (left) and quantized (right) value.

Given that $p_{\tilde{y}_i}(\tilde{y}_i) \leq p_{\tilde{y}_i}(\hat{y}_i)$, as Figure 5.4 illustrates, the training entropy is higher than the test entropy. The same scheme from Figure 5.4 applies to Gaussians with an arbitrary mean. Preliminary analyses have also indicated that this effect occurs with arbitrary distributions. This conclusion is based on observing deviations from the average rate in the non-parametric model. It is essential to highlight that the average deviation was smaller, as will be explored further in Section 5.5.

The consequence of this misalignment between the training entropy and the test-time entropy is an over-penalization of the loss function through the rate function $f(\mathbf{y}; r^{target})$, as the $R^{est}(\tilde{\mathbf{y}})$ during training will have to be shifted down more than necessary. This mismatch was a fact that likely went unnoticed by most approaches since, in these other cases, the optimization is not done concerning a target rate. However, an interesting point about this deviation at lower rates is that it may hinder the performance of neural networks at low rates, being an aspect to be studied to improve network performance at low rates.

Nevertheless, an initial empirical experiment already pointed to problems based on this relaxation of quantization. Ballé briefly analyzed the impact of the continuous relaxation

of the additive uniform noise in his seminal work [45]. The authors selected a random subset of 2169 images and plotted the points corresponding to many different λ values in one experiment. The plot, which depicted the discrete entropy, where quantization is applied, versus the differential entropy, which handles the noisy latent entropy estimations, shows a deviation from the identity line. This analysis identified the quantization approximation mismatch problem caused by the noisy estimations. This mismatch may be one of the causes that translated as the rate mean-shift problem. That is because, in the current approach, a tight estimation of the rate during training is necessary.

Approximations to address the non-differentiability issue of quantization are applied in practically all neural compression approaches. Thus, different strategies to simulate the operation can lead to different architectural behaviors. Since the loss function is rate-oriented in this chapter's proposal, this can cause different distortions concerning the hyper-parameters (r^{target}, β).

Given that this aspect of the networks became evident with the loss function, it is necessary either to find some way to circumvent the problem of the average rate deviation or to adopt some strategy to estimate the hyper-parameters (r^{target}, β). Based on this analysis, from now on, instead of using the r^{target} in the loss equations, it will be adopted the term R^{param}, which stands for the actual value for the parameter loss that would yield a real rate of r^{target}. It is a function of many variables of the neural network.

5.4.5 Hyper-parameter Estimation

As presented earlier, the proposed loss function is defined by two hyperparameters: (β, r^{target}). r^{target} is related to the average rate that the network produces, while β controls the deviation of the average rate. Both parameters should be defined depending on the architecture and the convergence of the neural network, something that can be difficult to specify beforehand.

The complexity is even greater concerning β since it depends on the rate, in the same way that λ is in the standard rate-distortion Lagrangian (Equation 4.1). Additionally, there is the issue of quantization, which can lead to misalignment between estimates during training and inference time.

A general training heuristic has been designed to define the introduced hyper-parameters. The goal is to avoid many empirical tests, which would complicate the application in real scenarios. The proposed heuristic for training is presented in Algorithm 3, where the R^{param} is the term adjusted to compose the loss and $R^{est}(\hat{\mathbf{y}})$, the rate estimation based on the quantized latent space, refers to the real rate obtained, not including the header of the actual bitstream compression, as it is almost negligible on the rate.

Algorithm 3: Pseudocode of the heuristic for hyper-parameter adjustment

Input : $r^{target} \implies$ desired average rate; $\sigma^2 \implies$ allowed variance for compression; $t_{iter} \implies$ number of iterations for training; $\delta_\beta \implies$ granularity of the adjustment of parameter β

Output : A neural network performing compression with average rate r^{target} using the parameter R^{param} and variance σ^2 using the parameter β

Initialization: $R^{param} = r^{target}$; $\beta = 1$;

while $R^{est}(\hat{\mathbf{y}}) \neq r^{target}$ *and* $\hat{\sigma} \neq \sigma$ **do**
  Train (or refine) the network with the loss function $\{R^{param}, \beta\}$ on the training set until the rate becomes stable, following any desired criterion.
  Perform inference with the pre-trained network on the validation database, calculating the actual compression rate obtained for each image. Considering the set of obtained rates, calculate the average rate $R^{est}(\hat{\mathbf{y}})$ and the variance of the rates, $\hat{\sigma}^2$
  if $R^{param} \neq r^{target}$ **then**
    $R^{param} = R^{param} - [R^{est}(\hat{\mathbf{y}}) - r^{target}]$
  if $\hat{\sigma}^2 > \sigma^2$ **then**
    $\beta = \beta + \delta_\beta$
  if $\hat{\sigma}^2 < \sigma^2$ **then**
    $\beta = max(\epsilon, \beta - \delta_\beta)$
end
After obtaining the parameters $\{R^{param}, \beta\}$ of the loss function that causes the network to compress at an average rate r^{target} with a variance σ^2, train the neural network with the hyper-parameters $\{R^{param}, \beta\}$ for t_{iter} iterations on the training database.

The inputs for the algorithm are defined by the variables $(r^{target}, \sigma^2, \delta_\beta)$, which have the following meanings:

- r^{target} is the average rate the model should achieve (the target rate);
- σ^2 is the allowed tolerance for deviations from r^{target}, defined in terms of the rate variance;
- t_{iter} is the number of iterations for the training stage;
- δ_β represents the granularity of adjustments made to the parameter β.

The heuristic consists of two major procedures: an adjustment stage and a training stage. In the adjustment stage, representing lines $[1, 10]$ of the algorithm, the goal is to estimate values for the pair (β, R^{param}). In this notation, R^{param} represents the actual value used as the real parameter of the loss function. In the second phase, the pre-trained network from the previous phase is refined with these estimated values, which should provide compression at the desired average rate r^{target} and with the required deviation tolerance σ^2. It is essential to highlight that due to the average deviation problem, R^{param} will be set at a value different from the desired value r^{target}, as will be detailed in Section 5.5.

An essential point in the adjustment phase is that the network training, line 2 of the algorithm, should be conducted until stabilization of the rate function for a pair $\{R^{param}, \beta\}$. Figure 5.5 shows an example where the behavior of the estimated entropy during training can be observed as different values for the pair $\{R^{param}, \beta\}$ are chosen.

In Figure 5.5, the typical behavior of the estimated entropy during the training of the parametric model can be observed. Each segment, represented by a different color, indicates a $\{R^{param}, \beta\}$ change in the pair. Changes in R^{param} shift the signal's mean, while alterations in β reduce the signal's variance. In the first segment, in blue, the output generates an average rate significantly distant from the target rate, which is expected since the training has just begun. Each modification of the parameters in the following segments brings the average rate closer to the desired value. For the stopping criterion, about line 2 of the pseudo-code, various techniques can be employed to analyze the signal's convergence around a fixed value, such as techniques that examine the stationarity of functions [118].

In this case, the network used at each step is the resultant network from the previous step. That is, it's a sequence of refinements because training the network from scratch at each stage could delay the initial convergence of the rate, and this approach has proven efficient in terms of results. Figure 5.5 demonstrates that many iterations at each step of the adjustment phase are unnecessary, as the entropy stabilizes quite rapidly once the parameters have been altered. However, it is essential to note that this is architecture-

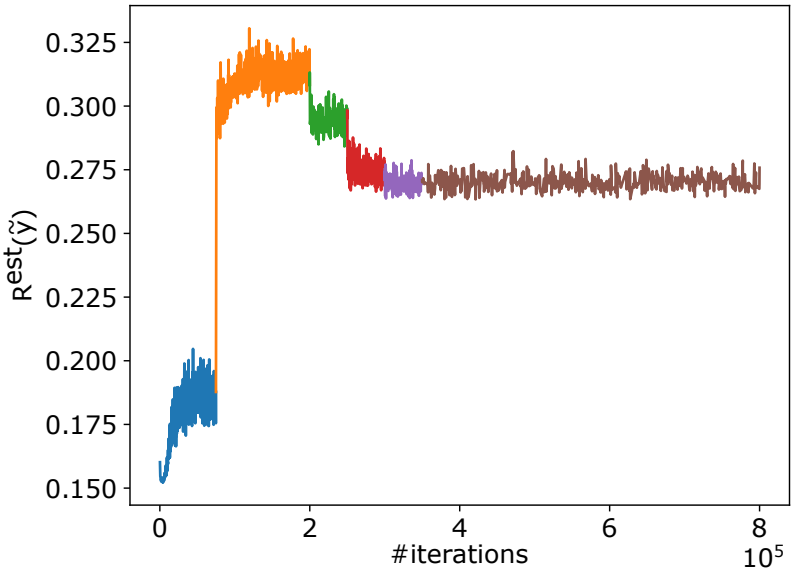


Figure 5.5: This figure displays the behavior of the estimated rate during the training of a version of the parametric architecture with the rate function as an example. Each color represents the refinement of the model with a different set of parameters $\{R^{param}, \beta\}$.

dependent. Approaches utilizing more complex neural models will naturally require more iterations.

Once the network has been trained in one step of the adjustment stage, a validation database must be used. If the results are unsatisfactory, it will be necessary to adjust R^{param} and β (lines 3-9). The term R^{param} can be shifted concerning the average rate difference from the desired value. On the other hand, the term β can be increased or decreased if less or more variance is desired, with the step change determined by δ_β. It is worth noting that the granularity δ_β impacts the time of the refinement stage. This impact is because lower values for δ_β will demand more refinement stages, while higher values for the constant δ_β will expedite the adjustment speed. However, if the granularity is too high, fine control of the compression rate variance may not be possible.

Thus, the first stage of the heuristic can be seen as a procedure for estimating the value of the loss function's hyper-parameters. The result is a set $\{R^{param}, \beta\}$ of hyper-parameters that yield a network with the desired rate and variance. This procedure allows the application of the loss function in different architectures, as it will compensate for variations in the behavior of the loss function across different architectures. The overarching goal is to obtain suitable hyper-parameters regardless of the chosen architecture.

A fundamental point about the heuristic concerns the convergence $R^{est}(\hat{\mathbf{y}}) \rightarrow r^{target}$. Currently, there is no way to mathematically demonstrate that $R^{est}(\hat{\mathbf{y}}) \rightarrow r^{target}$ and that the variance will become the desired one with the heuristic. Since $R^{est}(\hat{\mathbf{y}})$ is a variable of the loss function, which uses the constant r^{target} as a reference, the convergence of the

heuristic relies on the convergence of the neural architecture, as is the case for any use of neural networks.

Considering this fact and the experimental results that will be shown in the results section, it can be empirically posited that the rate-oriented loss function generally has good convergence properties and, consequently, $R^{est}(\hat{\mathbf{y}}) \rightarrow r^{target}$ in most cases. However, if any convergence issues occur, one path is to attempt different or more robust architectures. It is essential to mention that the network and heuristic convergence may be the subject of future studies.

5.5 Results

This section will present the experiments and results obtained with the proposed approach. Section 5.5.1 will detail the datasets and the procedures performed for training and inference. Meanwhile, Section 5.5.2 will introduce the first experiment to validate the rate-oriented approach. The feature related to the variational loss function will be explored in Section 5.5.4. The comprehensive rate-distortion analysis and comparison with reference architectures will be displayed in Section 5.5.5. Figures and charts for comparison and subjective evaluation of the results will be presented in Section 5.5.8.

5.5.1 Training and Testing Specifications

The parametric and non-parametric architectures, whether original or modified, were implemented using a framework provided by the authors via Tensorflow [113]. For the training of the models tested in this chapter, images from the following datasets were considered:

- CLIC Professional dataset [119];
- CLIC Mobile dataset [119];
- DIV2K dataset [120];
- Ultra-Eye Ultra HD dataset [121];
- MCL-JCI [122, 123];
- FLICKR2K dataset [124];

The image preparation involved extracting fragments of size 256×256 pixels from all the datasets. The extracted image fragments do not overlap. To test the method's generality, eight neural networks were trained, each aiming to achieve one of the following rates:

$\{0.06, 0.12, 0.25, 0.5, 0.75, 1.0, 1.5, 2.0\}$, considering each of the reference architectures, the parametric [45] and the non-parametric [23].

The criterion to consider that the model has achieved the target rate is a maximum deviation of 15% from the target rate. This tolerance is quite challenging, especially for lower rates. For instance, in the case of $r^{target} = 0.06$, the range is $[0.051, 0.069]$, allowing minimal variance in the rate. It is crucial to highlight that stricter criteria for the target rate will significantly impact rate-distortion performance

The number of iterations (t_{iter}, in heuristic 3) was set to $500,000$ to check if convergence could be achieved without overly prolonged training. The parameter δ_β was inversely proportional to the rate in the range $[500, 20,000]$. In this case, the lower rates had the most considerable update steps, which was unnecessary for the higher rates. The determining factor is the structure of Equations 5.8 and 5.12 because the relevance of β is relative to the magnitude of the distortion. Naturally, compression at lower rates will yield higher distortion D values, which decreases the importance of rate $R^{est}(\tilde{\mathbf{y}})$ relatively. A higher value of β is also needed to ensure the desired variance, as the tolerance for variation at lower rates is very small. It is worth noting that lower values for δ_β would increase the number of necessary steps for stipulating the hyperparameters R^{param}, β. Conversely, a very high value would reduce the fine control of the variance.

Two datasets were used for evaluating the trained models:

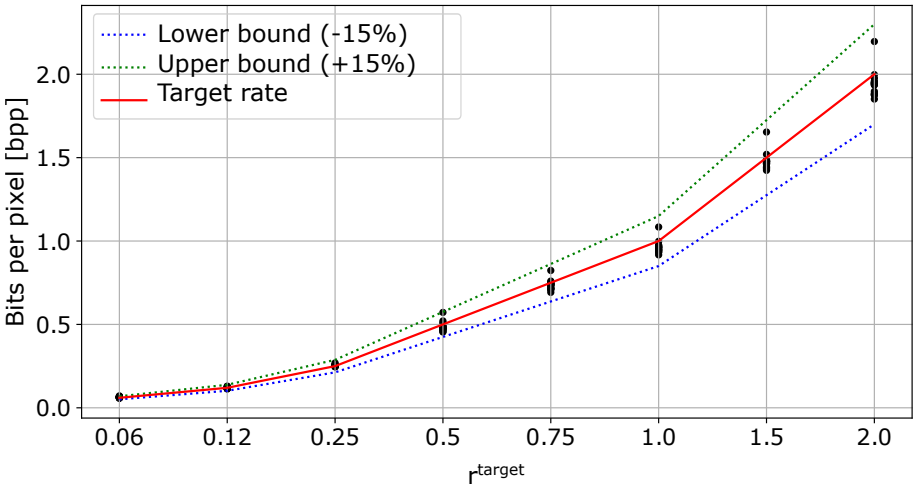
- JPEG AI dataset [125];
- Kodak dataset [75];

The Kodak dataset is the most widely used in the literature and consists of 24 uncompressed images. The JPEG AI dataset, being more recent, comprises 16 images. It is an interesting dataset because it includes high-resolution images, such as 4K, which is a feature not present in the Kodak dataset images.

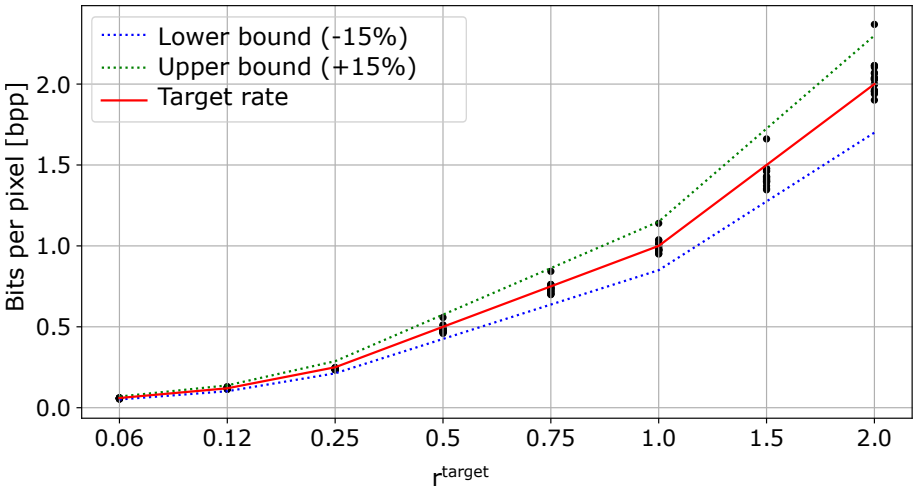
For comparison purposes, six models of each of the architectures were trained, using the following values of $\lambda = \{10^{-4}, 10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}, 2 \times 10^{-1}\}$. The values of λ were chosen to cover the low and high rates usually obtained with these values for this constant. The training process also adopted $500,000$ iterations on the same datasets used for the rate-oriented models.

5.5.2 Analysis of Average Rate and Variance

The initial step is to demonstrate that the proposed approach attains the desired outcomes. In this case, each model aims to achieve one of the specified rates. Thus, experiments were similar to those illustrated in Figure 4.2. The outcomes are displayed in



(a) Parametric Architecture

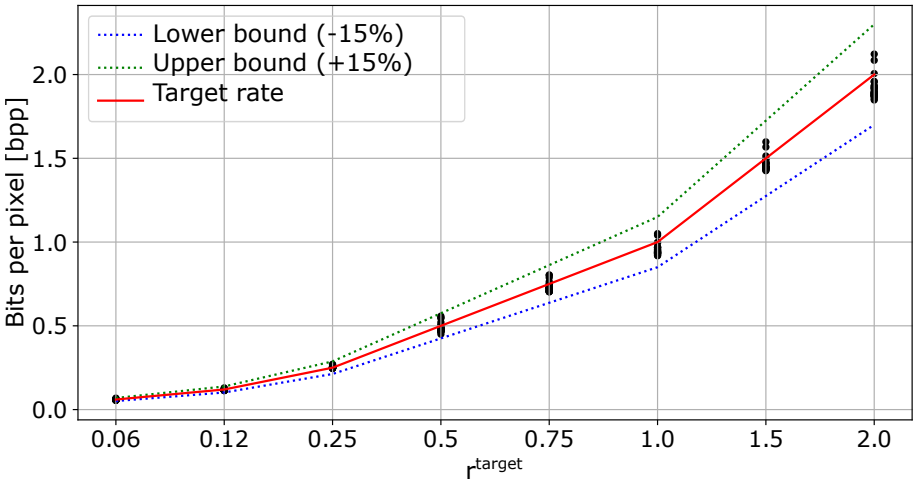


(b) Non-parametric Architecture

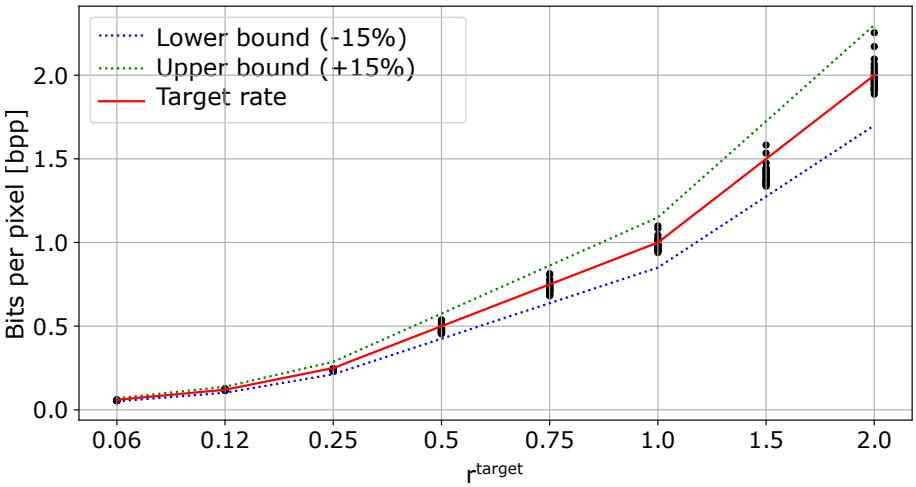
Figure 5.6: Results of both architectures on the JPEG AI database.

Figures 5.6a, 5.6b, 5.7a, and 5.7b. In the case of the parametric architecture, the only point outside the tolerance interval occurred at the rate of 0.06. It is not visible as the deviation was minor, given the small allowable interval at this rate. In contrast, for the non-parametric architecture, the only point outside the tolerance interval occurred at the rate of 2.0. In this case, it is visible in the graph since the tolerance for this rate is higher.

Figures 5.6a and 5.6b show the results for the parametric and non-parametric models applied to the JPEG AI database, respectively. In the case of the parametric model, there was only one violation of the tolerance interval, occurring at the rate of 0.06. It is not apparent in the graph as the interval was very narrow at the rate of 0.06, and the deviation was also barely noticeable. Conversely, in the non-parametric model, the compression of one image was outside the tolerance interval at the rate of 2.0. This violation of the interval is visible in the graph because the tolerance interval is wider at that point.



(a) Parametric Architecture



(b) Non-parametric Architecture

Figure 5.7: Results of both architectures on the Kodak database. In this instance, no points were outside the tolerance interval.

Results on the Kodak database are illustrated in the pair of Figures 5.7a and 5.7b, referring to the parametric and non-parametric models, respectively. In this instance, there were no points outside the tolerance interval. Notably, the images in the Kodak database are of size 512×768 pixels or 768×512 pixels. Therefore, this represents a more straightforward scenario than the JPEG AI database, which contains images of much larger sizes with more details.

Considering the JPEG AI database, which contains 16 images, each evaluated with 8 models (for each of the 8 rates), a total of 128 reconstructions were performed for each architecture. This total implies that only 1 out of 128 reconstructions fell outside the interval for each architecture. It is worth mentioning that these reconstructions were above the tolerance interval by only 3% deviation. In the case of Kodak, containing 24 images results in a total of 192 reconstructions for each architecture.

Thus, it can be asserted that most points lie within the tolerance interval, contrasting with the results in Figure 4.2. These findings suggest that the proposed loss function performs well for images not seen during training and, therefore, can generalize with the rate constraint included via Lagrangian relaxation.

A noteworthy aspect regarding the generalization of the loss function pertains to the number of iterations used to train each model. In this scenario, the networks underwent training for only $500,000$ iterations. One way to enhance the neural network's generalization is to train for more iterations, provided it is not overfitting. Given the convergence exhibited by the introduced loss function, this can potentially decrease the likelihood of obtaining images outside the tolerance interval. These scenarios are tied to the network's generalization capability because the loss function introduces the rate parameter in the optimization. If an increase in the number of iterations results in an excessively long adjustment stage of the heuristic 3, a viable strategy is to increase the value of δ_β to reduce the number of adjustment steps for β.

5.5.3 Impact of the β Parameter

The previous section highlighted that it is possible to achieve the desired rates by appropriately defining the values of the parameters R^{param}, β of the loss function. It was also mentioned in this chapter that the constant β, besides depending on the values related to the rate R^{param}, also controls the variance of the rate.

This β parameter effect can be seen through an experiment where 6 rate-oriented models were trained using the non-parametric architecture. The training specifications followed the ones described in Section 5.5.1. Thus, these models were each trained with $500,000$ iterations. All models had $r^{target} = 0.06$, and each was parameterized by a different value for β. The set of values is given by $[10^0, 10^1, 10^2, 10^3, 10^4, 10^5]$.

Therefore, it is possible to evaluate the impact of the variation of the β parameter on the rate's variance, as per Figure 5.8, which presents the rate values for reconstructions of the Kodak database. The graph shows that the increase of the β parameter decreases the variance of the rates. The decrease obtained by varying the parameter value from 10^0 to 10^2 is especially notable. Beyond this, the reductions in variance are less noticeable. It is worth mentioning that this decrease in variance, resulting from the increase in the value of β, comes at the cost of a drop in rate-distortion performance, as previously mentioned. This drop is because, from the perspective of Lagrangian relaxation, increasing the relevance of the constraint brings the relaxed Lagrangian closer to the constrained Lagrangian, which typically represents a complex problem to be optimized. Thus, defining a β value tunned for the application's requirements is interesting to avoid drops in performance.

Another interesting point regarding the variation of the value of β is related to the effect it has not only on the variance but also on the convergence in the mean. For instance, when the value of β was set to 10^0 and 10^1, it is evident that not even the mean rate was around 0.06. An effect of better convergence of the mean to the target rate $r^{target} = 0.06$ is only observed from $\beta = 10^2$ onwards.

Lastly, it is essential to emphasize that this impact of β depends on r^{target} as it represents the relative relevance of converging the rate requirements with the distortion. In this manner, larger β values are expected to be significant for the results to begin to be substantial. Conversely, smaller β values are expected to have a more significant impact at higher rates. This fact will be evidenced in the next section, presenting the results from Table 5.3a.

Another interesting aspect is the relationship of the β parameter to the distortion measure, in this case, MSE. This relationship can be seen in Figure 5.9. As expected, the increase in the parameter value has an inverse effect on the MSE. This is because different images require different rates to be compressed to the same quality. By forcing the reduction of the variance of the rates produced by the model, this will result in more variation in distortions.

For example, observing the dispersion of distortions at $\beta = 10^0$, one can notice that the points are relatively close, except for the point with the maximum MSE. However, at $\beta = 10^5$, the mean distortion has shifted upwards. Moreover, some points have become entirely separated from the primary grouping. Analyzing the impact of these points, which have experienced significant distortion shifts, can provide a better understanding of network convergence. This understanding might inspire improved strategies to prevent potential excessive deterioration of the reconstructions.

5.5.4 Variational Loss Function Characteristics

As discussed in this chapter, the loss function employed in VAEs, initially proposed in [22], is associated with the variational formulation of Bayesian inference and is defined in terms of the KL divergence between the variational family $q_\phi(\mathbf{y}|\mathbf{x})$ and the true *posterior* $q_\theta(\mathbf{y}|\mathbf{x})$. Considering the modeling presented in the reference work [45], where quantization is modeled as additive uniform noise of width 1, the loss function becomes the rate-distortion Lagrangian.

Not all distortions or perceptual measures lead to a normalizable density. Therefore, equivalence with VAEs cannot always be assured. Affine and invertible transforms and any translation-invariant metric correspond to a normalizable density [45]. It is noteworthy that the loss function introduced in Equations 5.8 and 5.12 may not be considered a

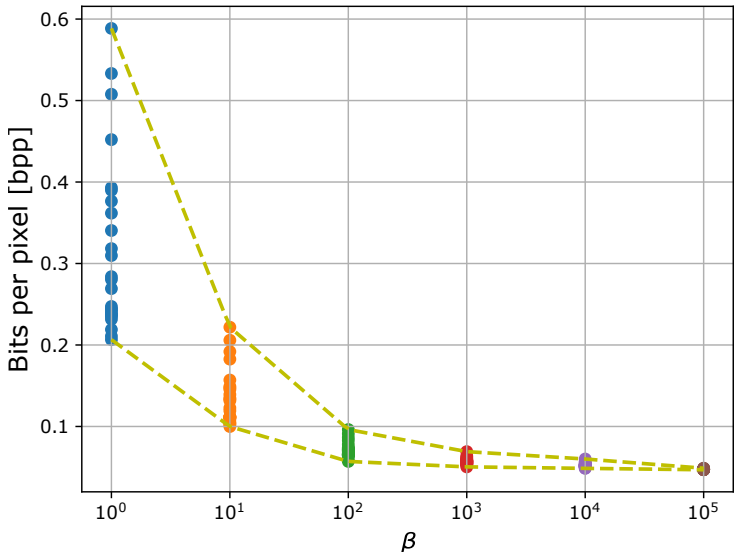


Figure 5.8: The graph shows the relationship between β and the rates produced for reconstructions in the Kodak database. It can be seen that the parameter decreases the variance of the rates and forces the model in question to produce reconstructions with the average rate following the desired.

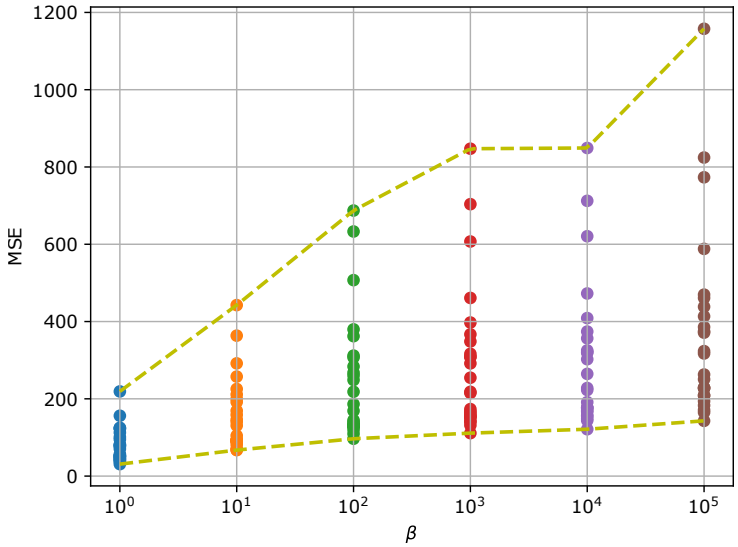


Figure 5.9: The graph shows the relationship between β and the MSE for reconstructions in the Kodak database. It can be observed that, in this case, the parameter has the opposite effect on the MSE. This impact is expected since different images require different rates to be compressed to the same quality. By forcing the reduction of the variance of the rates, the resulting distortions will consequently increase.

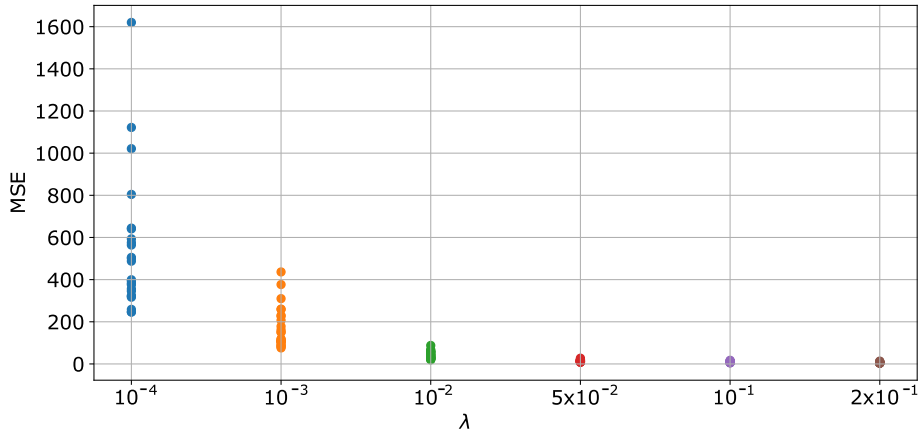


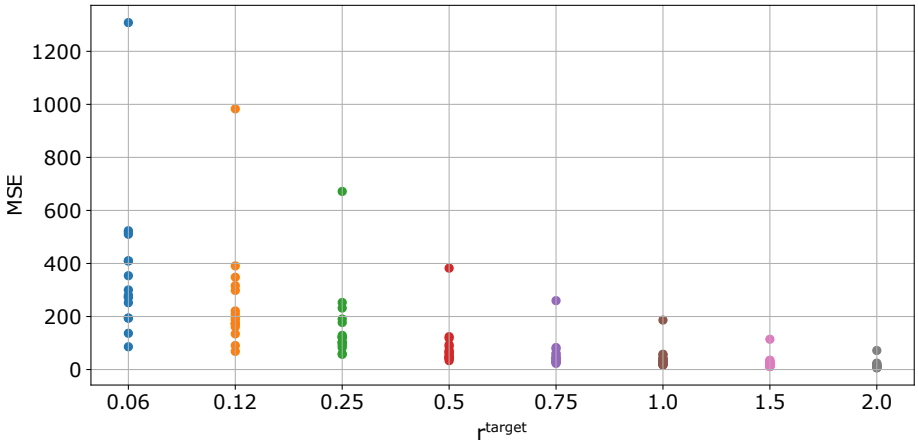
Figure 5.10: MSE behavior for the parametric architecture on Kodak for different λ values. It can be seen that the constant affects the reconstruction variance in each of the models. For instance, in the model with $\lambda = 10^4$, the range of values for the MSE is around $[200, 1600]$. In the case of $\lambda = 2 \times 10^1$, the reconstruction values have significantly reduced variance.

variational optimization function, as presented in Section 5.4.2, since it exhibits some distinct characteristics.

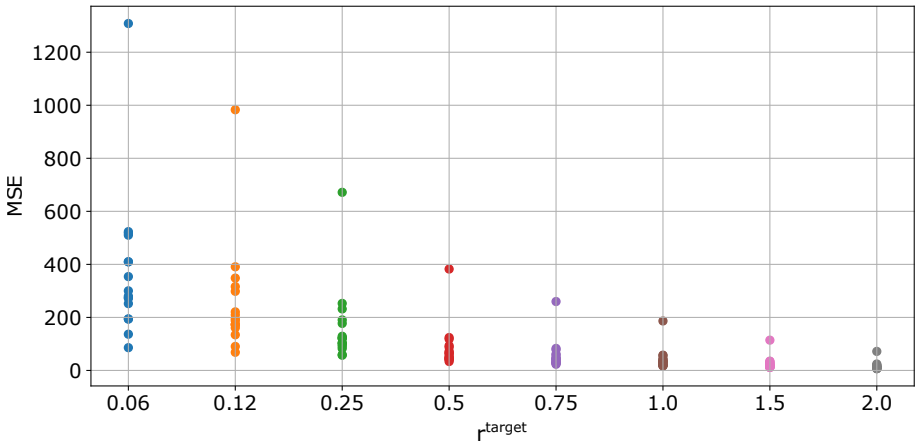
In the case of the Bayesian loss function adopted in VAEs [22], the term λ represents the inverse of the likelihood variance in variational Bayesian inference. It is a term related to reconstruction: high values will reduce the variance of the reconstruction, while low values will allow for higher variance in the reconstruction values. The behavior of this term λ can be observed in Figure 5.10, which displays the MSE values on Kodak for each model trained with a different λ. This experiment refers to the parametric model [23] trained as the reference architecture for comparison with the target rate models.

It is noteworthy that the proposed loss function retains this characteristic of the Bayesian inference function, as demonstrated by Figures 5.11a, 5.11b, 5.12a, and 5.12b. The graphs indicate that the parameter R^{param}, and by transitivity the constant r^{target}, has an effect equivalent to λ in the variational Bayesian formulation. Thus, higher R^{param} values result in lower MSE variance, while lower R^{param} values lead to higher MSE variance.

Another aspect to consider concerns the other hyperparameter of the loss function, β. This parameter will also impact the dispersion of the reconstructions. As previously mentioned, there is an intrinsic link between the value of β and R^{param}. Additionally, β tightens or loosens the rate variation requirements. This change can cause fluctuations in the MSE, which tend to become relatively more consistent as the rates converge to a specific value. However, this effect is less significant than that induced by R^{param} considering the actual values used for β during training.

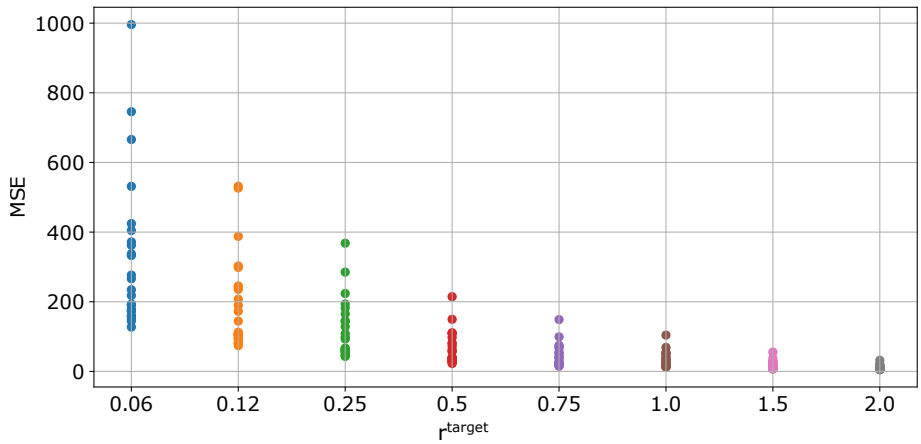


(a) Parametric Architecture

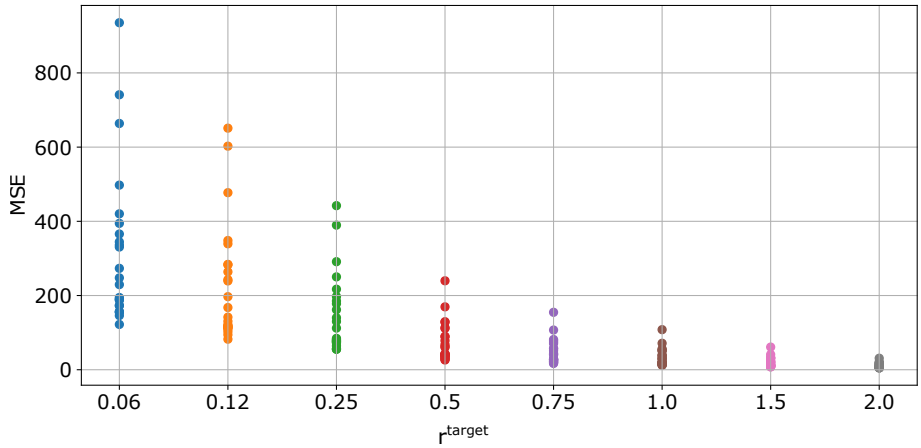


(b) Non-Parametric Architecture

Figure 5.11: MSE on the JPEG AI dataset in both architectures.



(a) Parametric Architecture



(b) Non-Parametric Architecture

Figure 5.12: MSE on the Kodak dataset in both architectures.

The conclusion from this experiment is that the variance characteristics of the reconstruction of the variational function in VAEs are primarily achieved with R^{param} and peripherally with β. There exists a possibility that there might be some mathematical relation with the variational formulation that is not so evident. The mathematical study of loss functions can be insightful for discovering relationships with the standard formulation of Bayesian inference or even with different problem formulations. The intriguing aspects include connections with variational inference and enhancements in the results presented in this section. Thus, it could be an interesting avenue for future work.

5.5.5 Rate-Distortion Performance

This section will present the results considering the rate-distortion performance of the proposed approach. Table 5.1 displays the reconstructions' averages for parametric and non-parametric architectures. The chosen metrics were SSIM, MS-SSIM, and PSNR. It is noteworthy that perceptual metrics have received particular attention concerning neural compression, as neural networks tend to produce good reconstructions considering these metrics even when not explicitly optimized for them.

The averages in the table consider only the images whose reconstructions were within the tolerance range for each rate. An unexpected point is that the parametric model performed similarly or worse than the non-parametric one. The results were equivalent in perceptual metrics, but in PSNR, the non-parametric model exhibited performance comparable to the parametric model. This outcome does not align with the papers that proposed these architectures [45, 23], where the parametric architecture demonstrated notable improvements compared to the non-parametric architecture. This difference is an intriguing finding, as it is a different behavior from the standard optimization of autoencoders and could be studied in more detail in the future.

Table 5.2 displays the results on the Kodak dataset, where the same criteria applied to Table 5.1 were implemented. In this instance, the outcomes also suggest that the parametric architecture did not yield superior results to the non-parametric architecture, reinforcing that the proposed loss function has some specificities concerning the standard loss function.

This result indicates a more significant relative deterioration in the parametric model than the non-parametric model concerning the reference architectures. A relevant point to consider is that the issue of average rate deviation, formally analyzed in this chapter, affected the parametric architecture more than the non-parametric architecture.

This higher mismatch can be observed in Table 5.3, which shows the value of the parameters to achieve the desired rate and variance. It is intriguing to note that the issue of average rate deviation affected the parametric architecture significantly more than the

Table 5.1: Results for the parametric and non-parametric architecture on the JPEG AI base. Averages are calculated considering only the images within the tolerance range for each target rate, representing this value by the column "#". A distinct point from the reference works [45, 23] is that the parametric architecture did not yield better results than the non-parametric architecture

(a) Parametric Architecture

Rate	**#**	**SSIM**	**MSSSIM**	**PSNR**
0.06	15	0.617	0.835	23.18
0.12	16	0.735	0.915	25.62
0.25	16	0.817	0.951	27.80
0.5	16	0.882	0.973	30.04
0.75	16	0.916	0.982	31.67
1.0	16	0.935	0.986	32.79
1.5	16	0.964	0.992	35.04
2.0	16	0.978	0.995	36.75

(b) Non-parametric Architecture

Rate	**#**	**SSIM**	**MSSSIM**	**PSNR**
0.06	16	0.629	0.836	23.20
0.12	16	0.701	0.900	24.90
0.25	16	0.790	0.942	26.93
0.5	16	0.875	0.970	29.74
0.75	16	0.912	0.980	31.42
1.0	16	0.936	0.986	32.84
1.5	16	0.959	0.991	34.77
2.0	15	0.978	0.995	37.44

Table 5.2: Results for the parametric and non-parametric architecture on the Kodak dataset. Here, it can also be observed that the non-parametric architecture had results similar to those of the parametric one.

(a) Parametric Architecture.

Rate	**#**	**SSIM**	**MSSSIM**	**PSNR**
0.06	24	0.609	0.832	23.73
0.12	24	0.710	0.909	26.08
0.25	24	0.797	0.949	28.30
0.5	24	0.870	0.972	30.76
0.75	24	0.909	0.982	32.53
1.0	24	0.930	0.987	33.75
1.5	24	0.961	0.993	36.42
2.0	24	0.975	0.995	38.36

(b) Non-parametric Architecture.

Rate	**#**	**SSIM**	**MSSSIM**	**PSNR**
0.06	24	0.611	0.833	23.82
0.12	24	0.678	0.896	25.40
0.25	24	0.768	0.939	27.32
0.5	24	0.863	0.969	30.30
0.75	24	0.906	0.981	32.17
1.0	24	0.933	0.987	33.74
1.5	24	0.956	0.992	35.86
2.0	24	0.974	0.995	38.33

non-parametric one. As can be seen in the parametric model table, higher values for R^{param} were necessary at lower rates to achieve the target rate. For instance, to reach the actual rate 0.06, an $R^{param} = 0.254$ was required.

Another relevant point, as addressed in the theoretical analysis of Section 5.4.4, is that this effect diminishes as higher entropies are desired. In the case of the parametric model, up to the actual rate of 1.5, a deviation from the mean is noticeable, requiring R^{param} values more significant than the target rate. However, at 2.0, the R^{param} value fluctuated around the exact rate, indicating that the mean shift's effects are negligible.

The contrast becomes evident when observing the parameter values for the non-parametric architecture. A slight mean shift could be attributed to the actual rates 0.06 and 0.12, which required R^{param} values of 0.07 and 0.145, respectively. Percentage-wise, these were the highest deviations between the actual rate and the R^{param} parameter.

This scattered result raises quite an interesting aspect regarding the approach of this chapter. There seems to be some correlation between the mean shift and the degradation of the architecture's performance, which could be further substantiated with additional investigations. If this hypothesis generally holds, it could point to some directions for future work. As the degradation of the reconstruction seems to be tied to the mean shift, which may result from the misalignment of the approximate quantization during training and the actual quantization at test time, the mean shift might be a parameter to guide better quantization strategies. Using the mean shift as a parameter could be helpful in modeling architectures that are not necessarily rate-oriented. One possibility is obtaining better-performing models by observing this effect on the objective rate function, even if these models are not rate-oriented. It is worth mentioning that the problem of the quantization operation in neural compression is still an open issue.

The second experiment of this section aims to compare the degradation that the rate constraint, included via Lagrangian relaxation, has caused in the models. The standard models trained under the same circumstances as the rate-oriented models were used. It is important to recall that these standard models have different rate and distortion values for each λ, dependent on the input image.

The 6 standard models trained for each architecture were interpreted as a single CODEC to conduct a fair comparison. Each test image was applied to each of the 6 models. If any of the models generated a reconstruction whose rate is within the tolerance limit for the target rate, the image was considered for comparison purposes. Moreover, only the images that satisfy the tolerance range concerning the rate-oriented models were chosen. Thus, a set of images with a specific rate is obtained. This selection enables a direct comparison using the averages of the reconstructions, as was done in Tables 5.2 and 5.1.

Table 5.3: The parameters $\{R^{param}, \beta\}$ used for the loss function, after the heuristic adjustment step. A notable point is that the parametric model suffered much more intensely from the mean shift. For instance, to achieve an actual rate of 0.06, it was necessary to use the value $R^{param} = 0.254$. In contrast, the non-parametric model had little to no mean shift. This demonstrates that Gaussians are more affected by the shift than arbitrary distributions.

(a) Parametric Architecture.

Target Rate	R^{param}	β
0.06	0.254	60000
0.12	0.275	55000
0.25	0.410	30000
0.5	0.65	9000
0.75	0.86	6500
1.0	1.08	5500
1.5	1.55	5000
2.0	1.93	2500

(b) Non-parametric Architecture.

Target Rate	R^{param}	β
0.06	0.07	55000
0.12	0.145	55000
0.25	0.25	25000
0.5	0.53	9000
0.75	0.8	6000
1.0	1.08	5500
1.5	1.5	3000
2.0	2.0	1500

The comparisons are displayed in Table 5.4 for the parametric architectures and in Table 5.5 for the non-parametric architectures. In both cases, SSIM, MSSSIM, and PSNR are assessed. In all scenarios, it is interesting that the difference in subjective metrics is relatively small, except at the lower rates. The difference becomes more significant in PSNR. However, it is known that SSIM and MS-SSIM correlate more with human perceptual quality [126, 127].

It is worth noting that no specific training was conducted for the perceptual metrics, with MSE being adopted as the distortion measure in all models. However, as previously emphasized, neural networks produce images with high perceptual quality even when not explicitly trained for such metrics. Therefore, MSE was chosen to evaluate PSNR while maintaining high perceptual quality.

As expected, even using reconstructions from any of the 6 standard trained models, only some of these reconstructions meet the requirements of target rates, showing that coding with rate restriction is a very relevant feature in the field of neural coding. On the other hand, this chapter's proposal can achieve good rate-distortion results at each rate using only one model per rate.

The results from Tables 5.4 and 5.5 also make evident the fact mentioned earlier about the deterioration of the parametric model. This relative deterioration is more pronounced in the pair of parametric architectures than in the pair of non-parametric architectures, using PSNR as a reference. These results re-emphasize the possible correlation between the mean deviation and deterioration concerning the reference models.

Table 5.4: Comparison of the proposed parametric quantized model (denoted as "Proposal") to the rate with the Ballé parametric model (denoted as "Standard") on both JPEG AI and Kodak databases. The number of common images whose rate met the tolerance criteria is displayed in the "#" column.

(a) JPEG AI.

Rate	**#**	**SSIM**		**MSSSIM**		**PSNR**	
		Standard	**Proposal**	**Standard**	**Proposal**	**Standard**	**Proposal**
0.06	4	0.736	0.660	0.888	0.849	27.07	24.16
0.12	2	0.737	0.701	0.920	0.912	24.22	23.21
0.25	5	0.936	0.898	0.981	0.972	33.28	29.60
0.5	3	0.934	0.908	0.987	0.982	32.71	29.46
0.75	6	0.975	0.960	0.995	0.990	37.96	33.50
1.0	12	0.948	0.929	0.989	0.984	36.45	33.03
1.5	12	0.972	0.958	0.994	0.991	37.47	34.88
2.0	6	0.979	0.982	0.995	0.995	36.25	34.93

(b) Kodak.

Rate	**#**	**SSIM**		**MSSSIM**		**PSNR**	
		Standard	**Proposal**	**Standard**	**Proposal**	**Standard**	**Proposal**
0.06	9	0.756	0.703	0.902	0.872	27.83	25.06
0.12	3	0.614	0.618	0.870	0.885	24.19	23.67
0.25	10	0.876	0.840	0.968	0.958	33.52	30.53
0.5	6	0.894	0.871	0.976	0.973	33.65	30.58
0.75	13	0.932	0.917	0.987	0.984	36.45	33.38
1.0	16	0.949	0.933	0.990	0.987	38.19	34.46
1.5	20	0.964	0.960	0.993	0.993	38.46	35.83
2.0	11	0.974	0.973	0.995	0.995	38.23	36.19

Table 5.5: Comparison of the non-parametric models, following the same structure as Table 5.4.

(a) JPEG AI.

Rate	**#**	**SSIM**		**MSSSIM**		**PSNR**	
		Standard	**Proposed**	**Standard**	**Proposed**	**Standard**	**Proposed**
0.06	3	0.679	0.664	0.866	0.861	27.65	26.94
0.12	5	0.734	0.699	0.904	0.901	24.68	23.80
0.25	1	0.705	0.697	0.917	0.915	31.156	30.54
0.5	4	0.853	0.860	0.960	0.963	30.82	29.82
0.75	6	0.934	0.925	0.985	0.982	35.07	32.53
1.0	5	0.940	0.944	0.988	0.988	34.23	32.56
1.5	15	0.964	0.960	0.992	0.991	37.38	35.25
2.0	10	0.970	0.972	0.994	0.994	37.64	36.70

(b) Kodak.

Rate	**#**	**SSIM**		**MSSSIM**		**PSNR**	
		Standard	**Proposed**	**Standard**	**Proposed**	**Standard**	**Proposed**
0.06	2	0.721	0.702	0.881	0.875	27.62	26.87
0.12	3	0.513	0.500	0.826	0.842	22.82	22.21
0.25	1	0.912	0.887	0.974	0.965	34.58	30.58
0.5	5	0.837	0.845	0.960	0.965	30.37	29.31
0.75	9	0.920	0.921	0.984	0.984	34.99	32.98
1.0	11	0.932	0.935	0.986	0.987	36.78	34.88
1.5	21	0.962	0.958	0.993	0.992	38.75	36.44
2.0	16	0.968	0.973	0.994	0.994	37.87	37.39

Table 5.6: The mean rate obtained by the quantized models.

Target Rate	Parametric Model	Non-parametric Model
0.06	0.08	0.07
0.12	0.116	0.180
0.25	0.246	0.260
0.5	0.531	0.512
0.75	0.752	0.757
1.0	0.989	1.013
1.5	1.511	1.516
2.0	2.020	2.031

5.5.6 Quantization-Based Architecture to Overcome the Mean Rate Shift Problem

The mean rate shift problem was analyzed theoretically in Section 5.4.4. Even though it presents a potential reason for the shift problem, it does not follow a deductive approach, which does not guarantee validity. Therefore, a small and simple experiment was devised to test the impact of eliminating the additive uniform noise of the architectures. Instead of adopting this approximation, it followed a simple rule of applying the quantization directly on training and ignoring the gradients of this operation, making its derivative have a value of 1. Therefore, the effect is to ignore the quantization non-differentiability problem in the neural network.

The training setup followed the ones of Section 5.5.1. A set of neural networks was obtained adopting the loss parameter to be equal to the desired rate, $R^{param} = r^{target}$. So, this strategy considered the match of the real rate and the estimated loss paramater, and did not apply the heuristics described in Section 5.4.5. The results of the mean rates obtained in the Kodak dataset can be seen in Table 5.6. It is essential to highlight that the exact value of β was adopted for these models, as the focus was not on observing this perfect tuning of the models, only if the mismatch would occur. Nevertheless, these models exhibited more variation, which would require a higher value for the β constant.

The main focus of observation here is that the high shift, especially in the parametric model, did not occur. As a rewind, the parametric model required a $R^{param} = 0.254$ to attain a rate $r^{target} = 0.06$, while a value $R^{param} = 0.410$ was required to achieve a rate of $r^{target} = 0.25$. Therefore, it is strong evidence, along with the theoretical derivations of Section 5.4.4, for the additive uniform noise to cause the mean rate shift problem.

Another point is that as the quantized models seem to not suffer from the mean rate shift problem, they could be used instead of the additive uniform noise, requiring only a search for the parameters of β. However, they produced inferior results, mainly in the low

Table 5.7: Table showing the results of the quantized model experiment.

Target Rate	Parametric Model			Non-parametric Model		
	PSNR	SSIM	MSSSIM	PSNR	SSIM	MSSSIM
0.06	19.787	0.456	0.595	16.539	0.409	0.437
0.12	22.363	0.562	0.755	19.271	0.460	0.542
0.25	25.150	0.701	0.882	23.515	0.600	0.793
0.5	29.642	0.854	0.959	29.589	0.850	0.957
0.75	31.728	0.901	0.976	31.866	0.901	0.975
1.0	33.358	0.929	0.984	33.634	0.931	0.984
1.5	36.087	0.960	0.992	36.366	0.960	0.992
2.0	38.463	0.976	0.995	38.616	0.975	0.995

rates, compared with the additive uniform noise equivalent, as shown in Table 5.7. Even though they have comparable results in higher rates, these rates are specifically where the mean rate shift problem does not occur strongly. This set of observations makes these models useless for the primary strategy: obtaining a general model and training approach. Nevertheless, the study and tuning of these models, which do not use proxies for the quantization operation, can be the object of future work. Section 3.3.9 showed some approaches that study the quantization problem.

The results seen in Table 5.7 can be seen in Image 5.13, where a comparison of a reconstruction from the models is observed. It is clear the heavy quality deterioration suffered by the quantized model, which is expected from the metrics shown in Table 5.7. Therefore, the quantized model does not suffer from the mean rate shift problem but considerably degrades at low rates. Also, this small experiment gives some evidence to the mean rate shift problem theoretically explained in Section 5.4.4.

5.5.7 Application in Different Architectures

One of the goals of this proposal is that the modified loss can be applied to different architectures. The baseline models selected were the seminal works in the area, named Balle's parametric and non-parametric approaches [45, 23]. Nevertheless, to test the extensiveness of the approach not only through architectures that explore different entropy models but also through different neural operations, a simple experiment was devised on a non-parametric entropy approach which, instead of pairs of convolution + GDN, has successive residual layers, widely used in the field of deep learning, and initially proposed in [128] to solve the problem of gradients vanishing/exploding in deep architectures.

Therefore, instead of using the layers of convolutions followed by GDNs, an alternative transform that uses the same number of layers but adopts the residual blocks is imple-

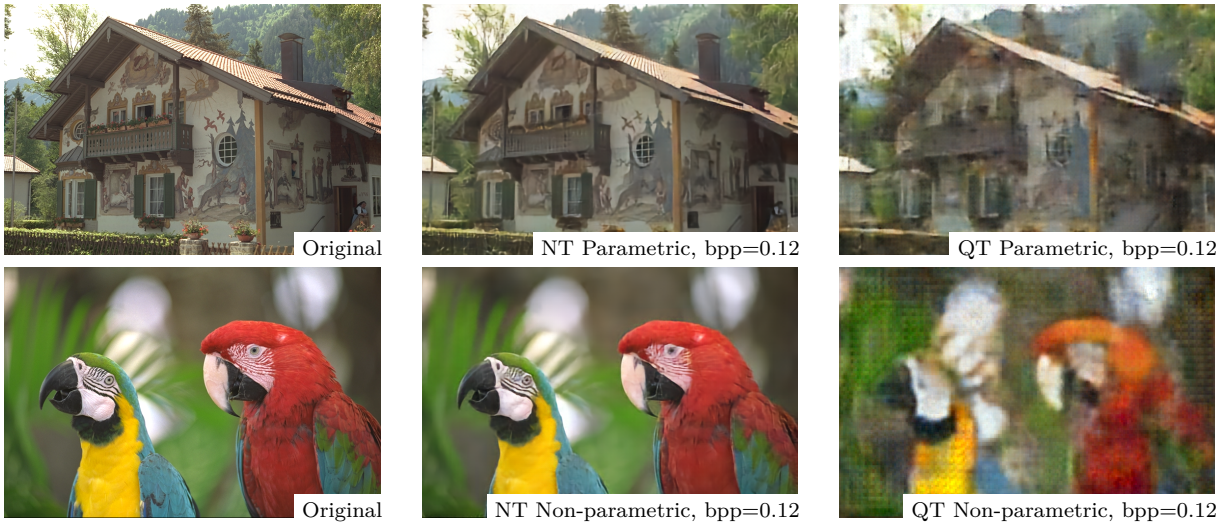


Figure 5.13: Reconstructions obtained using the **parametric** and **non-parametric** models for images from the Kodak dataset. These images compare the target model, which applied the uniform noise (NT), and the target model, which directly applies the quantization on training (QT). It is possible to see a great deterioration of the images, something already seen by the metrics exposed in Table 5.7.

Table 5.8: The parameters $\{R_t, \beta\}$ adopted in the residual layers transform for the non-parametric models evaluated.

r	R_t	β
0.06	0.065	55000
0.75	0.77	6000
2.0	2.0	1500

mented. A residual layer can be seen in Figure 5.14. The layers were linear, removing the last ReLU activations on the last layer of the analysis and the synthesis transform. It is the same way it is done with the GDN convolutional layers. One hundred twenty-eight filters were adopted for this experiment.

As this experiment was devised only to see if a different transform would converge to restrict the rate of the entropy model, the complete heuristic was not applied here. The β values were set the same as the GDN non-parametric transform, using the values of Table 5.3a. There were only some adjustments in the R_t constant to converge the mean rate to the desired rate. Also, since it is an experiment with a focus on testing the extensiveness of the approach, it was trained only for some target rates $r = \{0.06, 0.75, 2.0\}$, which are the extreme rates explored in this work, besides an intermediate rate. The parameters for the loss are shown in Table 5.8.

The rate results for this experiment are shown in Table 5.9 considering both validation datasets in the three specified rates. The rate's mean, minimum, maximum, and standard

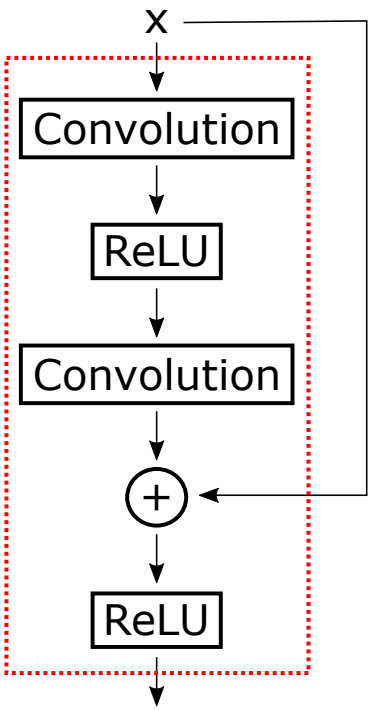


Figure 5.14: The building block of the residual layer. The dashed lines represent a single residual layer.

deviation statistics are presented. These results show that the approach can still achieve the desired bitrate even using a completely different network architecture. The standard deviation in each case was minimal, suggesting that lower betas could be used, which, in turn, would probably improve the mean PSNR. However, tuning ResNet-like architectures was not the aim of this experiment. The PSRN was only presented to show the monotonic behavior of the quality metric while increasing the rate, which is the expected behavior. Therefore, experiments with these different types of transforms will not be considered for the RD performance comparisons and may be an object of future works.

Table 5.9: The results of the residual layers experiment considering the non-parametric model for the specified target rate. It is possible to see that with the parameters specified by Table 5.8, it is possible to reach a good convergence on the rate on both datasets. The PSRN quality measure shows a monotonic behavior of the quality when increasing the rate, which is the expected behavior.

Target Rate	Kodak					JPEG AI				
	Rate				Mean PSNR	Rate				Mean PSNR
	Mean	Min	Max	σ		Mean	Min	Max	σ	
0.06	0.059	0.057	0.061	0.0008	20.77	0.059	0.057	0.067	0.0012	19.32
0.75	0.742	0.721	0.798	0.0206	31.77	0.749	0.719	0.822	0.0231	30.89
2.0	2.004	1.965	2.123	0.0423	34.95	2.030	1.958	2.208	0.0563	33.26

5.5.8 Subjective Evaluation

Figures 5.15 and 5.19 display some reconstructions using the standard parametric architecture for each database. Each row represents one of the images whose reconstruction fell within the tolerance range for both the standard and proposed architectures in this chapter. Similarly, examples of non-parametric models are showcased in Figures 5.17 and 5.21.

It is noteworthy that in some cases, the proposed model yields better perceptual results than the standard architecture: the second row in Figure 5.15 and the first row in Figure 5.17. The standard architecture seems slightly superior in the first row of Figure 5.19. Meanwhile, the decoded images have perceptible distortions in different locations in the first row of Figure 5.21. However, the reconstructions have little to no perceptual difference for most bit rates. Thus, the proposed method does not introduce perceptible artifacts differing from the reference models.

Rate-distortion curves for all valid metrics for each of the selected images for the parametric models can be seen in Figures 5.16 and 5.20 for each database, respectively. The rate-distortion curves for the pair of non-parametric models are displayed in Figures 5.18 and 5.22. As can be observed, although there is a loss in terms of PSNR, SSIM, and MSSSIM, the proposed method exhibits rate-distortion results that are pretty competitive with the standard models. It even outperforms the standard models for some images at medium and low rates. It is worth noting that a significant drop in PSNR for restricted rate coding is expected as it is present even in traditional CODECs, such as VVC [129].

5.6 Conclusions

This chapter introduces an approach for rate-constrained coding in neural networks, an open problem in CODECs of this type. A loss function parameterized by $\{\beta, R^{param}\}$ was proposed, considering the average target rate and a desired variance. A heuristic was proposed to estimate the values of $\{\beta, R^{param}\}$, making the methodology generic and applicable to various architectures. Rate control is a desirable feature in practical scenarios, and the proposed method can aid in introducing a neural image CODEC in real applications.

Like any image encoder, the rate-distortion performance is affected when introducing a rate control mechanism. Nonetheless, the proposed method exhibits minimal losses with the MSSSIM and SSIM metrics. For some images, a rate-distortion gain can be achieved at medium to low rates in these metrics. Additionally, from the visual examples, it is evident that there are no significant perceptual differences in the reconstructions compared to the standard approaches.

The design of the rate-oriented training also highlighted the rate misalignment. The mismatch may be caused by the approximation of the quantization operation during training time, not excluding other possible unexplored causes. This misalignment became evident with the average rate deviation issue, inspiring the adoption of heuristics to estimate the loss function parameters. Empirical tests and theoretical analysis pointed out that the misalignment between the estimated rate during training and the actual rate during inference is related to additive uniform noise. Other causes may influence this behavior, but in this work, only this perspective is considered and analyzed. This scenario may result in a rate overestimation and, consequently, an over-penalization during training, shifting the actual rate at inference time below the expected value.

The rate-oriented approach could be improved. For instance, a more detailed study of the impact of different loss functions on rate-distortion performance could be conducted. This loss analysis might inspire architectures with minimal deterioration compared to reference architectures.

Different quantization strategies could also be explored to mitigate or even avoid the problem of average rate deviation. Obtaining efficient approximations for quantization is, in itself, an open problem in the field. However, the main benefit of this chapter's proposal is aligning the loss function hyper-parameters with the actual values obtained later during inference, thus reducing the need for a heuristic for parameter estimation.

Another line of study relates to the hypothesis of a correlation between performance deterioration and the problem of average rate deviation. Future work investigating this hypothesis in rate-oriented or standard approaches could provide insights into aspects related to quantization, for example. If there is indeed a correlation, exploring quantization strategies using average rate deviation as a reference could lead to better approximations of quantization and, consequently, enhance the overall performance of neural compression.

Applying the proposed approach to architectures with multiple target rates could also constitute interesting future work. Instead of obtaining one model representing multiple arbitrary points in the convex case, achieving multiple desired rates with this single model would be possible. Considering all that has been presented in this chapter, the next chapter will address the main issues of this work with an improved and clever proposal for the same problems outlined in the current chapter.

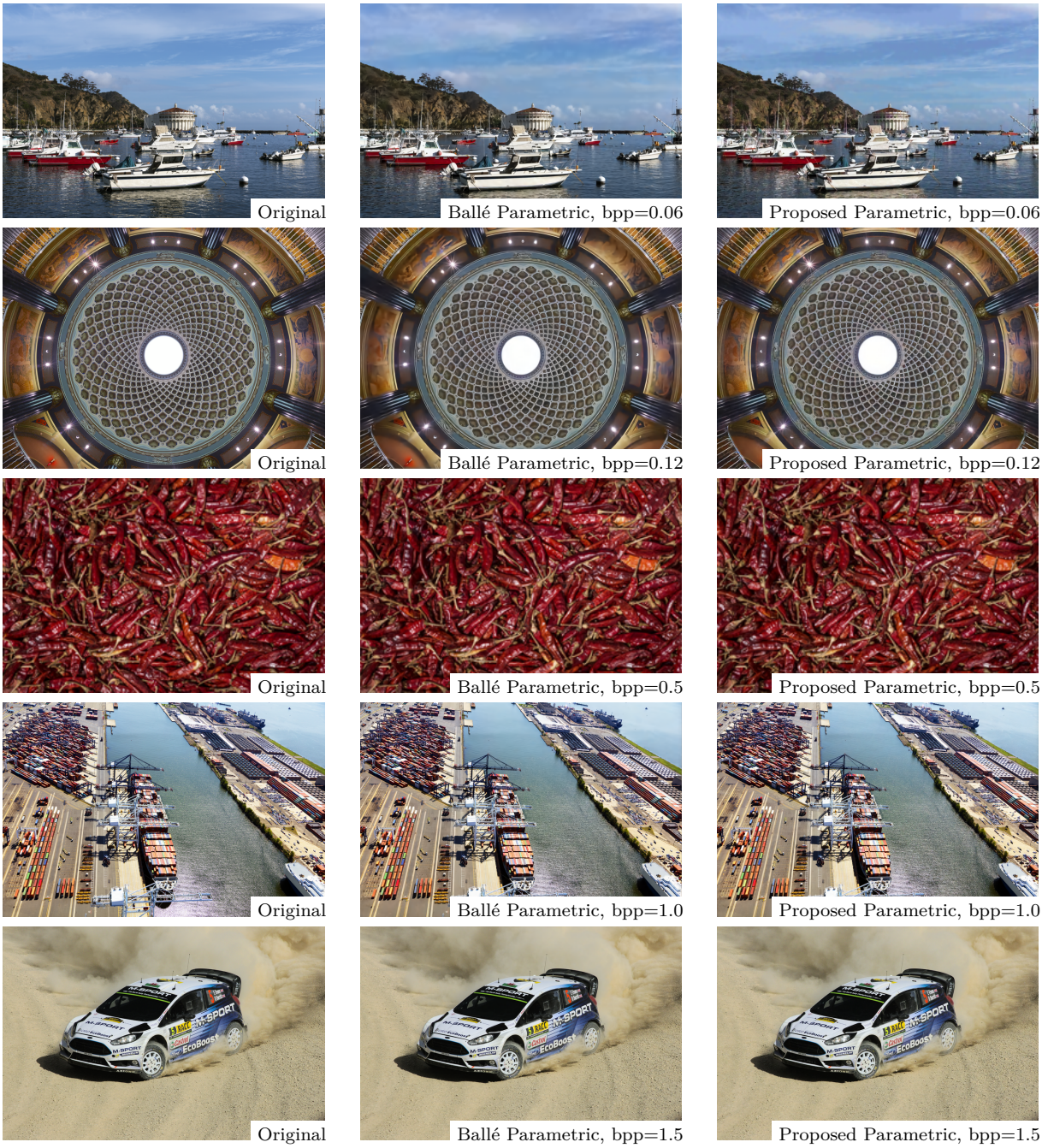


Figure 5.15: Reconstructions obtained using the **parametric** models for images from the JPEG AI database. Original (on the left), Ballé Parametric (in the middle), and Proposed Parametric (on the right). The rates considered for the reconstructions are $\{0.06, 0.12, 0.5, 1.0, 1.5\}$, each representing a row of images.

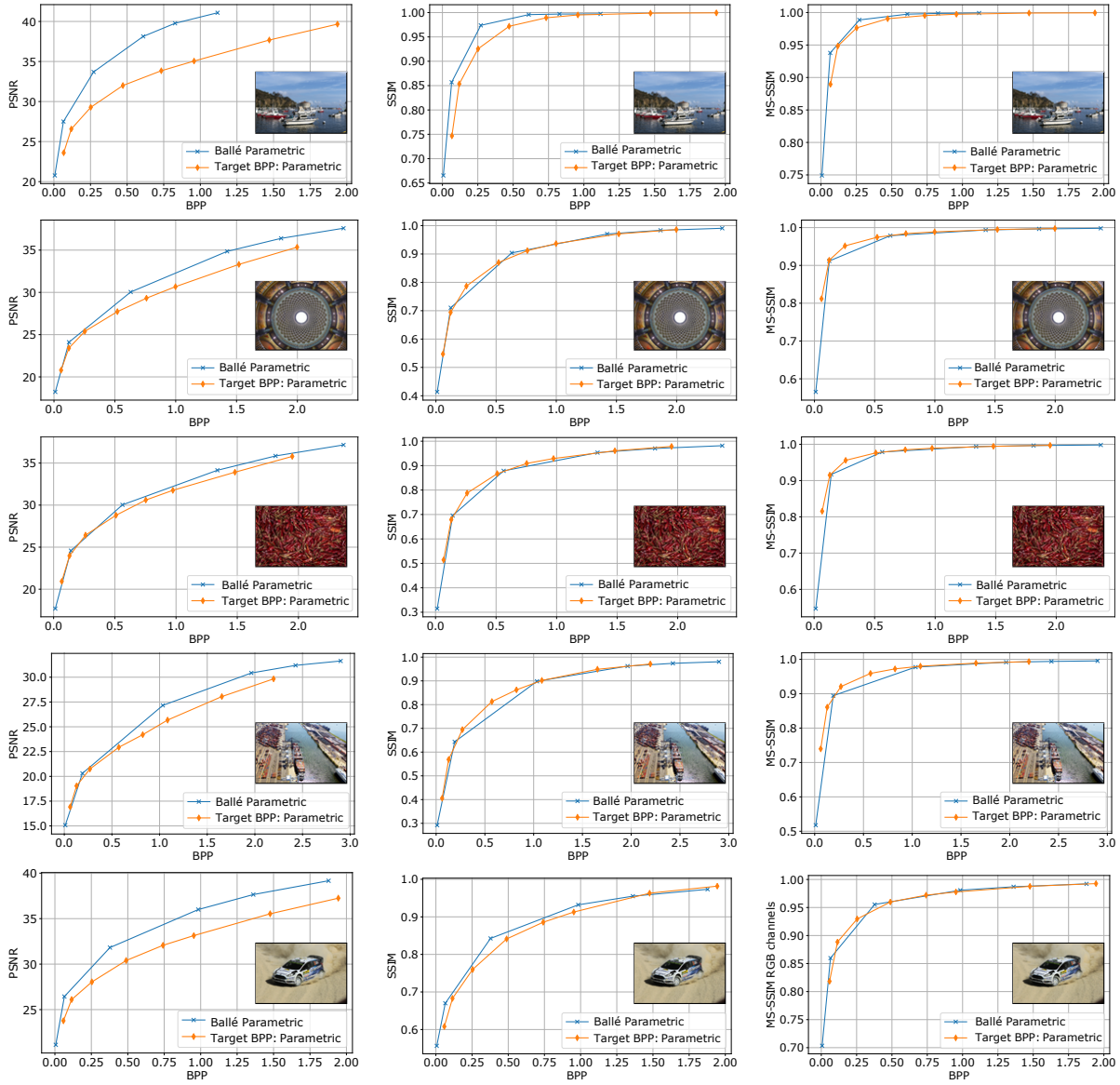


Figure 5.16: Rate-distortion curves for the **parametric** models, of the reconstructions in Figure 5.15, utilizing the metrics PSNR (on the left), SSIM (in the middle), and MS-SSIM (on the right).

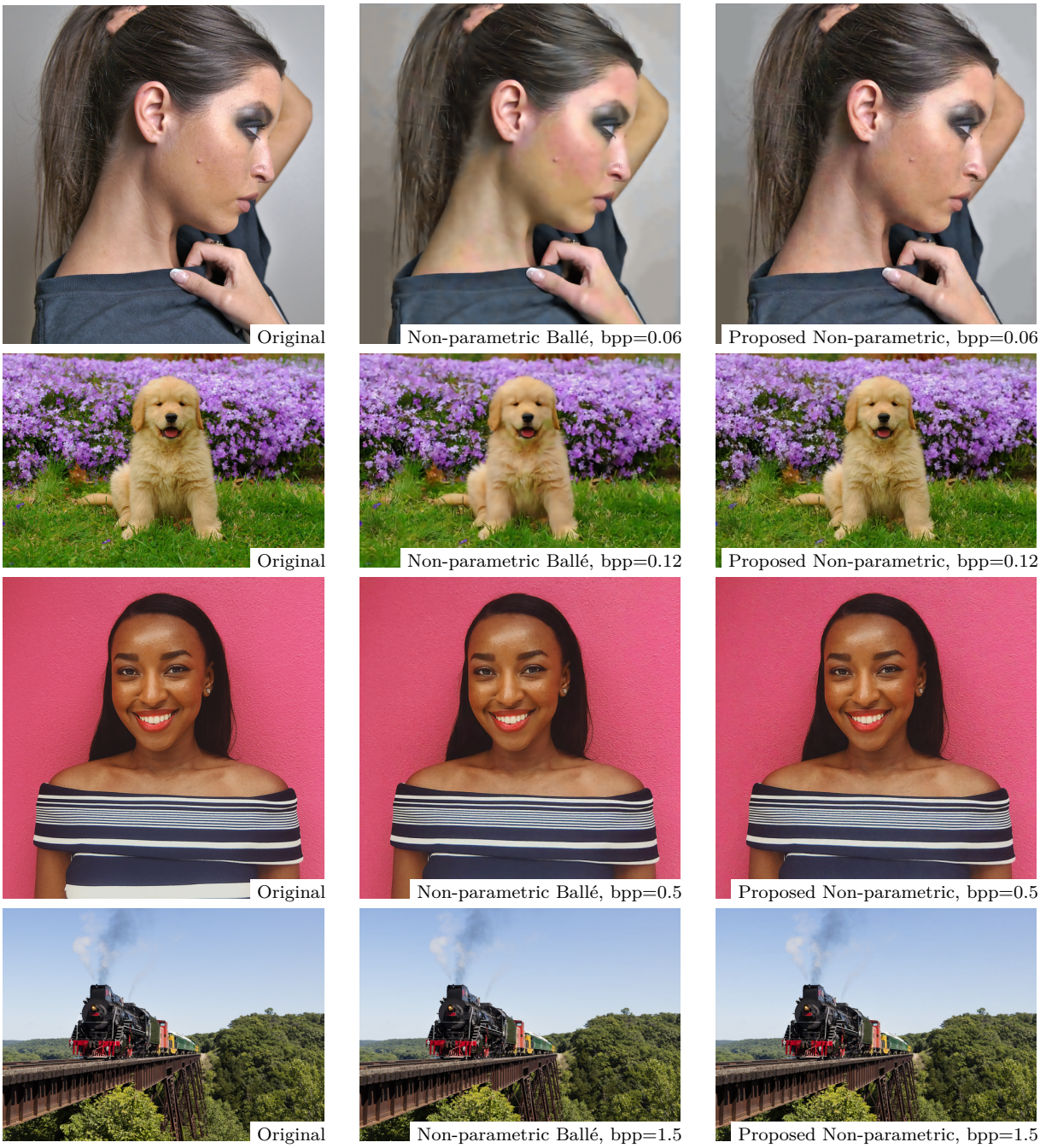


Figure 5.17: Reconstructions obtained using the **non-parametric** models for images from the JPEG AI database. Original (on the left), parametric Ballé (in the middle), and proposed parametric (on the right). The rates considered for the reconstructions are $\{0.06, 0.12, 0.5, 1.5\}$, each representing a row of images.

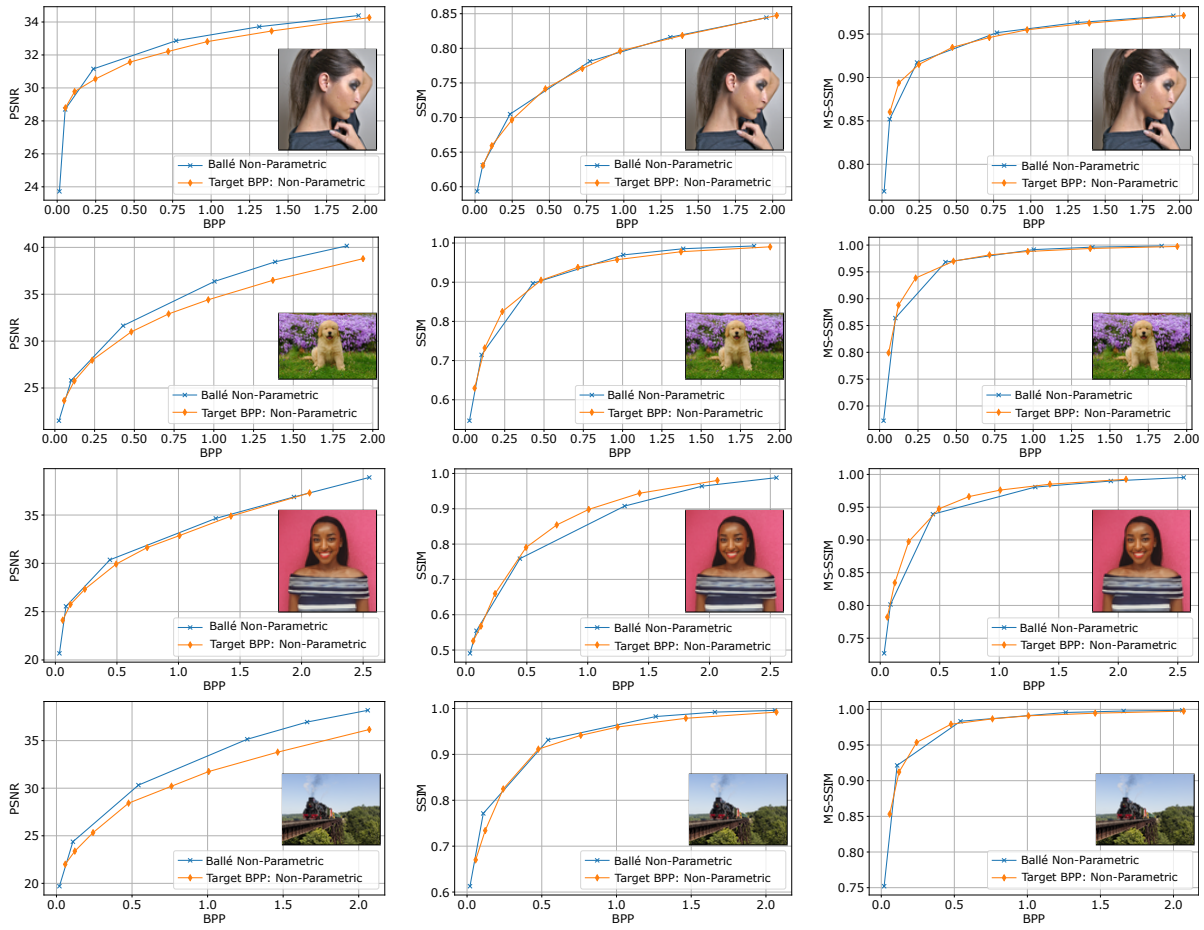


Figure 5.18: Rate-distortion curves for the **non-parametric** models, corresponding to the reconstructions in Figure 5.17, using the PSNR (left), SSIM (middle), and MS-SSIM (right) metrics.

Figure 5.19: Reconstructions obtained using the **parametric** models for images from the Kodak dataset. Original (on the left), Ballé Parametric (in the middle), and Proposed Non-parametric (on the right). The considered rates for the reconstructions are $\{0.12, 0.25, 0.5, 0.75, 1.5\}$, each representing a row of images.

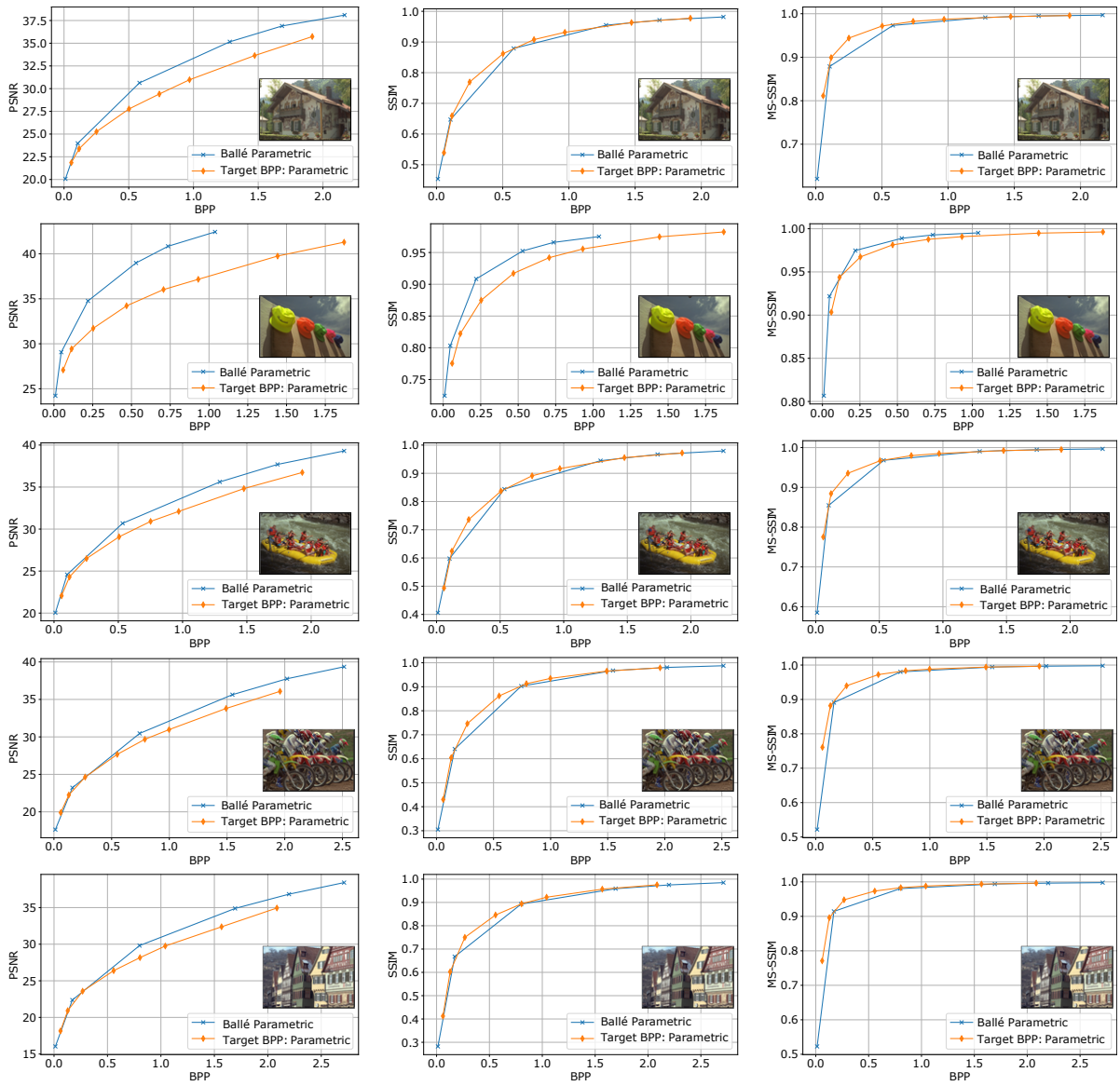


Figure 5.20: Rate-distortion curves for the **parametric** models, relating to the reconstructions in Figure 5.19, using the PSNR metrics (on the left), SSIM (in the middle), and MS-SSIM (on the right).

Figure 5.21: Reconstructions obtained using the **non-parametric** models for images from the Kodak dataset. Original (left), Non-parametric Ballé (center), and Proposed Non-parametric (right). The considered bit rates for the reconstructions are $\{0.12, 0.25, 0.5, 0.75, 1.5\}$, each representing a row of images.

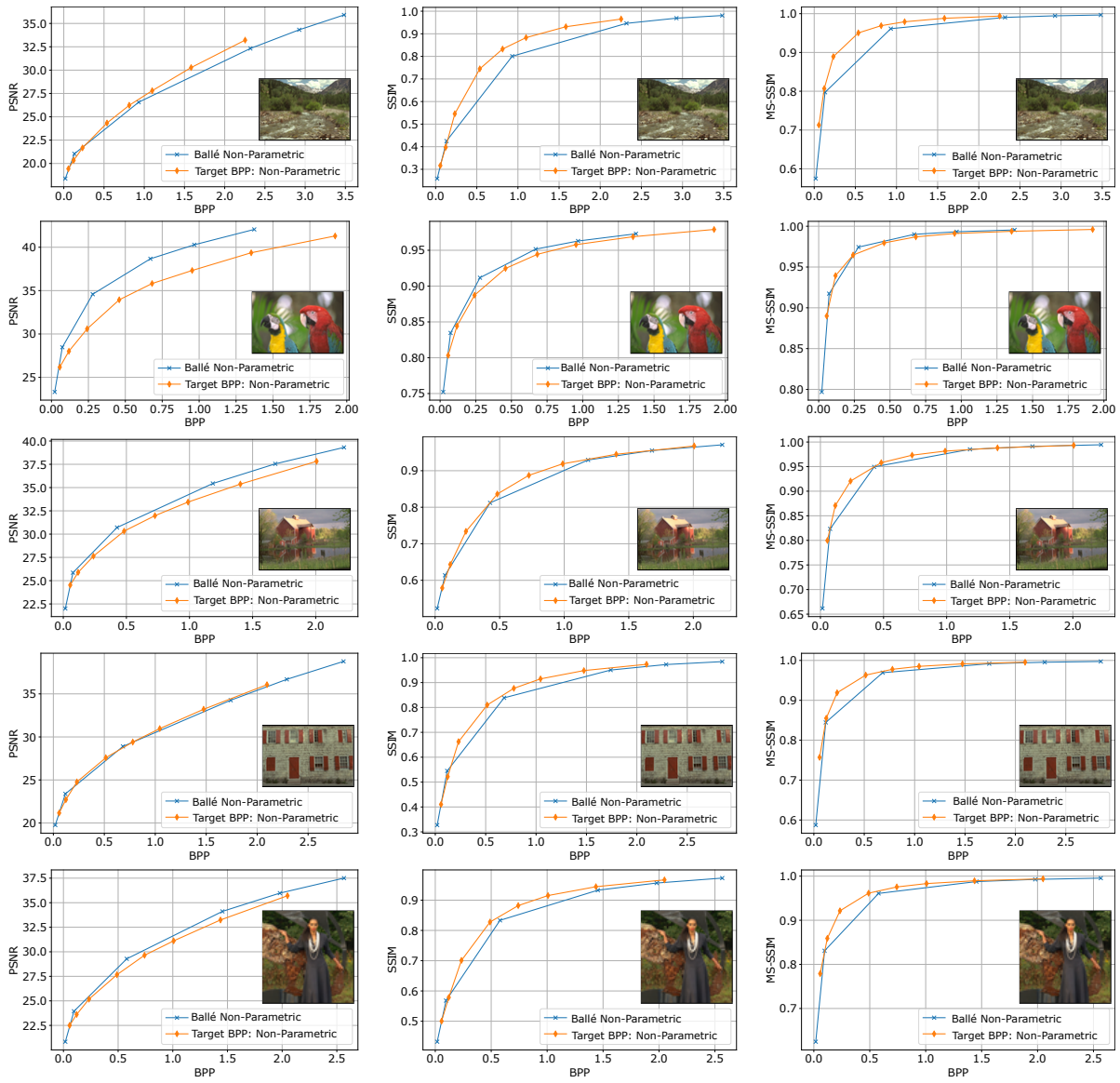


Figure 5.22: Rate-distortion curves for the **non-parametric** models, relating to the reconstructions in Figure 5.21, using the PSNR (on the left), SSIM (in the middle), and MS-SSIM (on the right) metrics.

Chapter 6

Learning-Based Image Compression with Parameter-Adaptive Rate-Constrained Loss

This chapter details the adoption of Reinforcement Learning as a solution for achieving target bitrate control in neural-based compression approaches. This idea generalizes the earlier proposal. Section 6.1 will review the main ideas from the last chapter, motivating the current work. Additionally, the concept of Reinforcement Learning in neural training will be introduced to serve as a foundation for later sections.

Section 6.2 will present the generalized loss proposed in this chapter. The reader will be guided through the interpretation of training as a time evolution process, leading to the derivation and expanded analysis of the loss. Section 6.3 will detail the experiments using this technique and provide an ablation study. The notations adopted in this chapter are consistent with Chapter 5, as described in Section 5.1.

6.1 Problem Statement

The mean rate shift problem attributes the rate discrepancy to using additive uniform noise as a differential proxy for the quantization process. When an input $\mathbf{x}$ is fed into this network, it produces a raw latent vector $\mathbf{y}$. During training, additive uniform noise is incorporated into the latent components. The initial strategy to address the rate shift problem involved alternating training and validation steps. During training, the noisy latent space was considered, while during validation, the quantization operation was applied. By swiftly converging to the desired rate, a relatively small number of iterations enabled the network output to stabilize, facilitating the measurement of shifts and adjustments to the mean-rate hyperparameter R^{param} of the loss function [48].

This heuristic method effectively rectified the mean-shift mismatch attributed to the noisy latent space [48]. However, it necessitated numerous alternating training and validation steps. Despite setting the training iterations to $50,000$, achieving the desired rate convergence of the hyperparameter R^{param} often required many alternating steps. Subsequently, a complete training of $500,000$ iterations was conducted using the corrected R^{param}. This approach, reliant on neural network training, incurs high computational costs, prompting the exploration of more efficient solutions to address this challenge.

6.1.1 Reinforcement Learning Modeling for the Rate-Constrained Model

A relationship between reinforcement learning and the target-rate loss can be inferred from iterative updates, extending Guerin's work [48]. The primary target loss (Equation 5.14) comprises two hyperparameters $\{\beta, R^{param}\}$, which undergo iterative updates through the alternating steps of the heuristic outlined in Algorithm 3.

The parameter search heuristic resembles the basic concept of reinforcement learning, as depicted in Section 2.2.8. Therefore, it suggests viewing neural network training as a temporal adaptive process. As iterations progress, the neural network (the Agent) better understands the data (the Environment). This interpretation implies that the neural network can increasingly correct the rate mismatch problem during continuous training. This process implicitly reflects the notion of a policy in reinforcement learning. As stated in Section 3.3.10, few works in supervised/unsupervised learning paradigms interpret neural network training as a temporal process with updated values that were otherwise constants in canonical approaches. However, none of these works are close to the interpretation proposed here.

6.2 Parameter-Adaptive Target-Rate Loss

This section details the approach devised to overcome the mean-rate mismatch without using the costly heuristic. The idea is to devise a time-adaptive target-rate loss that constantly corrects the rate estimation mismatch. The entropy modeling introduced in [45, 23] is adopted as described in the following sections.

6.2.1 Training as a Temporal Process

The basic setup of the reinforcement learning paradigm is to model the environment as a Markov Decision Process, which analyzes the temporal evolution of the training. This idea can be interpreted by viewing the iterations of neural network training as a temporal

evolution process. To this proposal, it is necessary to generalize the notations adopted in Chapter 5.

Let $\mathbf{x}_t$ denote the input to the neural network at time t. Time now stands for the iterations of the neural network. For example, the basic setup training of models proposed in Chapter 5 was a total of 500000 iterations after the heuristic search stage. Now, each of these iterations will be indexed by the time variable. Therefore, $\mathbf{x}_0$ stands for the input of the neural network at the first iteration.

Analogously, $\tilde{\mathbf{x}}_t$ and $\hat{\mathbf{x}}_t$ define the outputs of the neural networks at time t using the notations of the noisy approximation of the latent space and the quantized latent space, respectively. This notation extends to the latent space variables, with $\tilde{\mathbf{y}}_t$ and $\hat{\mathbf{y}}_t$ representing the noisy and quantized latent space at time t, respectively.

6.2.2 Latent Rate Estimation

The estimations for the rate remain the same, but now adopting the subscript t. For simplicity, the index t will be used only in core values related to the time adaptation proposed in this approach. This will help keep the notation concise and not overly complex. Given the entropy model $p_{\tilde{\mathbf{y}}}$, which will not have the t subscript, it is possible to estimate the rate using Shannon's entropy:

$$R_t^{est}(\tilde{\mathbf{y}}) = -\mathbb{E}_{q_{\tilde{\mathbf{y}}}}\left[\log p_{\tilde{\mathbf{y}}}(\tilde{\mathbf{y}})\right] = -\sum_i^n q_{\tilde{y}_i}(\tilde{y}_i)\log p_{\tilde{y}_i}(\tilde{y}_i) \tag{6.1}$$

where $R_t^{est}(\tilde{\mathbf{y}})$ is the estimated rate of the noisy latent at time t, $\mathbb{E}_{p_{\tilde{\mathbf{y}}}}$ represents the expectation, and n is the dimensionality of the latent space. Equation 6.1 is obtained considering the factorization of the distribution $p_{\tilde{\mathbf{y}}}$, where $\tilde{\mathbf{y}} = [\tilde{y}_i]_{i=1}^n$ and $p_{\tilde{\mathbf{y}}} = [p_{\tilde{y}_i}]_{i=1}^n$. Analogously, it considers the factorization of the variational distribution $\boldsymbol{q}$. The rate can also evaluate for $\hat{\mathbf{y}}$:

$$R_t^{est}(\hat{\mathbf{y}}) = -\mathbb{E}_{q_{\hat{\mathbf{y}}}}\left[\log p_{\hat{\mathbf{y}}}(\hat{\mathbf{y}})\right] = -\sum_i^n q_{\hat{y}_i}(\hat{y}_i)\log p_{\hat{y}_i}(\hat{y}_i) \tag{6.2}$$

where $R_t^{est}(\hat{\mathbf{y}})$ is the estimated rate for the quantized latent $\hat{\mathbf{y}}$ at time t, and $\hat{\mathbf{y}} = [\hat{y}_i]_{i=1}^n$. Note that the distribution is the noisy distribution obtained in the training of the neural network, as it is used in the evaluation time, where quantization is adopted. These two rate estimations will be central to generalizing the target rate loss of Equation 5.14.

6.2.3 The Mean-shift Function

As explained earlier, the basic idea is inspired by the reinforcement learning process of iterative updates of some parameters that improve the agent's behavior. The fundamental equation for this idea is called the mean-shift function. This function aims to measure the mismatch occurring at time t. It is defined as follows:

$$R_t^{shift}(\mathbf{y}) = R_t^{est}(\tilde{\mathbf{y}}) - R_t^{est}(\hat{\mathbf{y}}) \tag{6.3}$$

To simplify the notation, R_t^{shift} is posed as a function of $\mathbf{y}$ since its definition uses both $\tilde{\mathbf{y}}$ and $\hat{\mathbf{y}}$. Nevertheless, both terms can be considered as $\mathbf{y}$ functions. Equation 6.3 directly measures the current mismatch between the noisy estimation $R_t^{est}(\tilde{\mathbf{y}})$ and the quantized estimation $R_t^{est}(\hat{\mathbf{y}})$. It does not yet incorporate the temporal dependency, which will be constructed in the following sections.

6.2.4 Time-adaptive Rate Parameter

The next step in generalizing the work of [48] involves modifying the loss rate parameter. Initially, after the search heuristics, R^{param} was a fixed constant during the "real" training after the heuristic. Now, it will be changed to a temporal function that accounts for the shift estimated by Equation 6.3.

Considering the desired rate r^{target}, the target parameter related to the control of the mean rate obtained by the neural network can be given as:

$$R_t^{param}(\mathbf{y}; r^{target}) = r^{target} + R_{t-1}^{shift}(\mathbf{y}) \tag{6.4}$$

where R_{t-1}^{shift} is the rate mismatch occurring at training time $t-1$. If $t = 0$, then $R_{t-1}^{shift} = 0$. Therefore, the rate loss parameter R_t^{param}, in the t-th iteration, is based on information from past iterations, namely the $(t-1)$-th iteration, which structures a temporal expansion, as follows:

$$\begin{aligned} R_t^{param}(\mathbf{y}; r^{target}) &= r^{target} + R_{t-1}^{shift}(\mathbf{y}) \\ R_t^{param}(\mathbf{y}; r^{target}) &= r^{target} + \Big(R_{t-1}^{est}(\tilde{\mathbf{y}}) - R_{t-1}^{est}(\hat{\mathbf{y}})\Big) \end{aligned} \tag{6.5}$$

The fundamental piece of this new approach relies on this change. R^{param}, once a constant, is generalized to $R_t^{param}(\mathbf{y}; r^{target})$, which is a function of the latent space at the past time, specifically the past mismatch estimation. It should be clear that more robust corrections relying on more past information can be applied here. However, for simplicity, this basic setup relying on the $(t-1)$-th iteration showed promising results. Future work

could refine this iterative correction process to account for more helpful information about the current estimations of the neural network regarding the mismatch.

When applying gradient descent optimization, it is assumed that $R_t^{param}(\mathbf{y}; r^{target})$ is constant. This approach is justified since, from the perspective of the target rate loss, the rate parameter can be viewed as a conventional constant hyperparameter. Empirical tests have shown significant stability caused by this change. This simple convention makes training far more stable and reliable.

6.2.5 The Time-Adaptive Target-Rate Loss

Considering the steps developed in the earlier sections, it is possible to derive the full rate-parameter time-adaptive loss, generalizing the heuristic and fixed rate-parameter loss proposed in [48]. This new loss, using the t indexing, is given as:

$$J_t(\mathbf{x}, \tilde{\mathbf{x}}) = D_t(\mathbf{x}, \tilde{\mathbf{x}}) + \beta f_t(\mathbf{y}; r^{target}) \tag{6.6}$$

where $D_t(\mathbf{x}, \tilde{\mathbf{x}})$ denotes the distortion measured by the mean squared error, and β represents the fixed variance hyperparameter proposed in [48], which, in the current step of the proposal, remains a conventional constant hyperparameter. Expanding the $f_t(\mathbf{y}; r^{target})$ in terms of the new notation:

$$f_t(\mathbf{y}; r^{target}) = \left(\frac{R_t^{param}(\mathbf{y}; r^{target}) - R_t^{est}(\tilde{\mathbf{y}})}{R_t^{param}(\mathbf{y}; r^{target})} \right)^2 \tag{6.7}$$

The Equation 6.7 can be applied in Equation 6.6 to get the full view of the equation of the loss:

$$J_t(\mathbf{x}, \tilde{\mathbf{x}}) = D_t(\mathbf{x}, \tilde{\mathbf{x}}) + \beta \left(\frac{R_t^{param}(\mathbf{y}; r^{target}) - R_t^{est}(\tilde{\mathbf{y}})}{R_t^{param}(\mathbf{y}; r^{target})} \right)^2 \tag{6.8}$$

$$J_t(\mathbf{x}, \tilde{\mathbf{x}}) = D_t(\mathbf{x}, \tilde{\mathbf{x}}) + \beta \left(\frac{r^{target} + \left(R_{t-1}^{est}(\tilde{\mathbf{y}}) - R_{t-1}^{est}(\hat{\mathbf{y}}) \right) - R_t^{est}(\tilde{\mathbf{y}})}{r^{target} + (R_{t-1}^{est}(\tilde{\mathbf{y}}) - R_{t-1}^{est}(\hat{\mathbf{y}}))} \right)^2$$

Several estimation functions are involved in this generalization, related to the iterative corrections of the time-adaptive rate parameter. Figure 6.1 illustrates the training flow of this neural network. The normal mapping from $\mathbf{x}$ to $\tilde{\mathbf{x}}$ yields $\mathbf{y}$, the noisy $\tilde{\mathbf{y}}$, and the quantized $\hat{\mathbf{y}}$. Both $\tilde{\mathbf{y}}$ and $\hat{\mathbf{y}}$ have their entropy estimated by the current entropy model. This entropy model, shown as a separate block in the figure, can be considered part of

the encoder. The estimated noisy rate and the real rate are obtained, and based on these estimates, iterative corrections of the parameters can be applied.

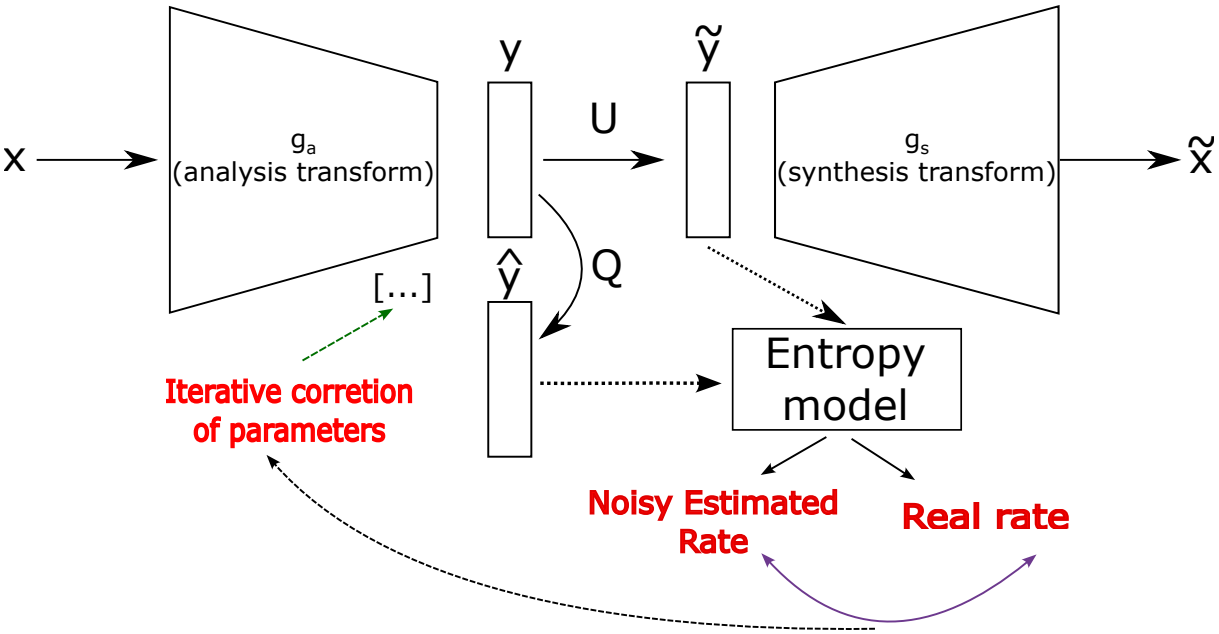


Figure 6.1: The figure depicts the schematics of the time-dynamical feedback flow of the neural network.

Analysis of the Time-Adaptive-Parameter Loss

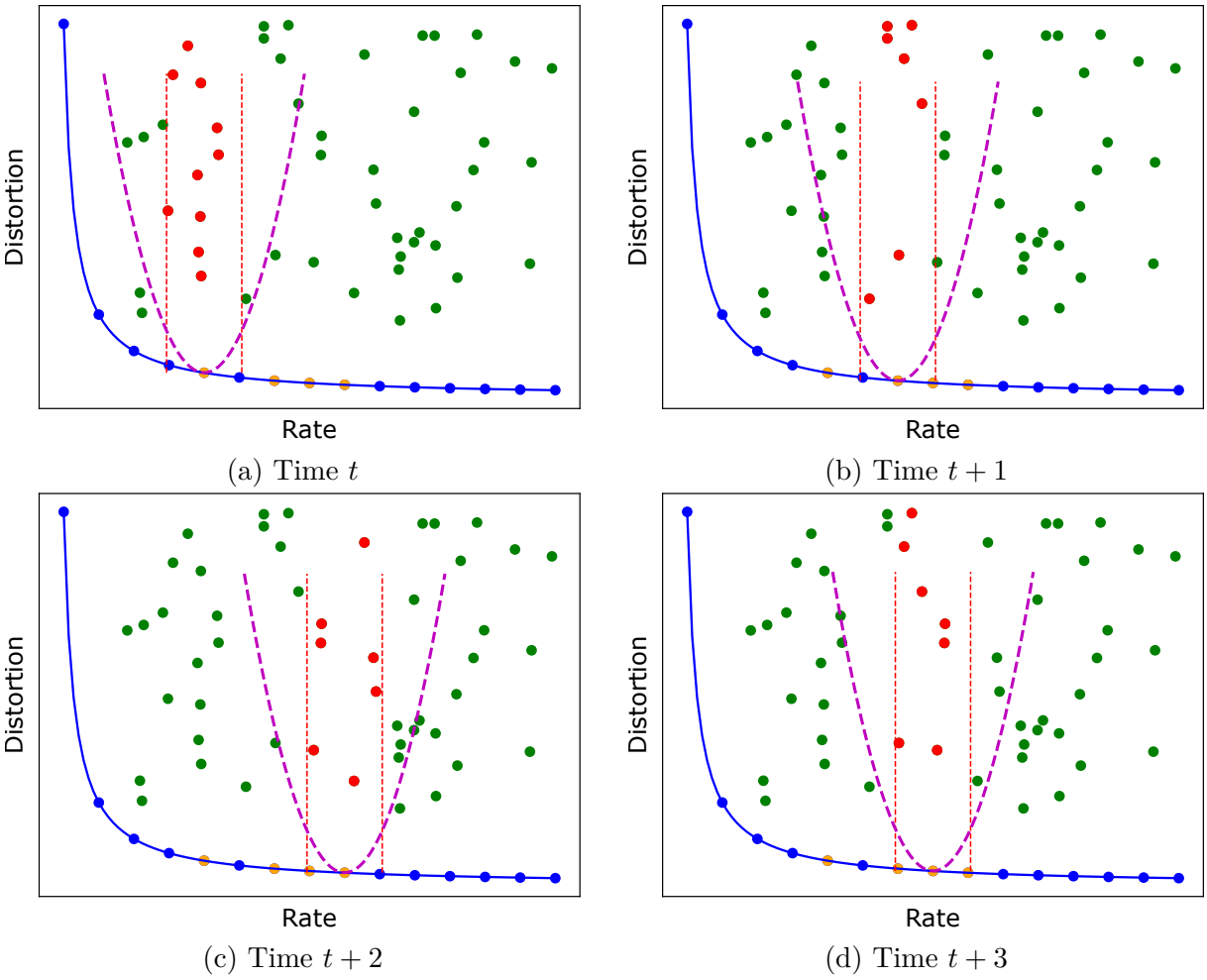
The derivations of the new loss remain closely related to the version described in Section 5.4. The proposed loss function can be expressed as follows:

$$J_t(\mathbf{x}, \tilde{\mathbf{x}}) = D_t(\mathbf{x}, \tilde{\mathbf{x}}) + \beta \frac{\left(R_t^{est}(\tilde{\mathbf{y}})\right)^2}{\left(R_t^{param}(\mathbf{y}; r^{target})\right)^2} - 2\beta \frac{R_t^{est}(\tilde{\mathbf{y}})}{R_t^{param}(\mathbf{y}; r^{target})} + \beta \tag{6.9}$$

which resembles the structure presented in Equation 5.15. When $R_t^{est}(\tilde{\mathbf{y}}) = R_t^{param}(\mathbf{y}; r^{target})$, the loss function reduces to $J_t(\mathbf{x}, \tilde{\mathbf{x}}) = D_t(\mathbf{x}, \tilde{\mathbf{x}})$. As before, only the distortion will be minimized whenever the current rate estimation meets the time-adaptive rate parameter. Following this idea, the target function induces a local minimum at $R_t^{est}(\tilde{\mathbf{y}}) \to R_t^{param}(\mathbf{y}; r^{target})$. The full function, however, has local minima influenced not only by $R_t^{est}(\tilde{\mathbf{y}}) \to R_t^{param}(\mathbf{y}; r^{target})$, but also by low values of $D_t(\mathbf{x}, \tilde{\mathbf{x}})$.

The term β still controls the width of this dynamic parabola. Even though the local minima considering only the target change as time t progresses, β remains a constant hyperparameter, keeping the width of the parabolas fixed. Therefore, higher or lower values of β will again determine the penalization for the deviation from the rate. This penalization is also dependent on r^{target} and, consequently, on $R_t^{param}(\mathbf{y}; r^{target})$. This

dependency is due to the values of $D_t(\mathbf{x}, \tilde{\mathbf{x}})$, which will be higher or lower depending on the target rates.



(a) Time t (b) Time $t+1$

(c) Time $t+2$ (d) Time $t+3$

Figure 6.2: The convex hull depicted with the parabola penalization, dynamically changing.

Figure 6.2 generalizes the idea presented in Figure 5.2, but now the schematic follows a time variation. The values of the $R_t^{param}(\mathbf{y}; r^{target})$ can oscillate as a feedback reaction to the mismatch function, trying to compensate for the noisy-caused rate shift at each time. In this scenario, $R_t^{param}(\mathbf{y}; r^{target})$ which was once a constant hyperparameter, now is a function of time and may be considered more challenging in terms of convergence of the neural network. In part, as it heavily dictates the behavior of the convergence of the neural network.

The Generalized Loss Relation with the Variational Inference

In many aspects, this section follows the corresponding Section 5.4.2. The architectures adopted are named VAE due to their equivalence with variational Bayesian inference

optimization. While this more complex function can seem complicated, the focus remains on changing the constant R^{param} to the function $R_t^{param}(\mathbf{y}; r^{target})$. This function is also considered a constant during the network backpropagation and does not contribute to the gradient descent. With this setup, it is feasible to establish an equivalence between them.

The starting point is Equation 5.16, which corresponds to the minimization of the KL-divergence between the true *a posteriori*, depicted by the function p, and the approximated *a posteriori*, represented as q. Therefore, it is possible to derive the current formulation into its expanded form:

$$J(\mathbf{x}, \tilde{\mathbf{x}}) = D(\mathbf{x}, \tilde{\mathbf{x}}) + \beta \frac{(R_t^{param}(\mathbf{y}; r^{target}))^2 - 2R_t^{param}(\mathbf{y}; r^{target})R^{est}(\tilde{\mathbf{y}}) + (R^{est}(\tilde{\mathbf{y}}))^2}{(R_t^{param}(\mathbf{y}; r^{target}))^2} \tag{6.10}$$

$$J(\mathbf{x}, \tilde{\mathbf{x}}) = D(\mathbf{x}, \tilde{\mathbf{x}}) + \beta - \frac{2\beta}{R_t^{param}(\mathbf{y}; r^{target})} R^{est}(\tilde{\mathbf{y}}) + \frac{\beta}{(R_t^{param}(\mathbf{y}; r^{target}))^2} (R^{est}(\tilde{\mathbf{y}}))^2$$

$$J(\mathbf{x}, \tilde{\mathbf{x}}) = D(\mathbf{x}, \tilde{\mathbf{x}}) + R_t^{c1}(\mathbf{y}; r^{target})R^{est}(\tilde{\mathbf{y}}) + R_t^{c2}(\mathbf{y}; r^{target})(R^{est}(\tilde{\mathbf{y}}))^2 + c_3$$

where $R_t^{c1}(\mathbf{y}; r^{target}) = \frac{2\beta}{R_t^{param}(\mathbf{y}; r^{target})}$ and $R_t^{c2}(\mathbf{y}; r^{target}) = \frac{\beta}{\left(R_t^{param}(\mathbf{y}; r^{target})\right)^2}$. These terms correspond to the $c1$ and $c2$ terms in Equation 5.18. Mathematically, they are functions that complicate the analysis. However, as stated earlier, from the optimization perspective, these are constants. Thus, the optimization factors do not relate to these new complicated functions compared to Equation 5.18. As a result, it is possible to "change" the notation to constants, as in Equation 5.18:

$$J(\mathbf{x}, \tilde{\mathbf{x}}) = D(\mathbf{x}, \tilde{\mathbf{x}}) + c_1' R^{est}(\tilde{\mathbf{y}}) + c_2'(R^{est}(\tilde{\mathbf{y}}))^2 + c_3 \tag{6.11}$$

where c_1' stands for c_1 and c_2' replaces c_2 in the form of Equation 5.18. Considering this interpretation, at any fixed time t, as the optimization flows, the $R_t^{param}(\mathbf{y}; r^{target})$ derived "coefficients" are constants of the optimization.

In this scenario, again, the term $J(\mathbf{x}, \tilde{\mathbf{x}}) = D(\mathbf{x}, \tilde{\mathbf{x}}) + c_1' R^{est}(\tilde{\mathbf{y}})$ stands to the VAE equivalent rate-distortion loss. But the term $c_2'(R^{est}(\tilde{\mathbf{y}}))^2$ is not correspondent to the traditional variational loss. Therefore, the generalization of the loss remains as if it is not a directly VAE-derived function. Nevertheless, as done in Section 5.4.2, this new

formulation can be put into a non-canonical variational loss as follows:

$$
\begin{aligned}
J(\mathbf{x}, \tilde{\mathbf{x}}) &= D(\mathbf{x}, \tilde{\mathbf{x}}) + \beta \left(\frac{R_t^{param}(\mathbf{y}; r^{target}) - R^{est}(\tilde{\mathbf{y}})}{R_t^{param}(\mathbf{y}; r^{target})} \right)^2 \\
&= D(\mathbf{x}, \tilde{\mathbf{x}}) + \beta \left(\frac{R_t^{param}(\mathbf{y}; r^{target}) - R^{est}(\tilde{\mathbf{y}})}{R_t^{param}(\mathbf{y}; r^{target})} \right)^2 \frac{R^{est}(\tilde{\mathbf{y}})}{R^{est}(\tilde{\mathbf{y}})} \\
&= D(\mathbf{x}, \tilde{\mathbf{x}}) + \beta \left(\frac{R_t^{param}(\mathbf{y}; r^{target}) - R^{est}(\tilde{\mathbf{y}})}{R_t^{param}(\mathbf{y}; r^{target}) \sqrt{R^{est}(\tilde{\mathbf{y}})}} \right)^2 R^{est}(\tilde{\mathbf{y}}) \\
&= D(\mathbf{x}, \tilde{\mathbf{x}}) + h(\mathbf{y}; R_t^{param}(\mathbf{y}; r^{target}), \beta) R^{est}(\tilde{\mathbf{y}})
\end{aligned} \tag{6.12}
$$

where, in comparison with Equation 5.19, $R_t^{param}(\mathbf{y}; r^{target})$, a function taken as a constant in the optimization replaces the old parametrization of h which was r^{target}. This substitution, as a consequence, may represent a generalization of rate-distortion classical loss with $h(\mathbf{y}; R_t^{param}(\mathbf{y}; r^{target}), \beta)$ playing the role of a dynamical component, instead of the classical λ Lagrangian term. Following this idea, it is also possible to represent a version where this complex variable relevance term is accompanied by the $D(\mathbf{x}, \tilde{\mathbf{x}})$:

$$
\begin{aligned}
J(\mathbf{x}, \tilde{\mathbf{x}}) &= h^{-1}(R^{est}(\tilde{\mathbf{y}}); R_t^{param}(\mathbf{y}; r^{target}), \beta) D(\mathbf{x}, \tilde{\mathbf{x}}) + R^{est}(\tilde{\mathbf{y}}) \\
&= \frac{1}{\beta} \left(\frac{R_t^{param}(\mathbf{y}; r^{target}) \sqrt{R^{est}(\tilde{\mathbf{y}})}}{R_t^{param}(\mathbf{y}; r^{target}) - R^{est}(\tilde{\mathbf{y}})} \right)^2 D(\mathbf{x}, \tilde{\mathbf{x}}) + R^{est}(\tilde{\mathbf{y}})
\end{aligned} \tag{6.13}
$$

In these two equations above, $R_t^{param}(\mathbf{y}; r^{target}) - R^{est}(\tilde{\mathbf{y}})$ plays a fundamental role in both h and h^{-1}. In h, this term zeroes the function as $R_t^{param}(\mathbf{y}; r^{target}) \to R^{est}(\tilde{\mathbf{y}})$ because it is located in the numerator. In the second version, using h^{-1}, $R_t^{param}(\mathbf{y}; r^{target}) - R^{est}(\tilde{\mathbf{y}})$ is in the denominator. As $R_t^{param}(\mathbf{y}; r^{target}) \to R^{est}(\tilde{\mathbf{y}})$, the function increases the importance of minimizing the distortion since the rate is close to the desired value.

The remaining term in the fractions, $R_t^{param}(\mathbf{y}; r^{target}) \sqrt{R^{est}(\tilde{\mathbf{y}})}$, can be interpreted as relevance factors that contribute to this dynamic relevance in the rate-distortion generalization view. Therefore a similar VAE behavior in this tradeoff is expected, apart from the more unstable convergence due to its dynamic and more complex structure. The parallel analysis in Equations 6.12 and 6.13 does not represent a formal argument regarding a rate-distortion interpretation. However, it can help enhance the understanding of the behavior of this loss function and its relation to the variational formulation in a non-linear fashion.

6.2.6 The General Time-Adaptive Target-Rate Loss

The definition of $R_t^{param}(\mathbf{y}; r^{target})$ replaces the old constant R^{param} in several equations derived in this chapter. As shown in the previous sections, R^{param}, related to mean-rate convergence, heavily influences the neural network's behavior. Since this influence factor is stable, there will be no oscillations in the neural network's convergence apart from the expected statistical variability of the signals. In this chapter, $R_t^{param}(\mathbf{y}; r^{target})$ can oscillate over time. For convergence in this scenario, as training progresses, $R_t^{param}(\mathbf{y}; r^{target})$ must start to oscillate less. This allows the neural network to converge to the desired behavior. Therefore, a fundamental aspect of employing the complex formulation presented earlier is controlling the stability of this training process.

Based on these assumptions, there is still some interesting generalization of the behavior of the loss in Equation 6.7 regarding the stability of $R_t^{param}(\mathbf{y}; r^{target})$. Additionally, an empirical strategy will be provided to show improvements in the low-rate compression range.

Exponential Desired-Rate Decay

An empirical hypothesis when applying this loss function, despite the greater difficulty in optimization, is that it would be beneficial to train the neural network at a higher rate initially and then gradually impose increasing rate restrictions. The rationale is that low-rate compression is very restricted due to the high β values in function 6.7. Given the variability of $R_t^{param}(\mathbf{y}; r^{target})$, it is challenging to start neural network optimization under such restrictive and potentially unstable conditions.

The idea is to train a neural network more "freely" at first and then impose higher restrictions. A mechanism was devised to stimulate a smooth transition between this "pre-training" step and the final training. This separation is for clarity only, as it is still a regular full training. Therefore, an exponential target-rate decay was implemented into $R_t^{param}(\mathbf{y}; r^{target})$, based on the following parameter:

$$r_t^p(r^{target}, r^{max}, b, t^{iter}) = \begin{cases} \min\left(r^{max}, r^{target} \cdot b^{\frac{\max(0, t^{iter}-t)}{t^{iter}}}\right), & \text{if } r^{target} \leq 0.5 \\ r^{target}, & \text{otherwise} \end{cases} \tag{6.14}$$

for $t > 0$, where min and max denote the minimum and maximum between two values, respectively, and t^{iter} controls the decay's velocity and the b constant. Hereafter, $r_t^p(r^{target}, r^{max}, b, t^{iter})$ stands for the parameter adopted in the next equations. It is, again, a parameterizer that is a function of the time variable. In the first case, it behaves as an exponential function, while in the latter, it shows a constant behavior. The alter-

nation of this behavior when $r^{target} \leq 0.5$ is empirically given by focusing this strategy on very-low-to-low rate compression. A more robust way of smoothly selecting the behavior of the neural network fundamental parameter can be the object of future studies. This set of parameters will be described as $\theta^{\mathbf{rpar}} = \{r^{target}, r^{max}, b, t^{iter}\}$, leading to the definition of the function as $r_t^p(\theta^{\mathbf{rpar}})$, representing the time-dependent target rate parametrized by $\theta^{\mathbf{rpar}}$.

Instead of using the common target r^{target}, it will be adopted in the already shown equations to formulate a new rate parameter as follows, extending the behavior of Equation 6.4:

$$R_t^{param}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}}))) = r_t^p(\theta^{\mathbf{rpar}}) + R_{t-1}^{shift}(\mathbf{y}) \tag{6.15}$$

which would extend to the following two scenarios:

$$R_t^{param}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}}))) = \begin{cases} \min\left(r^{max}, r^{target} \cdot b^{\frac{\max(0, t^{iter}-t)}{t^{iter}}}\right) + R_{t-1}^{shift}(\mathbf{y}), & \text{if } r^{target} \leq 0.5 \\ r^{target} + R_{t-1}^{shift}(\mathbf{y}), & \text{otherwise} \end{cases} \tag{6.16}$$

Exponential Moving Average

In machine learning, particularly in time series forecasting, the Exponential Moving Average (EMA) is a pivotal tool for enhancing predictive accuracy and robustness. The Exponential Moving Average, a key concept in financial analysis, is seamlessly integrated into machine learning frameworks to smooth historical data and uncover underlying trends. Unlike traditional moving averages, the EMA assigns exponentially decreasing weights to historical observations, providing a nuanced understanding of evolving patterns and trends in time series data. A classic example of using EMA in machine learning relates to one of the most adopted optimizers, Adam [130].

Although it is widely adopted and known, it is useful to define EMA to clarify later equations. The definition is as follows:

$$ema_t = \alpha x_t + (1 - \alpha) ema_{t-1} \tag{6.17}$$

where x_t represents a temporal series, ema_t is the value of the average at time t and α is a value on the interval $[0, 1]$. This parameter controls the weight relevance of the new value compared to the estimated average.

As depicted in earlier sections, $R_t^{param}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}})))$ is more prone to data and training variance. As a preventive measure, it is useful to derive a smoother version of

$R_t^{param}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}}))$, which is an application of the definition in Equation 6.17:

$$R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}})) = \alpha R_t^{param}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}})) + (1-\alpha) R_{t-1}^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}})) \tag{6.18}$$

The Composed Loss

Considering the derivations in the previous sections and empirical testing, it is possible to derive a more general formulation of Equation 6.9. The first component is the parameter $r_t^p(\theta^{\mathbf{rpar}})$ from Equation 6.14. The last modification is the smoothed $R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}}))$ as derived in Equation 6.18. The general target-rate parameterized function is given by:

$$f_t(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}})) = \left(\frac{R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}})) - R_t^{est}(\tilde{\mathbf{y}})}{R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}}))} \right)^2 \tag{6.19}$$

Equation 6.19 is used in the general target-distortion formulation, defined as:

$$J_t(\mathbf{x}, \tilde{\mathbf{x}}) = D_t(\mathbf{x}, \tilde{\mathbf{x}}) + \beta \left(\frac{R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}})) - R_t^{est}(\tilde{\mathbf{y}})}{R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}}))} \right)^2 \tag{6.20}$$

In this context, the full expansion of the loss into a parabola results in the following set of equations:

$$\begin{aligned}
J(\mathbf{x}, \tilde{\mathbf{x}}) &= D(\mathbf{x}, \tilde{\mathbf{x}}) + \beta \frac{(R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}})))^2 - 2R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}}))R^{est}(\tilde{\mathbf{y}}) + (R^{est}(\tilde{\mathbf{y}}))^2}{(R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}})))^2} \\
J(\mathbf{x}, \tilde{\mathbf{x}}) &= D(\mathbf{x}, \tilde{\mathbf{x}}) + \beta - \frac{2\beta}{R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}}))} R^{est}(\tilde{\mathbf{y}}) + \frac{\beta}{(R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}})))^2} (R^{est}(\tilde{\mathbf{y}}))^2 \\
J(\mathbf{x}, \tilde{\mathbf{x}}) &= D(\mathbf{x}, \tilde{\mathbf{x}}) + R_t^{d1}(\mathbf{y}; \theta^{\mathbf{rpar}}))R^{est}(\tilde{\mathbf{y}}) + R_t^{d2}(\mathbf{y}; \theta^{\mathbf{rpar}}))(R^{est}(\tilde{\mathbf{y}}))^2 + c_3
\end{aligned} \tag{6.21}$$

The main difference from Equation 6.10 is the coefficient terms of the penalization parabola. Previously, these terms were $R_t^{c1}(\mathbf{y}; r^{target}) = \frac{2\beta}{R_t^{param}(\mathbf{y}; r^{target})}$ and $R_t^{c2}(\mathbf{y}; r^{target}) = \frac{\beta}{\left(R_t^{param}(\mathbf{y}; r^{target})\right)^2}$. Now, there are new coefficients given by:

$$\begin{aligned}
R_t^{d1}(\mathbf{y}; r^{target}) &= \frac{2\beta}{R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}})} \\
R_t^{d2}(\mathbf{y}; r^{target}) &= \frac{\beta}{(R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}}))^2}
\end{aligned} \tag{6.22}$$

This interpretation still applies because, as in the scenario in Equation 6.10, the coefficients only concern terms that will be fixed in the gradient calculations of $J(\mathbf{x}, \tilde{\mathbf{x}})$. These target-rate parabolas can be interpreted as dynamic parabolas adapting to the many stochastic elements in training this neural network. However, Equation 6.21 exhibits a heavier time-dependent update flow given the EMA introduction in the rate parameter.

These derivations yield very complex formulas in terms of the fundamental components of the estimations. Still, their expansion does not highlight any easily interpretable aspect in general. Nevertheless, a more profound comprehension and analysis of the modeling could be done along with empirical experiments to understand the repercussions of this new model and provide the chance for better fine-tuning.

In a final comparison, we can derive a "dynamic rate-distortion" view of the formula as it was exhibited in the earlier versions of the loss:

$$\begin{aligned} J(\mathbf{x}, \tilde{\mathbf{x}}) &= D(\mathbf{x}, \tilde{\mathbf{x}}) + \beta \left(\frac{(R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}})) - R^{est}(\tilde{\mathbf{y}})}{(R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}}))} \right)^2 \\ &= D(\mathbf{x}, \tilde{\mathbf{x}}) + \beta \left(\frac{(R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}})) - R^{est}(\tilde{\mathbf{y}})}{(R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}}))} \right)^2 \frac{R^{est}(\tilde{\mathbf{y}})}{R^{est}(\tilde{\mathbf{y}})} \\ &= D(\mathbf{x}, \tilde{\mathbf{x}}) + \beta \left(\frac{(R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}})) - R^{est}(\tilde{\mathbf{y}})}{(R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}}))\sqrt{R^{est}(\tilde{\mathbf{y}})}} \right)^2 R^{est}(\tilde{\mathbf{y}}) \\ &= D(\mathbf{x}, \tilde{\mathbf{x}}) + h(\mathbf{y}; (R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}})), \beta) R^{est}(\tilde{\mathbf{y}}) \end{aligned} \tag{6.23}$$

which, again, could be interpreted as a dynamic rate-distortion formula whose weights are dynamically adjusted considering the whole set of estimations detailed in this proposal as a whole. As the results will highlight, this model behaves similarly to the one detailed in Chapter 5.

6.3 Results

The experiments performed using the proposal depicted in this chapter are described in this section. First, the training and inference setup is presented in Section 6.3.1. An analysis of the stability of the convergence of the target rate is depicted in Section 6.3.2. A complete rate-distortion comparison with the target-rate baseline is presented in Section 6.3.3. Lastly, Section 6.3.4 presents an ablation study.

6.3.1 Training and Inference Specifications

Non-overlapping patches of size 256×256 are used for model training, gathered from the following datasets:

- CLIC Professional dataset [119];
- CLIC Mobile dataset [119];
- DIV2K dataset [120];

Table 6.1: The models compared in the main results.

Models	**Reference**
Target Baseline (TB)	Models using Guerin's proposal (Equation 5.14), which were trained for 500k iterations [48]
Rate Adaptation (RA)	Models adopting the Time-adaptative-parameter idea (Equation 6.8), which were trained for 500k iterations
Best Strategy (BS)	Models implementing the General Time-adaptive-parameter approach (Equation 6.20), which were trained for 750k iterations

- Ultra-Eye Ultra HD dataset [121];
- MCL-JCI [122, 123];
- FLICKR2K dataset [124];

This alignment is useful for comparing the model proposed in Chapter 5 with the current one. Eight networks were trained to achieve rates $\{0.06, 0.12, 0.25, 0.5, 0.75, 1.0, 1.5, 2.0\}$, with a maximum deviation of 15%, in line with the setup in Chapter 5. The networks were trained for $500k$ and $750k$ iterations to evaluate the impact of increased training time on the newer losses.

The β constant in Equation 6.6 was set in accordance with [48]. This parameter is heuristic-dependent, but the next step is to apply the idea developed in this chapter to r^{target} in the β constant. The values were empirically set for the exponential moving average to $\alpha = 0.99$, $r^{max} = 3$, and $b = 50$, as it represents a high initial rate compared to the target rates. Lastly, the desired rate decay was performed for $t^{iter} = 150k$ iterations. The results presented here were evaluated only on the Kodak dataset and also applied to the JPEG AI dataset.

The details presented in the following sections compare three different models described in Table 6.1. The baseline model is the earlier approach, which is published in Guerin's work [48] and detailed in the last chapter. The Rate Adaptation (RA) model is the one presented in the current chapter under Equation 6.8, with models trained only for $500k$ iterations as in the previous approach. Lastly, the Best Strategy (BS) model is the one derived in Section 6.2.6 represented by Equation 6.20, with models trained for $750k$ iterations.

6.3.2 Analysis of the Average Rate and Variance

The first analysis in this version of the work is the convergence to the average rate and the variance for each of these rates. For this purpose, we analyzed the mean values and variance of the rates obtained in the Kodak dataset.

Table 6.2: The non-parametric models' mean and variance of rates reached.

Target Rate	**Average**			**Standard deviation**		
	Best Strategy	Rate Adaptation	Target Baseline	Best Strategy	Rate Adaptation	Target Baseline
0.06	0.062	0.058	0.057	0.003	0.004	0.001
0.12	0.119	0.119	0.12	0.003	0.006	0.003
0.25	0.252	0.252	0.239	0.007	0.007	0.004
0.50	0.508	0.505	0.485	0.024	0.019	0.024
0.75	0.764	0.759	0.729	0.031	0.029	0.035
1.0	1.009	1.007	0.99	0.036	0.038	0.04
1.5	1.524	1.513	1.405	0.04	0.045	0.059
2.0	2.044	1.975	1.995	0.065	0.062	0.085

Tables 6.2 and 6.3 show the expected results for both the parametric and non-parametric entropy models. Regarding Table 6.2, the mean values of all rates remained close to the target rates. The oscillations are due to the variance of the output rates and as in all these neural-based approaches, only a mean behavior with intrinsic variance can be observed. This variance can be controlled by the β constant in Equations 6.8, 6.20, and 5.14. However, since the β constant remained unchanged, the parameters from the results presented in Guerin's work were adopted here. These β values are shown in Table 5.3. Ideally, β values would differ for the approaches in this chapter. Alternatively, the current approaches would require a dynamic and clever way to set β values.

The results are quite good, even when adopting the non-ideal β values from Guerin's approach [48] for the current approach. A lower deviation is observed as the rates are lowered. This is expected because the β values are noticeably higher for these rates, as controlling a 15% deviation from the target becomes increasingly challenging. Therefore, the current models maintained the same final behavior as the baseline target approach.

It is important to note that all considerations applied to the non-parametric model results also apply to the parametric model results, depicted in Table 6.3. The mean rates reached values near the target rates, and the variance exhibited the same behavior as in the non-parametric results. This is expected because the interpretation of the β constant is the same for the parametric model, where lower rates require higher β. This also shows that the time-variable-parameter idea can work with different entropy models. In conclusion, the strategy applied here can potentially be extended, supporting the idea of being a simple plug-in solution for complex architectures.

f

Table 6.3: The mean and variance of rates reached by the parametric models.

Target Rate	**Average**			**Standard deviation**		
	Best Strategy	Rate Adaptation	Target Baseline	Best Strategy	Rate Adaptation	Target Baseline
0.06	0.06	0.058	0.059	0.004	0.002	0.003
0.12	0.119	0.118	0.118	0.007	0.003	0.004
0.25	0.253	0.246	0.252	0.01	0.006	0.006
0.50	0.496	0.498	0.49	0.029	0.029	0.027
0.75	0.753	0.753	0.731	0.043	0.043	0.03
1.0	1.024	1.024	0.956	0.051	0.051	0.034
1.5	1.534	1.534	1.467	0.053	0.053	0.041
2.0	2.034	2.034	1.917	0.074	0.074	0.067

6.3.3 Rate-distortion Comparison

This section shows the rate-distortion comparison of the three models in Table 6.1 in terms of PSNR, MSSIM, and MS-SSIM measures. The first comparison is presented in Table 6.4 and analyzes the PSNR results. These results show an average improvement in the basic Rate Adaptation model compared to the Baseline model. Furthermore, the Best Strategy model showed better average results than the Rate Adaptation model.

The great advantage of the proposal in this chapter is that it abandons the costly heuristic search presented in Chapter 5. Although the current strategy still relies on searching for the β parameter, which was maintained as in the previous approach, applying it to the β parameter is feasible. This adaptation can be straightforwardly adapted to the requirements of the β constant. Consequently, more robust modeling behavior on this variance parameter can be employed with the time-adaptive idea.

The rate-distortion results for the perceptual measures are presented in Table 6.5 for the SSIM measure and Table 6.6 for the MSSIM measure. It is important to note that the results for both SSIM and MSSSIM are almost equivalent between the baseline target and the two models proposed in this chapter. It is known in the neural-compression literature that neural networks, for many potential reasons, possess interesting reconstruction results regarding the subjective appearance of images. They exhibit considerable results even at low rates like 0.06.

From the perceptual point of view, these reconstructions were already good in Chapter 5 compared to Ballé's approaches [45, 23]. Early results showed almost no loss compared to the non-restricted optimization of Ballé's models. Consequently, extending this observation to the time-variable-parameter approach proposed in this chapter can be interesting. It also exhibits no loss compared to the baseline target and, by extension, to Ballé's original approaches.

Table 6.4: The mean results of the PSRN obtained in the Kodak dataset for the three models compared in this chapter. It is possible to see that the Best Strategy model consistently showed the best results among the three.

Rate x PSNR	**Non-parametric**			**Parametric**		
	Best Strategy	Rate Adaptation	Target Baseline	Best Strategy	Rate Adaptation	Target Baseline
0.06	**24.25**	23.86	23.82	**24.78**	24.09	23.73
0.12	**25.94**	25.91	25.40	**26.24**	25.60	26.08
0.25	**28.08**	27.81	27.32	28.20	27.58	**28.30**
0.50	**30.59**	30.51	30.30	**30.93**	30.74	30.76
0.75	32.35	**32.43**	32.17	**32.87**	32.67	32.53
1.0	**33.99**	33.91	33.74	**34.48**	34.20	33.75
1.5	**36.27**	36.15	35.86	**36.94**	36.56	36.42
2.0	**38.57**	38.09	38.33	**39.11**	38.65	38.36

Table 6.5: The mean results of the SSIM obtained in the Kodak dataset for the three models are compared in this chapter.

Rate × SSIM	**Non-parametric**			**Parametric**		
	Best Strategy	Rate Adaptation	Target Baseline	Best Strategy	Rate Adaptation	Target Baseline
0.06	**0.64**	0.61	0.61	**0.65**	0.62	0.61
0.12	**0.71**	0.70	0.68	**0.71**	0.70	**0.71**
0.25	**0.80**	0.79	0.77	**0.80**	0.78	**0.80**
0.50	**0.87**	**0.87**	0.86	0.87	**0.88**	0.87
0.75	**0.91**	**0.91**	**0.91**	**0.92**	**0.92**	0.91
1.0	**0.93**	**0.93**	**0.93**	**0.94**	**0.94**	0.93
1.5	**0.96**	**0.96**	**0.96**	**0.97**	**0.97**	0.96
2.0	**0.98**	0.97	0.97	**0.98**	**0.98**	0.97

Table 6.6: The mean results of the MSSSIM obtained in the Kodak dataset for the three models compared in this chapter.

Rate × MSSSIM	**Non-parametric**			**Parametric**		
	Best Strategy	Rate Adaptation	Target Baseline	Best Strategy	Rate Adaptation	Target Baseline
0.06	**0.85**	0.81	0.83	**0.86**	0.84	0.83
0.12	**0.90**	**0.90**	**0.90**	**0.91**	0.90	**0.91**
0.25	**0.95**	0.94	0.94	**0.94**	**0.94**	**0.94**
0.50	**0.97**	**0.97**	**0.97**	**0.97**	**0.97**	**0.97**
0.75	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**
1.0	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**
1.5	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**
2.0	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**

6.3.4 Ablation Study of the Adaptive Loss

Section 6.3.3 compared the results of three models: the Target Baseline model, the Rate Adaptation model, and the Best Strategy model. Please refer to Table 6.1 for the model nomenclature. The results of the Best Strategy model showed improvements compared to the other models. Also, even if no improvements in the mean rate-distortion analysis were observed, the time-adaptive idea can bring the most significant improvement of discarding the extensive space parameter search proposed by the heuristics described in the last chapter. This idea could be applied in many areas of machine learning where some parameters are constants but could be transformed into functions of the neural network's behavior concerning the model's primary objective.

We can compare the Rate Adaptation model directly with the Target Baseline model since the modification relies on one component of the initial model, the constant R^{param}. The Rate Adaptation model transforms it into a function $R_t^{param}(\mathbf{y}; r^{target})$. This function triggers a chain of dependency on the parameter with respect to the time evolution of the training and the intrinsic behaviors of the neural networks concerning the whole target-rate distortion objective. The rate-distortion results depicted in Tables 6.4, 6.5, and 6.6 show no degradation in this scenario. This potential degradation was a concern since the number of free adaptive variables increased, and the formula introduced time feedback.

This direct comparison, regarding the change of R^{param} to $R_t^{param}(\mathbf{y}; r^{target})$, can be concluded from this direct analysis. Nevertheless, the Best Strategy model adopted more processing steps for the parameter, leading to the mean-rate parameter $R_t^{ema}(\mathbf{y}; r_t^p(\theta^{\mathbf{rpar}}))$. This mean-rate parameter introduces an EMA calculation, now referred to as R_t^{ema}, and an inner parameter that controls a rate-decay strategy, depicted by $r_t^p(\theta^{\mathbf{rpar}})$. Therefore, it is interesting to derive statistics on the impact of these strategies to make the Best Strategy model more robust compared to the other two. Namely, the effect of the extended training time, the repercussions of the smoothing of $R_t^{param}(\mathbf{y}; r^{target})$, and the rate-decay target parameterized by $r_t^p(\theta^{\mathbf{rpar}})$. The following sections will explore this in more detail.

Impact of Training Time

As mentioned earlier, given the increased parameter complexity of the Rate Adaptation model and the Best Strategy model, increasing the training time may improve the models' outcomes. The statistical results are shown in Table 6.7, where a comparison between the Best Strategy and Rate Adaptation models is performed. In this scenario, where non-parametric entropy modeling is adopted, the increased training time consistently improved the results. As the perceptual metrics quickly saturate at high values when the compression method is neural-based, PSNR was chosen because it is more sensitive to

Table 6.7: Ablation study of the impact of the training time on both the Rate Adaptation model and the Best Strategy Model for the non-parametric modeling.

Rate × PSNR	**Non-parametric**			
	Rate Adaptation		**Best Stragey**	
Number of iterations	**500000**	**750000**	**500000**	**750000**
0.06	**23.86**	23.47	24.18	**24.25**
0.12	25.91	**25.94**	25.91	**25.95**
0.25	27.81	**27.96**	27.99	**28.08**
0.50	30.51	**30.61**	30.50	**30.79**
0.75	32.43	**32.53**	32.40	**32.47**
1.0	33.91	**33.92**	33.73	**33.84**
1.5	36.15	**36.38**	36.17	**36.41**
2.0	38.09	**38.48**	38.39	**38.68**

variations. Nevertheless, with $500,000$ iterations, the baseline results were already good in terms of SSIM and MSSSIM.

It is essential to highlight that the analysis of the ablation section will rely only on PSRN. Not only because of its sensitivity to changes in the models but also because when Ballé's models are considered as reference [45, 23], the basic target-rate models detailed in Chapter 5 demonstrated significative deteriorations only regarding the PSNR measure. Therefore, the improvements shown here imply a decrease in the gap between the non-restrict rate-distortion optimization and the restrict-relaxed rate-distortion optimization.

As in the non-parametric scenario, comparing the models with parametric modeling showed the same tendency. Improvements were observed with increased training time. Therefore, considering that the increase of 250,000 training steps is almost negligible in the current AI hardware scenario, it is worth considering this a general approach to apply to all parameter-adaptive models.

Smoothing of R_t^{param}

One hypothesis for making the parameter R_t^{param} a time-adaptive function is that the network would be more prone to training variations and noise due to the batch-based correction of the mean-rate mismatch. Therefore, applying EMA smoothing was intended to make it less susceptible to these oscillations, especially in the early training steps.

The results for the non-parametric model are shown in Table 6.9. The table compares the rate adaptation model, the Best Strategy model that applies only smoothing, and the full Best Strategy model. With only smoothing, slight improvements in the PSNR can be seen when comparing the Target Baseline model and the Best Strategy model. The Best

Table 6.8: Ablation study of the impact of the training time on both the Rate Adaptation model and the Best Strategy Model for the parametric modeling.

Rate × PSNR	**Parametric**			
	Rate Adaptation		**Best Stragey**	
Number of iterations	**500000**	**750000**	**500000**	**750000**
0.06	**24.09**	24.06	24.51	**24.78**
0.12	**25.60**	25.45	26.20	**26.24**
0.25	27.58	**27.65**	28.15	**28.20**
0.50	30.74	**30.87**	30.85	**30.90**
0.75	32.67	**32.83**	32.83	**32.91**
1.0	34.20	**34.36**	34.20	**34.37**
1.5	36.56	**36.73**	36.63	**36.69**
2.0	38.65	**38.81**	38.79	**39.02**

Strategy model shows, in general, almost no variation compared to the strategy without the decay component.

This is one of the highlights of this ablation study. The non-parametric model showed slight improvement with smoothing the mean-rate parameter. However, considering the whole strategy, which includes the decay strategy, almost no improvements are observable in any range of rates. As these two strategies are easily implemented and impose no computational burden on conventional hardware, they were adopted as the general strategy for both the parametric and non-parametric models.

The parametric model analysis, depicted in Table 6.10, shows similar results for the smoothing and the whole strategy for the parametric model, with some improvements in some scenarios and stability in others. It is interesting to note that, in general, the use of decay, especially for medium-to-high rates, deteriorated the results. This is why Section 6.2.6 established an empirical threshold for the technique's applicability. In general, these refinements made the Best Strategy model the most stable proposal. The decay analysis will be discussed in the next section of the ablation study.

The Rate Decay Strategy

The rate decay strategy is probably the most specific approach presented in this ablation study. This can be seen in the results of Table 6.9 and Table 6.10, where, at medium-to-high rates, the absence of decay presents worse results compared to the scenarios where only smoothing is applied. However, it showed stable results or slight improvements at very-low-to-low rates. For parametric and non-parametric models, these results are presented in Table 6.11 and Table 6.12, respectively.

Table 6.9: Ablation study of the impact of the smoothing on both the Rate Adaptation model and the Best Strategy Model for the non-parametric modeling.

Rate × PSNR	**Non-Parametric**		
Number of iterations	**Rate Adaptation**	**Best Strategy without Smoothing**	**Best Strategy**
0.06	23.47	23.96	**24.25**
0.12	25.94	**25.95**	**25.95**
0.25	27.96	**28.08**	**28.08**
0.50	30.61	30.60	**30.79**
0.75	32.53	32.35	**32.47**
1.0	33.92	**33.99**	33.84
1.5	36.38	36.27	**36.41**
2.0	38.48	38.57	**38.68**

Table 6.10: Ablation study of the impact of the smoothing on both the Rate Adaptation model and the Best Strategy Model for the parametric modeling.

Rate × PSNR	**Parametric**		
Number of iterations	**Rate Adaptation**	**Best Strategy without Smoothing**	**Best Strategy**
0.06	24.06	24.68	**24.78**
0.12	24.45	25.69	**26.24**
0.25	27.65	28.05	**28.20**
0.50	30.87	30.72	**30.90**
0.75	32.83	32.70	**32.91**
1.0	34.36	**34.41**	34.37
1.5	36.73	36.66	**36.69**
2.0	38.81	38.63	**39.02**

Table 6.11: Ablation study of the impact of decay on both the Rate Adaptation model and the Best Strategy Model for the non-parametric modeling.

Rate × PSNR	**Non-Parametric**		
Number of iterations	**Rate Adaptation**	**Best Strategy without Decay**	**Best Strategy**
0.06	23.47	24.24	**24.25**
0.12	25.94	**25.95**	**25.95**
0.25	27.96	28.04	**28.08**
0.50	30.61	30.74	**30.79**
0.75	32.53	**32.57**	32.47
1.0	33.92	**33.93**	33.84
1.5	36.38	**36.50**	36.41
2.0	38.48	**38.75**	38.68

In the non-parametric model, as shown in Table 6.11, the decay strategy alone presented improvements at a rate of 0.06 and stability or slight enhancements up to the target of 0.50. At higher rates, these improvements were not consistent. Therefore, it was hypothesized that the decay strategy could be helpful at very low rates.

The results for the parametric model, shown in Table 6.12, show consistent improvements at the lowest range of rates and some stability up to around 0.50. This observation inspired the idea proposed by Equation 6.14, which was applied with a threshold of 0.50.

This could improve parametric models at the lowest range and could be explored in further studies. As shown in the analysis of Chapter 5, the non-parametric model showed significant stability compared to the parametric model. Coincidentally or not, the non-parametric model exhibited a low mean rate shift effect. The studies and differences between these models could be of high interest, especially regarding potential tunings to make these models more robust to restricted optimizations.

6.3.5 Subjective evaluation

Figures 6.3 and 6.4 display reconstructions using the Target Baseline (TB) architecture and the Rate Adaptation (RA) model for the Kodak dataset. Each row represents an image whose reconstruction fell within the tolerance range for both architectures. Figure 6.3 relates to the non-parametric models, while Figure 6.4 shows results for the parametric models.

In this chapter, the same set of images selected in the Chapter 5 experiments were chosen, as depicted in Figures 5.21 and 5.19. As can be seen, perceptually, the images

Table 6.12: Ablation study of the impact of decay on both the Rate Adaptation model and the Best Strategy Model for the parametric modeling.

Rate × PSNR	**Parametric**		
Number of iterations	**Rate Adaptation**	**Best Strategy without Decay**	**Best Strategy**
0.06	24.06	24.06	**24.78**
0.12	25.45	25.69	**26.24**
0.25	27.65	27.63	**28.20**
0.50	30.87	**30.93**	30.90
0.75	32.83	32.87	**32.91**
1.0	34.36	**34.48**	34.37
1.5	36.73	**36.94**	36.69
2.0	38.81	**39.11**	39.02

maintain the same behavior of the Target Baseline model compared to the Rate Adaptation model. However, there are noticeable improvements in some images at very-low-to-low rates. For example, at a rate of 0.12, depicted in the first row of Figure 6.3, it is possible to see a perceptual improvement, especially in the river, from the Target Baseline to the Rate Adaptation. The perceptual quality sometimes fluctuates, as seen in Figure 5.19, where the Target Baseline model showed slightly less blurred image reconstructions. It is challenging to notice perceptual differences between the two approaches at higher rates.

The results of the rate-distortion metrics showed this equivalence. Therefore, this new rate adaptation model maintains, in general, the subjective quality around the baseline target and applies the strategy with the potential to overcome the need for search heuristics. It also shows improvements in some scenarios.

As a final observation, a comparison between the three models at low rates of $[0.12, 0.25]$ is shown in Figure 6.5. From this figure, it can be seen that the Best Strategy improves the subjective quality of the image compared to the Rate Adaptation model, which aligns with the rate-distortion measures shown earlier.

6.4 Conclusions

This chapter introduces an improved approach for rate-constrained coding in neural networks, building upon the earlier work detailed in Chapter 5. The previous approach proposed a target-rate loss, where the mean-rate shift problem was considered a conse-

Figure 6.3: Reconstructions obtained using the target **non-parametric** models for images from the Kodak dataset. Original (left), Non-parametric TB (center), and Non-parametric RA (right). The considered bit rates for the reconstructions are $\{0.12, 0.25, 0.5, 0.75, 1.5\}$, each representing a row of images.

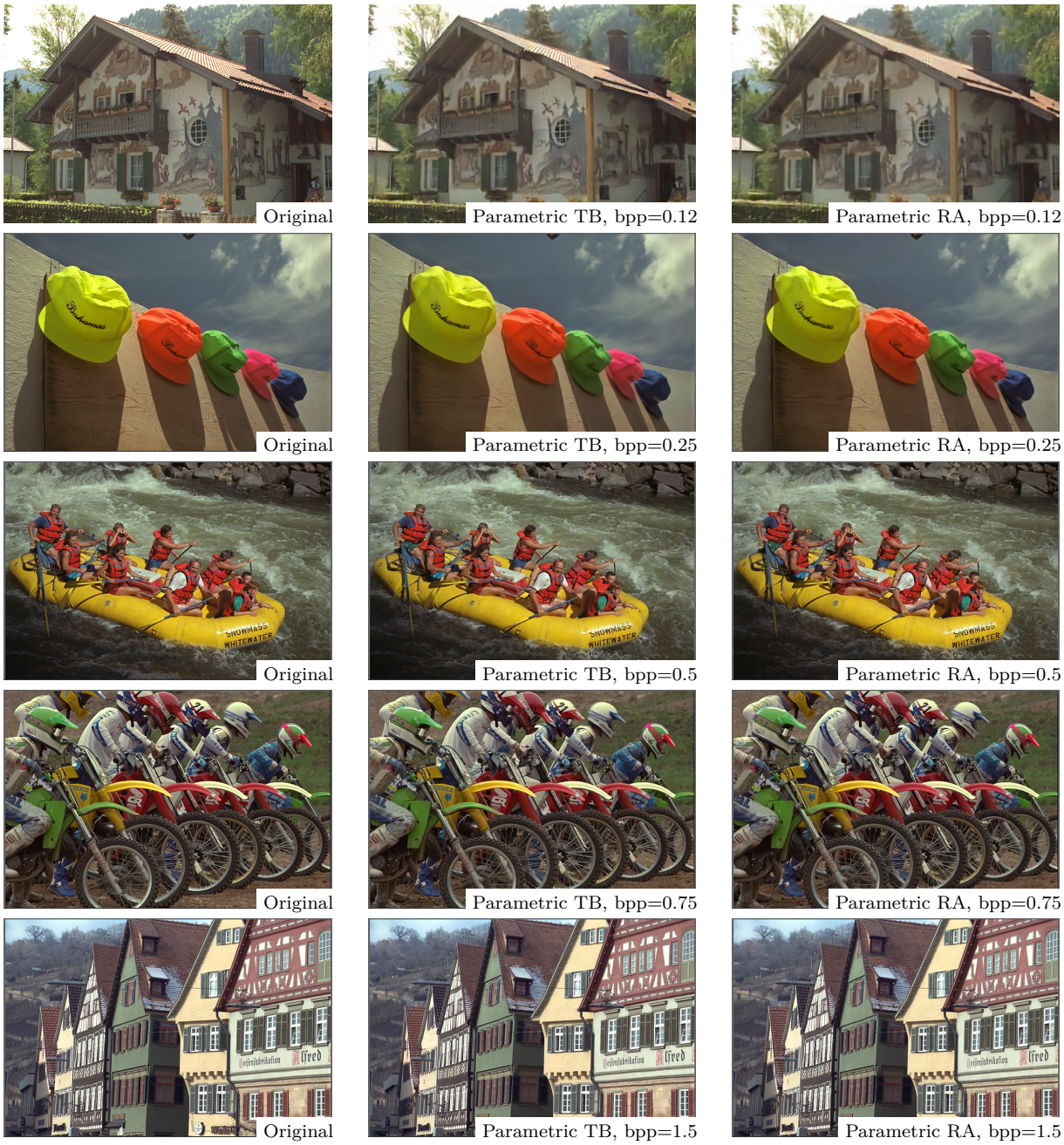


Figure 6.4: Reconstructions obtained using the **parametric** models for images from the Kodak dataset. Original (on the left), Parametric TB (in the middle), and Parametric RA (on the right). The considered rates for the reconstructions are $\{0.12, 0.25, 0.5, 0.75, 1.5\}$, each representing a row of images.

Figure 6.5: Reconstructions obtained using the **parametric** and **non-parametric** models for images from the Kodak dataset. Target baseline (on the left), Rate adaptation (on the center), and Best Strategy (on the right).

quence of the additive uniform noise approximation. Theoretical and empirical evidence was provided. The proposed solution in that scenario involved costly search heuristics to find optimal values for its hyperparameters.

The current chapter generalizes the earlier proposal, essentially following the same approach of devising a target-rate loss. However, inspired by reinforcement learning ideas, the mean-rate hyperparameter is now converted into a time-dynamical function of the neural network information, designed to make the model auto-correct the shift mismatches.

Compared to the target baseline of Chapter 5, the approach in this chapter shows improvements or similar results in all main measures (PSNR, SSIM, MSSSIM), but now adopts an idea that can release the approach from the burden of the costly search heuristics. The method remains a plug-in solution suitable for more complex approaches.

There is room for several improvements upon the parameter-adaptive loss. The most straightforward extension would be to adapt and apply the time-adaptive idea to the other hyperparameters of the loss, namely, the β. These temporal analysis ideas can be used in the core architecture to provide more robust modeling behaviors. For example, it could be applied to give the entropy model different properties. This core idea could also be used in other related approaches within the supervised and unsupervised learning paradigms.

Besides generalizing the mathematics to the β parameter, completely overcoming the heuristics could make this proposal extensible to multi-target rate models, as it will correct itself for the shift. The result could be a multi-target rate model that carries the basic idea proposed here without too many modifications. This extension would reinforce the plug-in idea of these approaches.

Additionally, a detailed analysis of the minor corrections applied, such as the EMA smoothing or the rate decay, could be conducted. Specifically, the decay approach could be fine-tuned to harness more potential from the neural networks at very low rates. Lastly, much of the future work described in Chapter 5 can also be applied here, such as using different quantization strategies to improve performance. Additionally, there is potential to study the differences between the parametric and non-parametric modeling proposed by Ballé's works to guide an improved entropy model for the current rate-restricted optimization.

Chapter 7

Conclusion

This thesis studied the general approach of using neural networks to perform end-to-end optimization based on variational methods. These variational approaches mathematically reduce to classical rate-distortion optimization. Specifically, this document thoroughly discussed the problem of imposed rate constraints. Despite progress in neural image compression, these methods have not been widely adopted in practical applications. One significant reason highlighted is the underexplored problem of rate control for reconstructing arbitrary images.

Approaches achieving the desired rate require encoding in multiple steps, making them complex for practical use. Seminal approaches generally require training multiple networks to obtain rate-distortion operating points. Additionally, only minor advancements have been made in target-constrained training, presenting the complexity of optimizing target rate restrictions. Given this setup, this thesis deeply discussed a set of ideas and architectures to address this underexplored problem.

The VAE-based architectures were thoroughly presented in Chapter 4. The main focus was to show the problems when controlling the rate. Seminal baselines do not control rate deviation, so each trained model operates on different RD points on the convex plane. This chapter also detailed seminal parametric and non-parametric entropy modeling works. This comprehensive approach was used in later chapters, where modifications with profound implications are presented.

Based on the earlier chapters' studies, the first basic approach was presented in Chapter 5. This approach modified the loss function to minimize distortion while including the rate restriction using Lagrangian relaxation ideas. It led to a loss function where training could yield models that reconstruct around a target rate. Some issues with the additive uniform noise approximation were covered, named the rate mean-shift problem.

The design of rate-oriented training highlighted the misalignment caused by the approximation of the quantization operation during training. This misalignment became

evident with the average rate deviation issue, inspiring heuristics to estimate the loss function parameters. Empirical tests and theoretical analysis concluded that the misalignment between the estimated rate during training and the actual rate during inference is related to additive uniform noise. This issue results in a rate overestimation and, consequently, an over-penalization during training, shifting the actual rate at inference time below the expected value.

Chapter 6 introduced an improved approach for rate-constrained coding in neural networks, building upon the earlier work detailed in Chapter 5. This method generalized the earlier proposal by devising a target-rate loss inspired by reinforcement learning ideas. The mean-rate hyperparameter was converted into a time-dynamical function of the neural network information, designed to auto-correct the shift mismatches. Compared to the target baseline of Chapter 5, the approach in Chapter 6 showed improvements or similar results in all main measures (PSNR, SSIM, MSSSIM). This approach adopts an idea that can eliminate the need for costly search heuristics, making it a plug-in solution suitable for more complex approaches.

This thesis presents several contributions. One is modifying a variational rate-distortion loss into a constrained target bitrate loss using Lagrangian relaxation. This method shows that even though these losses are not directly derived from variational Bayesian inference, their hyperparameters retain some features of the variational approach. Additionally, the analysis of the quantization mismatch problem, termed the mean rate shift problem, caused by using additive uniform noise as a proxy for non-differential quantization, is another contribution. A search heuristic was proposed to address this issue.

Another contribution is the solution using different ideas to solve the rate mismatch problem, namely adopting general reinforcement learning ideas. The training process is seen as a time-evolution stochastic process, allowing temporal information to fine-tune neural network training. Lastly, this thesis presents a range of analyses, potential extensions, and deeper investigations, highlighting future steps and contributing to the field.

7.1 Future Works

Several improvements can be made to the works proposed here. Regarding the parameter-adaptive loss, a straightforward extension would be to adapt and apply the time-adaptive idea to other hyperparameters of the loss (namely, β). These temporal analysis ideas can be used in the central architecture to provide more robust modeling behaviors. For example, they could give the entropy model different time-dynamic properties. This core

idea could be applied to other related approaches in supervised and unsupervised learning paradigms.

Another improvement is the detailed study of the applicability of these approaches to different or more complex architectures, fully basing the idea that this loss is generalizable to any compression end-to-end neural architecture. That is, whenever rate control is desired.

Besides generalizing the mathematics to the β parameter, eliminating the heuristics could make this proposal extensible to multi-target rate models since it will correct itself for the shift. The result can be a multi-target rate model that carries the basic idea proposed here without too many modifications. This potential extension reinforces the plug-in idea of these approaches. It is interesting to highlight that this extension could be more challenging using only the ideas of Chapter 5.

A thorough analysis of the minor corrections applied in the parameter-adaptive loss, like the EMA smoothing or the rate decay, can also be studied. Specifically, the decay approach could be fine-tuned to maximize the potential of neural networks at low rates. Lastly, much of the future work described in Chapter 5 can also be applied here. For example, different quantization strategies can be used to improve performance. There is also the potential to study the differences between the parametric and non-parametric modeling proposed by Ballé's works to guide an improved entropy model for the current rate-restricted optimization.

Appendix I

Information Theory

Information theory, originating with Claude Shannon in the mid-20th century, provides the foundation for the systematic study of data transmission and encoding schemes. This framework unravels the intricacies of data transmission—the means by which information is sent from a source to a receiver—and encoding schemes—strategies employed to represent data in a particular form [131]. The main goal of information theory is twofold: first, to define theoretical boundaries for efficient communication from a source through a channel using various encoding methods, and second, to develop encoding strategies that approach optimal performance [71, 132].

At the core of information theory are entropy and mutual information, both crucial within theoretical frameworks. Entropy measures the unpredictability or randomness of information content, while mutual information assesses the extent of information shared between two random variables. Entropy, rooted in thermodynamics, relates to discrete-time symbols from stochastic processes, representing the inherent information quantity of the process. Mutual information quantifies the knowledge one process has about another, differentiating itself from entropy, which relates only to a process's self-information [71, 132].

Aiming to illuminate foundational concepts in information theory, this appendix furnishes readers with the essential knowledge for navigating the upcoming chapters of this thesis, particularly fortifying the understanding of the neural compression approach adopted herein. Subsequent sections will delve into these concepts with enhanced depth, highlighting their relevance in the context of this research.

I.1 Introduction to Information Sources

Information theory, since its inception, has illuminated the understanding of information sources and effective communication by concentrating on the conceptual frameworks and

mechanisms that govern them [131]. Shannon's foundational work went beyond just facilitating communication. He provided a theoretical architecture that improved the efficiency of communication channels, considering both data compression and error correction [131].

Data transmission fundamentally encompasses the movement of data, rendered as symbols or messages, from a sender (source) to a receiver (destination) via a communication channel. Encoding schemes refer to the methodologies used to render messages in a format amenable to transmission. The *information source* is pivotal in this context. It acts as a model representing a physical entity, generating sequences of symbols in a stochastic manner [132].

To illustrate, consider a biased die as an analogy for comprehending the concept of an information source generating symbols stochastically. Each face (symbol) of the die (information source) may not have an equal likelihood of being selected (turning up when rolled). The *source alphabet* is essentially the repository of all conceivable symbols or outcomes, analogous to the six faces of a die.

Pivoting towards a more robust mathematical exploration, it is crucial to define some quintessential concepts with refined clarity. A *random variable*, X, is defined as a function that maps from a sample space S to the real line: $X : S \to \mathbb{R}$. The distribution $P(X)$ captures the probability of X holistically. For a random variable X, if it assumes a value $\boldsymbol{x}$, the expression $X = \boldsymbol{x}$ becomes pertinent. Additionally, $p_X(\boldsymbol{x})$ denotes the probability function value at $X = \boldsymbol{x}$, providing a more detailed, localized view of X's distribution.

The random variable type - discrete or continuous - dictates the kind of function employed to describe its probability distribution. If X is a continuous random variable, $p_X(\boldsymbol{x})$ denotes the probability density function (PDF) of X at $\boldsymbol{x}$. The PDF represents the likelihood of X falling within a certain range of values. Conversely, if X is a discrete random variable, its distribution is described using a probability mass function (PMF), similarly denoted by $p_X(\boldsymbol{x})$.

Given a function $p_X(\boldsymbol{x})$, the parameters relate to the distribution of a random variable X according to $X \sim \text{dist}(\boldsymbol{\theta})$, where the notation represents X following a distribution parameterized by $\boldsymbol{\theta}$. For instance, if X follows a normal (Gaussian) distribution, denoted as $X \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, $p_X(\boldsymbol{x}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\boldsymbol{x}-\boldsymbol{\mu}}{\boldsymbol{\sigma}}\right)^2}$, where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ are the mean and variance parameters, respectively. To simplify the discussion and maintain cohesiveness in the exposition, the notation $p_X(\boldsymbol{x})$ will occasionally be used with the implicit understanding that it refers to $p_X(\boldsymbol{x}; \boldsymbol{\theta})$ in the ensuing text.

When these variables are strung along discrete time indices, they form a *discrete-time random process*, X_n, with $n \in \mathcal{T}$, where $\mathcal{T}$ serves as an indexing set, and n and X_n designate the index and the nth random variable of the sequence, respectively [132].

For instance, the concept of the information source can be interlinked with a simple

stochastic process: consider a telegraph signal as the information source, where the state at a discrete time n, X_n, may represent a transmitted symbol at that instant, e.g., a dot or a dash in Morse code. If $\mathcal{T}$ signifies time, X_n might be the symbol being sent at time n, with the transition probabilities (change from one symbol to another) being determined stochastically, thereby forming a stochastic process.

This mathematical framework is tethered to a probabilistic space, often symbolized by the triplet $(S, \mathcal{F}, P)$ [132], wherein:

- S, the *sample space*, envelops all potential outcomes of the random process.
- $\mathcal{F}$, the *sigma-algebra*, represents a collection of events, generated as the power set of S, therefore encapsulating all possible sets of outcomes.
- P, the *probability measure*, adheres to probability axioms, gauging the events within $\mathcal{F}$ and bestowing a probability upon each feasible event.

I.2 Shannon Entropy

Claude E. Shannon's concept of Shannon entropy serves as a vital measure within the realm of information theory. It quantifies a random variable's unpredictability or inherent informational content. Delineating the average "information" or "surprise" engendered by a random variable indirectly provides insight into the variable's randomness or disorder.

Consider a random variable X, which is affiliated with a finite alphabet $\mathcal{A}$, where $\mathcal{A}$ encompasses all possible symbols or outcomes that X can attain. Shannon entropy, mathematically expressed as $\mathcal{H}(X)$, can be formulated as [71, 132]:

$$\mathcal{H}(X) = -\sum_{\boldsymbol{x}\in\mathcal{A}} p_X(\boldsymbol{x}) \log_b p_X(\boldsymbol{x}) \tag{I.1}$$

where $p_X(\boldsymbol{x})$ denotes the probability density function of the random variable X, explicating the probability of X assuming a specific value $\boldsymbol{x}$, which is represented as $P[X = \boldsymbol{x}]$. The logarithmic base b prescribes the unit of entropy: utilizing Euler's constant (e) frames entropy in "nats", while base 2 formulates it in "bits". For instance, due to its binary outcome, a fair coin toss incurs an entropy of 1 bit.

As the discussion transitions towards scenarios involving multiple random variables, it becomes pertinent to consider joint entropy. When contemplating two distinct variables, X and Y, each with respective finite alphabets $\mathcal{A}_X$ and $\mathcal{A}_Y$, the joint entropy, symbolizing the collective unpredictability across both variables, is denoted as $\mathcal{H}(X, Y)$, and calculated as [71, 132]:

$$\mathcal{H}(X, Y) = -\sum_{\boldsymbol{x}\in\mathcal{A}_X} \sum_{\boldsymbol{y}\in\mathcal{A}_Y} p_{XY}(\boldsymbol{x}, \boldsymbol{y}) \log p_{XY}(\boldsymbol{x}, \boldsymbol{y}) \tag{I.2}$$

where $p_{XY}(\boldsymbol{x}, \boldsymbol{y})$ is the joint probability distribution, defining the probability that X and Y assume values $\boldsymbol{x}$ and $\boldsymbol{y}$ concurrently. This combined perspective of multi-variable entropy paves the path towards exploring mutual information, a topic further explored in subsequent sections.

I.3 Conditional Entropy and Mutual Information

In the continual exploration of entropy, attention subtly shifts to its derivatives, particularly focusing on *conditional entropy* and *mutual information*. These concepts intertwine and build upon the fundamental principles of entropy discussed in preceding sections [71, 132].

I.3.1 Understanding Conditional Entropy

Conditional entropy, symbolized as $\mathcal{H}(Y|X)$, quantifies the residual uncertainty or entropy of a random variable Y, given the value of another variable X is known. For random variables X and Y with a joint distribution $p_{XY}(\boldsymbol{x}, \boldsymbol{y})$, the conditional entropy can be articulated through several equivalent mathematical expressions, as follows [71, 132]:

$$\begin{aligned}
\mathcal{H}(Y|X) &= \sum_{\boldsymbol{x}\in A_X} p_X(\boldsymbol{x})\mathcal{H}(Y|X=\boldsymbol{x}) && \text{(I.3)}\\
&= -\sum_{\boldsymbol{x}\in A_X} p_X(\boldsymbol{x}) \sum_{\boldsymbol{y}\in A_Y} p_{Y|X}(\boldsymbol{y}|\boldsymbol{x}) \log p_{Y|X}(\boldsymbol{y}|\boldsymbol{x}) && \text{(I.4)}\\
&= -\sum_{\boldsymbol{x}\in A_X} \sum_{\boldsymbol{y}\in A_Y} p_{Y|X}(\boldsymbol{y}|\boldsymbol{x}) \log p_{Y|X}(\boldsymbol{y}|\boldsymbol{x}) && \text{(I.5)}\\
&= -\mathbb{E}_{p_{Y|X}} \log p_{Y|X}(\boldsymbol{y}|\boldsymbol{x}) && \text{(I.6)}
\end{aligned}$$

In the above expressions, $p_{Y|X}(\boldsymbol{y}|\boldsymbol{x})$ symbolizes the conditional probability of Y given X, and $\mathbb{E}_{p_{Y|X}}$ represents the expected value operator concerning $p_{Y|X}$. Conceptually, conditional entropy encapsulates the remaining unpredictability of Y when X is ascertained, providing a measure of how much "new" information is presented by Y in the context of the known variable X.

I.3.2 Relating Entropies

The following relation harmoniously captures the synergy between joint entropy and conditional entropy, linking the entropy of a combination of two variables to the sum of the

entropy of one variable and the conditional entropy of the other [71]:

$$\mathcal{H}(X,Y) = \mathcal{H}(X) + \mathcal{H}(Y|X) \tag{I.7}$$

This relation subtly illustrates that the total unpredictability (or entropy) of a pair of variables (X,Y) is the sum of the individual unpredictability of X and the remaining unpredictability of Y once X is known.

I.3.3 Mutual Information: Quantifying Dependency

Moving deeper into the concept of mutual information, a significant metric that quantifies the information shared between two random variables is engaged, thus establishing a measure of their statistical dependence. Mathematically, mutual information, denoted as $\mathcal{I}(X,Y)$, is computed using the joint and marginal probability mass functions $p_{XY}(\boldsymbol{x},\boldsymbol{y})$, $p_X(\boldsymbol{x})$, and $p_Y(\boldsymbol{y})$, formulated as follows [71]:

$$\mathcal{I}(X,Y) = \sum_{\boldsymbol{x}\in\mathcal{A}_X}\sum_{\boldsymbol{y}\in\mathcal{A}_Y} p_{XY}(\boldsymbol{x},\boldsymbol{y}) \log \frac{p_{XY}(\boldsymbol{x},\boldsymbol{y})}{p_X(\boldsymbol{x})p_Y(\boldsymbol{y})} = \mathbb{E}_{p_{XY}} \log \frac{p_{XY}(\boldsymbol{x},\boldsymbol{y})}{p_X(\boldsymbol{x})p_Y(\boldsymbol{y})} \tag{I.8}$$

In the field of image compression, mutual information becomes pivotal in assessing how the value of one pixel (or a set of pixels or features) might influence others. Here, $\mathcal{I}(X,Y)$ is not merely a measure of shared information but might offer insights into identifying and leveraging dependencies between different parts of an image for more effective compression strategies.

Further, mutual information is also expressible in terms of entropy and conditional entropy [71]:

$$\begin{aligned}\mathcal{I}(X,Y) &= \mathcal{H}(X) + \mathcal{H}(Y) - \mathcal{H}(X,Y) &\text{(I.9)}\\ &= \mathcal{H}(X) - \mathcal{H}(X|Y) = \mathcal{H}(Y) - \mathcal{H}(Y|X) &\text{(I.10)}\end{aligned}$$

In a metaphorical sense, $\mathcal{I}(X,Y)$ reflects the intersection of the information content of X and Y. Thus, it provides an incisive view into the shared information realm of these variables, which is pictorially illustrated via a Venn diagram (see Figure I.1) [71].

I.4 Relative Entropy and Divergence

Information theory, especially in the context of finite alphabets, elucidates several pivotal information measures that burgeon essential properties, crucially emanating from the

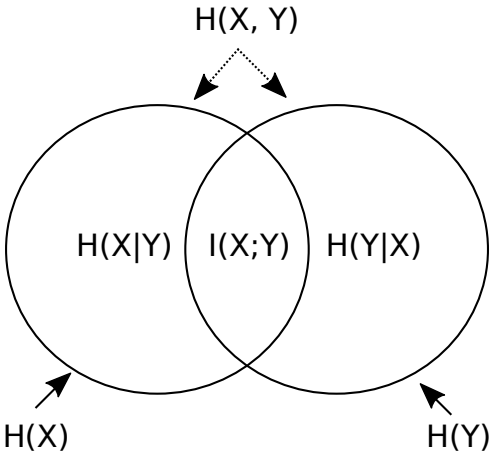


Figure I.1: Interplay between entropy and mutual information. Adapted from [71].

generalized divergences framework [132]. One such prominent measure is relative entropy, frequently denominated as divergence or Kullback-Leibler (KL) divergence.

KL divergence stands out as an instrumental measure in information theory, notwithstanding its non-compliance with specific mathematical properties typically associated with a distance measure. It does not assure symmetry and fails to satisfy the triangle inequality. Despite these mathematical peculiarities, it offers a sturdy methodology to gauge the dissimilarity between two probability distributions, represented as p_X and p_Y. Formally, the definition of relative entropy is provided as follows [71]:

$$\mathcal{KL}(p_X||p_Y) = \sup_{\mathcal{A}} \mathcal{KL}_{\mathcal{A}}(p_X||p_Y) \tag{I.11}$$

In the context of this equation, $\mathcal{A}$ symbolizes a finite alphabet of the sample space, while $\mathcal{KL}_{\mathcal{A}}$ denotes the KL divergence computed over the said alphabet. Furthermore, the operation $\sup_{\mathcal{A}}$ refers to the supremum (or the least upper bound) with respect to $\mathcal{A}$. To illustrate this, consider a set S defined as $S = \{x \in \mathbb{R} : x < 2\}$. Here, $\sup(S)$ is 2, given that 2 is the smallest real number greater than or equal to every number in S.

The depth of KL divergence extends further, elucidating its essence through a mathematical representation that defines divergence in the semblance of entropy, leveraged through the logarithmic ratio of the distributions [71]:

$$\mathcal{KL}(p_X||p_Y) = \sum_{\boldsymbol{x}\in\mathcal{A}} p_X(\boldsymbol{x}) \log \frac{p_X(\boldsymbol{x})}{p_Y(\boldsymbol{x})} \tag{I.12}$$

$$= \mathbb{E}_{p_X} \log \frac{p_X(\boldsymbol{x})}{p_Y(\boldsymbol{x})} \tag{I.13}$$

Here, $\mathbb{E}_{p_X}$ signifies the expected value concerning p_X. Further, it is pivotal to note that the non-negativity attribute of relative entropy entails that it is zero when $p_X = p_Y$,

portraying absolute similarity or a positive value, highlighting the disparity between the compared distributions. In the realm of image compression, this metric can serve as a critical indicator, illuminating the fidelity and efficacy of compression algorithms by manifesting the divergence from the original distribution, essentially providing a quantitative measure of the information loss during the compression process.

References

[1] J. Jiang, "Image compression with neural networks - A survey," *Signal Process. Image Commun.*, vol. 14, no. 9, pp. 737–760, 1999. 5, 18

[2] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952. 5, 18

[3] S. W. Golomb, "Run-length encodings (corresp.)," *IEEE Trans. Inf. Theory*, vol. 12, no. 3, pp. 399–401, 1966. 5, 18

[4] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, no. 6, pp. 520–540, 1987. 5, 18

[5] W. K. Pratt, J. Kane, and H. C. Andrews, "Hadamard transform image coding," *Proceedings of the IEEE*, vol. 57, no. 1, pp. 58–68, 1969. 5

[6] N. Ahmed, T. R. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Computers*, vol. 23, no. 1, pp. 90–93, 1974. 5, 18

[7] S. Ma, X. Zhang, C. Jia, Z. Zhao, S. Wang, and S. Wang, "Image and video compression with neural networks: A review," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 6, pp. 1683–1698, 2020. 5, 6, 18, 43, 45

[8] G. K. Wallace, "The JPEG still picture compression standard," *Commun. ACM*, vol. 34, no. 4, pp. 30–44, 1991. 5, 18, 63

[9] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The jpeg 2000 still image compression standard," *IEEE Signal processing magazine*, vol. 18, no. 5, pp. 36–58, 2001. 5, 18, 63

[10] D. S. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Process.*, vol. 9, no. 7, pp. 1158–1170, 2000. 5, 18

[11] F. Bellard, "The BPG image format." Online, April 21 2018. Available from `http://bellard.org/bpg/`. 5, 18

[12] F. Dufaux, G. J. Sullivan, and T. Ebrahimi, "The JPEG XR image coding standard [standards in a nutshell]," *IEEE Signal Process. Mag.*, vol. 26, no. 6, pp. 195–204, 2009. 5, 18

[13] A. Artusi, R. K. Mantiuk, T. Richter, P. Hanhart, P. Korshunov, M. Agostinelli, A. Ten, and T. Ebrahimi, "Overview and evaluation of the JPEG XT HDR image compression standard," *J. Real Time Image Process.*, vol. 16, no. 2, pp. 413–428, 2019. 6, 18

[14] J. Alakuijala, R. van Asseldonk, S. Boukortt, M. Bruse, Z. Szabadka, I.-M. Comșa, M. Firsching, T. Fischbacher, E. Kliuchnikov, S. Gomez, *et al.*, "Jpeg xl next-generation image compression architecture and coding tools," in *Applications of Digital Image Processing XLII*, vol. 11137, p. 111370K, International Society for Optics and Photonics, 2019. 6, 19

[15] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar, "Variable rate image compression with recurrent neural networks," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2016. 6, 7, 43, 44, 45, 48, 60

[16] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 5435–5443, IEEE Computer Society, 2017. 6, 8, 45, 46, 60

[17] M. Covell, N. Johnston, D. Minnen, S. J. Hwang, J. Shor, S. Singh, D. Vincent, and G. Toderici, "Target-quality image compression with recurrent, convolutional neural networks," *CoRR*, vol. abs/1705.06687, 2017. 6, 46, 60

[18] D. Minnen, G. Toderici, M. Covell, T. T. Chinen, N. Johnston, J. Shor, S. J. Hwang, D. Vincent, and S. Singh, "Spatially adaptive image compression using a tiled deep network," in *2017 IEEE International Conference on Image Processing, ICIP 2017, Beijing, China, September 17-20, 2017*, pp. 2796–2800, IEEE, 2017. 6, 46, 47, 60

[19] H. C. Jung, N. D. Guerin Jr, R. S. Ramos, B. Macchiavello, E. Peixoto, E. M. Hung, T. Campos, R. C. Silva, V. Testoni, and P. G. Freitas, "Multi-mode Intra Prediction for Learning-based Image Compression," in *International Conference on Image Processing (ICIP)*, Oct. 2020. 6

[20] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. T. Chinen, S. J. Hwang, J. Shor, and G. Toderici, "Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 4385–4393, IEEE Computer Society, 2018. 6, 46, 47, 60

[21] J. Ballé, V. Laparra, and E. P. Simoncelli, "Density modeling of images using a generalized normalization transformation," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2016. 7, 30, 31, 67, 69

[22] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2014. 7, 40, 41, 48, 65, 67, 68, 71, 103, 105

[23] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018. 7, 8, 30, 48, 49, 52, 55, 61, 65, 66, 67, 73, 74, 75, 77, 84, 99, 105, 108, 109, 115, 129, 143, 146

[24] D. Minnen, J. Ballé, and G. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada* (S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 10794–10803, 2018. 7, 49, 50, 51, 52, 55, 73, 75

[25] D. Minnen, G. Toderici, S. Singh, S. J. Hwang, and M. Covell, "Image-dependent local entropy models for learned image compression," in *2018 IEEE International Conference on Image Processing, ICIP 2018, Athens, Greece, October 7-10, 2018*, pp. 430–434, IEEE, 2018. 7, 50

[26] M. Li, K. Ma, J. You, D. Zhang, and W. Zuo, "Efficient and effective context-based convolutional entropy modeling for image compression," *IEEE Trans. Image Process.*, vol. 29, pp. 5900–5911, 2020. 7, 50, 51

[27] J. Zhou, S. Wen, A. Nakagawa, K. Kazui, and Z. Tan, "Multi-scale and context-adaptive entropy model for image compression," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019, Long Beach, CA, USA, June 16-20, 2019*, p. 0, Computer Vision Foundation / IEEE, 2019. 7, 50

[28] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017. 7, 51

[29] D. Minnen and S. Singh, "Channel-wise autoregressive entropy models for learned image compression," in *IEEE International Conference on Image Processing, ICIP 2020, Abu Dhabi, United Arab Emirates, October 25-28, 2020*, pp. 3339–3343, IEEE, 2020. 7, 51

[30] Y. Hu, W. Yang, and J. Liu, "Coarse-to-fine hyper-prior modeling for learned image compression," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 11013–11020, AAAI Press, 2020. 7, 51, 52, 61

[31] N. Johnston, E. Eban, A. Gordon, and J. Ballé, "Computationally efficient neural image compression," *CoRR*, vol. abs/1912.08771, 2019. 7, 52

[32] Y. Choi, M. El-Khamy, and J. Lee, "Variable rate deep image compression with a conditional autoencoder," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 3146–3154, IEEE, 2019. 7, 53

[33] C. Cai, L. Chen, X. Zhang, and Z. Gao, "Efficient variable rate image compression with multi-scale decomposition network," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 12, pp. 3687–3700, 2019. 7, 53

[34] Z. Cui, J. Wang, B. Bai, T. Guo, and Y. Feng, "G-VAE: a continuously variable rate deep image compression framework," *CoRR*, vol. abs/2003.02012, 2020. 7, 53, 61

[35] Y. Yang, R. Bamler, and S. Mandt, "Variable-bitrate neural compression via bayesian arithmetic coding," *CoRR*, vol. abs/2002.08158, 2020. 7, 53

[36] Y. Chen, H. Fan, B. Xu, Z. Yan, Y. Kalantidis, M. Rohrbach, S. Yan, and J. Feng, "Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 3434–3443, IEEE, 2019. 7, 54

[37] M. Akbari, J. Liang, J. Han, and C. Tu, "Generalized octave convolutions for learned multi-frequency image compression," *CoRR*, vol. abs/2002.10032, 2020. 7, 55, 61

[38] J. Lin, M. Akbari, H. Fu, Q. Zhang, S. Wang, J. Liang, D. Liu, F. Liang, G. Zhang, and C. Tu, "Variable-rate multi-frequency image compression using modulated generalized octave convolution," in *22nd IEEE International Workshop on Multimedia Signal Processing, MMSP 2020, Tampere, Finland, September 21-24, 2020*, pp. 1–6, IEEE, 2020. 7, 55

[39] M. Li, W. Zuo, S. Gu, J. You, and D. Zhang, "Learning content-weighted deep image compression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020. 7, 55, 60

[40] L. Zhou, Z. Sun, X. Wu, and J. Wu, "End-to-end optimized image compression with attention mechanism," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019, Long Beach, CA, USA, June 16-20, 2019*, p. 0, Computer Vision Foundation / IEEE, 2019. 7, 55

[41] T. Chen, H. Liu, Z. Ma, Q. Shen, X. Cao, and Y. Wang, "Neural image compression via non-local attention optimization and improved context modeling," 2019. 7, 55

[42] H. Liu, T. Chen, P. Guo, Q. Shen, X. Cao, Y. Wang, and Z. Ma, "Non-local attention optimized deep image compression," *CoRR*, vol. abs/1904.09757, 2019. 7, 55

[43] J. Lee, S. Cho, and M. Kim, "An end-to-end joint learning scheme of image compression and quality enhancement with improved entropy minimization," *arXiv preprint arXiv:1912.12817*, 2019. 7, 56, 61

[44] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. V. Gool, "Generative adversarial networks for extreme learned image compression," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 221–231, IEEE, 2019. 7, 56

[45] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017. 8, 30, 48, 49, 50, 52, 61, 65, 66, 67, 69, 70, 71, 72, 73, 75, 77, 84, 94, 99, 103, 108, 109, 115, 129, 143, 146

[46] T. van Rozendaal, G. Sautière, and T. S. Cohen, "Lossy compression with distortion constrained optimization," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*, pp. 634–639, Computer Vision Foundation / IEEE, 2020. 8, 56

[47] S. Zhang, L. Wang, X. Mao, F. Yang, and S. Wan, "Rate controllable learned image compression based on RFL model," in *IEEE International Conference on Visual Communications and Image Processing, VCIP 2022, Suzhou, China, December 13 - 16, 2022*, pp. 1–5, IEEE, 2022. 8, 57, 76

[48] 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*, vol. 101, p. 116544, 2022. 9, 31, 128, 129, 131, 132, 141, 142

[49] N. D. Guerin, R. C. da Silva, and B. Macchiavello, "Learning-based image compression with parameter-adaptive rate-constrained loss," *IEEE Signal Processing Letters*, vol. 31, pp. 1099–1103, 2024. 9

[50] I. E. Richardson, *Video Codec Design: Developing Image and Video Compression Systems*. USA: John Wiley & Sons, Inc., 2002. 11, 12, 13, 14, 15, 18

[51] K. Sayood, *Introduction to Data Compression, Third Edition*. The Morgan Kaufmann series in multimedia information and systems, Elsevier Morgan Kaufmann, 2006. 15

[52] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. Adaptive computation and machine learning, MIT Press, 2012. 19

[53] M. A. Nielsen, "Neural networks and deep learning," 2018. 20, 21

[54] U. Michelucci, *Applied Deep Learning: A Case-Based Approach to Understanding Deep Neural Networks*. USA: Apress, 1st ed., 2018. 22

[55] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. 22, 23, 24, 25, 26

[56] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986. 22

[57] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997. 26

[58] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL* (A. Moschitti, B. Pang, and W. Daelemans, eds.), pp. 1724–1734, ACL, 2014. 26

[59] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 802–810, 2015. 26, 27

[60] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 1874–1883, IEEE Computer Society, 2016. 26, 27

[61] A. Apicella, F. Donnarumma, F. Isgrò, and R. Prevete, "A survey on modern trainable activation functions," *Neural Networks*, vol. 138, pp. 14–32, 2021. 27, 28, 29, 30

[62] O. Schwartz and E. P. Simoncelli, "Natural signal statistics and sensory gain control," *Nature neuroscience*, vol. 4, no. 8, pp. 819–825, 2001. 30

[63] J. Shin, T. A. Badgwell, K.-H. Liu, and J. H. Lee, "Reinforcement learning – overview of recent progress and implications for process control," *Computers Chemical Engineering*, vol. 127, pp. 282–294, 2019. 31, 32

[64] A. Leon-Garcia, *Probability, Statistics, and Random Processes for Electrical Engineering*. Upper Saddle River, NJ: Pearson/Prentice Hall, 3 ed., 2008. 33, 34

[65] B. J. Brewer, "Stats 331 - introduction to bayesian statistics, lecture notes," 2017. Auckland University of Technology. Available on: `https://www.stat.auckland.ac.nz/~brewer/stats331.pdf`. 34

[66] S. O. Nyberg, *The Bayesian Way*. John Wiley & Sons, Ltd, 2018. 34

[67] J. Ghosh, M. Delampady, and T. Samanta, *An Introduction to Bayesian Analysis: Theory and Methods*. Springer Texts in Statistics, Springer New York, 2007. 35, 36

[68] D. Tzikas, A. Likas, and N. Galatsanos, "The variational approximation for bayesian inference life after the EM algorithm," *Signal Processing Magazine, IEEE*, vol. 25, pp. 131 – 146, 12 2008. 36

[69] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017. 37, 38, 39

[70] L. K. Saul, T. S. Jaakkola, and M. I. Jordan, "Mean field theory for sigmoid belief networks," *J. Artif. Intell. Res.*, vol. 4, pp. 61–76, 1996. 37

[71] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. USA: Wiley-Interscience, 2006. 38, 158, 160, 161, 162, 163

[72] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Found. Trends Mach. Learn.*, vol. 12, no. 4, pp. 307–392, 2019. 38, 39, 40, 41, 67, 68

[73] W. Han and Y. Yang, "Statistical inference in mean-field variational bayes," *arXiv preprint arXiv:1911.01525*, 2019. 48

[74] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 48

[75] R. W. Franzen, "Kodak image dataset," Updated on January 27 2013. Available from `http://r0k.us/graphics/kodak/`. 48, 99

[76] J. Ballé, P. A. Chou, D. Minnen, S. Singh, N. Johnston, E. Agustsson, S. J. Hwang, and G. Toderici, "Nonlinear transform coding," *IEEE J. Sel. Top. Signal Process.*, vol. 15, no. 2, pp. 339–353, 2021. 50, 61

[77] 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 2023, Ottawa, ON, Canada, 29 October 2023- 3 November 2023* (A. El-Saddik, T. Mei, R. Cucchiara, M. Bertini, D. P. T. Vallejo, P. K. Atrey, and M. S. Hossain, eds.), pp. 7618–7627, ACM, 2023. 50, 61

[78] Z. Duan, M. Lu, Z. Ma, and F. Zhu, "Lossy image compression with quantized hierarchical vaes," in *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2023, Waikoloa, HI, USA, January 2-7, 2023*, pp. 198–207, IEEE, 2023. 51, 61

[79] A. Luo, H. Sun, J. Liu, and J. Katto, "Memory-efficient learned image compression with pruned hyperprior module," in *2022 IEEE International Conference on Image Processing, ICIP 2022, Bordeaux, France, 16-19 October 2022*, pp. 3061–3065, IEEE, 2022. 52

[80] A. van den Oord, N. Kalchbrenner, L. Espeholt, K. Kavukcuoglu, O. Vinyals, and A. Graves, "Conditional image generation with pixelcnn decoders," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain* (D. D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, and R. Garnett, eds.), pp. 4790–4798, 2016. 53

[81] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 3483–3491, 2015. 53

[82] Z. Cui, J. Wang, S. Gao, T. Guo, Y. Feng, and B. Bai, "Asymmetric gained deep image compression with continuous rate adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 10532–10541, Computer Vision Foundation / IEEE, 2021. 53

[83] Y. Lu, Y. Zhu, Y. Yang, A. Said, and T. S. Cohen, "Progressive neural image compression with nested quantization and latent ordering," in *2021 IEEE International Conference on Image Processing, ICIP 2021, Anchorage, AK, USA, September 19-22, 2021*, pp. 539–543, IEEE, 2021. 53

[84] F. Baldassarre, A. El-Nouby, and H. Jégou, "Variable rate allocation for vector-quantized autoencoders," in *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2023, Rhodes Island, Greece, June 4-10, 2023*, pp. 1–5, IEEE, 2023. 53

[85] C. Kao, Y. Weng, Y. Chen, W. Chiu, and W. Peng, "Transformer-based variable-rate image compression with region-of-interest control," in *IEEE International Conference on Image Processing, ICIP 2023, Kuala Lumpur, Malaysia, October 8-11, 2023*, pp. 2960–2964, IEEE, 2023. 54

[86] J. Liang, M. Liu, C. Yao, C. Lin, and Y. Zhao, "SIGVIC: spatial importance guided variable-rate image compression," in *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2023, Rhodes Island, Greece, June 4-10, 2023*, pp. 1–5, IEEE, 2023. 54, 61

[87] K. Tong, Y. Wu, Y. Li, K. Zhang, L. Zhang, and X. Jin, "QVRF: A quantization-error-aware variable rate framework for learned image compression," in *IEEE International Conference on Image Processing, ICIP 2023, Kuala Lumpur, Malaysia, October 8-11, 2023*, pp. 1310–1314, IEEE, 2023. 54

[88] Z. Duan, M. Lu, J. Ma, Y. Huang, Z. Ma, and F. Zhu, "Qarv: Quantization-aware resnet vae for lossy image compression," *IEEE Transactions on Pattern Analysis amp; Machine Intelligence*, vol. 46, pp. 436–450, jan 2024. 54

[89] S. Cai, Z. Zhang, L. Chen, L. Yan, S. Zhong, and X. Zou, "High-fidelity variable-rate image compression via invertible activation transformation," in *MM '22: The 30th*

ACM International Conference on Multimedia, Lisboa, Portugal, October 10 - 14, 2022 (J. Magalhães, A. D. Bimbo, S. Satoh, N. Sebe, X. Alameda-Pineda, Q. Jin, V. Oria, and L. Toni, eds.), pp. 2021–2031, ACM, 2022. 54, 61

[90] J. Lee, S. Jeong, and M. Kim, "Selective compression learning of latent representations for variable-rate image compression," in *NeurIPS*, 2022. 54

[91] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017. 55

[92] Y. Zhang, K. Li, K. Li, B. Zhong, and Y. Fu, "Residual non-local attention networks for image restoration," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019. 55

[93] A. A. Jeny, M. B. Islam, M. S. Junayed, and D. Das, "Improving image compression with adjacent attention and refinement block," *IEEE Access*, vol. 11, pp. 17613–17625, 2023. 55, 61

[94] C. Huang, H. Liu, T. Chen, Q. Shen, and Z. Ma, "Extreme image coding via multiscale autoencoders with generative adversarial optimization," in *2019 IEEE Visual Communications and Image Processing, VCIP 2019, Sydney, Australia, December 1-4, 2019*, pp. 1–4, IEEE, 2019. 56

[95] Z. Guo, Z. Zhang, R. Feng, and Z. Chen, "Soft then hard: Rethinking the quantization in neural image compression," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 3920–3929, PMLR, 2021. 57

[96] S. Pan, C. Finlay, C. Besenbruch, and W. J. Knottenbelt, "Three gaps for quantisation in learned image compression," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2021, virtual, June 19-25, 2021*, pp. 720–726, Computer Vision Foundation / IEEE, 2021. 57

[97] J. Shin, T. A. Badgwell, K. Liu, and J. H. Lee, "Reinforcement learning - overview of recent progress and implications for process control," *Comput. Chem. Eng.*, vol. 127, pp. 282–294, 2019. 58

[98] X. Zhe, L. Ou-Yang, and H. Yan, "Improve l2-normalized softmax with exponential moving average," in *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*, pp. 1–7, IEEE, 2019. 58

[99] J. Zhang, Q. Guo, Y. Dong, F. Xiong, and H. Bai, "Adaptive parameters softmax loss for deep face recognition," in *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, pp. 1680–1684, 2019. 58

[100] L. Huang, C. Zhang, and H. Zhang, "Self-adaptive training: Bridging the supervised and self-supervised learning," *CoRR*, vol. abs/2101.08732, 2021. 58

[101] M. Xu, Z. Zhang, H. Hu, J. Wang, L. Wang, F. Wei, X. Bai, and Z. Liu, "End-to-end semi-supervised object detection with soft teacher," in *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 3040–3049, IEEE, 2021. 58

[102] O. Rippel and L. D. Bourdev, "Real-time adaptive image compression," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 2922–2930, PMLR, 2017. 59, 60

[103] M. Akbari, J. Liang, and J. Han, "DSSLIC: deep semantic segmentation-based layered image compression," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pp. 2042–2046, IEEE, 2019. 59, 60

[104] J. Sneyers and P. Wuille, "Flif: Free lossless image format based on maniac compression," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 66–70, 2016. 59

[105] F. Bellard, "Bpg image format." `https://bellard.org/bpg/`, 2014. Retrieved 15 de maio de 2024. 59

[106] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. V. Gool, "Practical full resolution learned lossless image compression," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 10629–10638, Computer Vision Foundation / IEEE, 2019. 59, 60

[107] F. Mentzer, L. V. Gool, and M. Tschannen, "Learning better lossless compression using lossy compression," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 6637–6646, IEEE, 2020. 60

[108] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *Advances in Neural Information Processing Systems*, pp. 10771–10780, 2018. 61

[109] Y. M. Yeung and O. C. Au, "Efficient rate control for JPEG2000 image coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 3, pp. 335–344, 2005. 63

[110] S. Ma, W. Gao, and Y. Lu, "Rate-distortion analysis for H.264/AVC video coding and its application to rate control," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 12, pp. 1533–1544, 2005. 63

[111] Z. Chen and K. N. Ngan, "Recent advances in rate control for video coding," *Signal Process. Image Commun.*, vol. 22, no. 1, pp. 19–38, 2007. 63, 64

[112] J. Domke and D. R. Sheldon, "Importance weighting and variational inference," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada* (S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 4475–4484, 2018. 65

[113] Johannes Ballé and Sung Jin Hwang and Nick Johnston and David Minnen, "Tensorflow compression." Available from `https://tensorflow.github.io/compression/`. 66, 98

[114] J. Ballé, "Efficient nonlinear transforms for lossy image compression," 2018. 67

[115] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2325–2383, 1998. 71

[116] C. M. Bishop, "Latent variable models," in *Learning in graphical models*, pp. 371–403, Springer, 1998. 74

[117] C. Lemaréchal, "Lagrangian relaxation," in *Computational combinatorial optimization*, pp. 112–156, Springer, 2001. 80

[118] M. Basseville, I. V. Nikiforov, *et al.*, *Detection of abrupt changes: theory and application (Prentice-Hall information and system sciences series)*, vol. 104. Prentice Hall, 1993. 96

[119] "Dataset of the CVPR workshop and challenge on learned image compression (CLIC)." Available from `http://www.compression.cc`. 98, 140

[120] E. Agustsson and R. Timofte, "NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017. DIV2K dataset: DIVerse 2K resolution high quality images as used for the challenges at NTIRE (CVPR 2017 and CVPR 2018) and at PIRM (ECCV 2018), Available from `https://data.vision.ee.ethz.ch/cvl/DIV2K/`. 98, 140

[121] H. Nemoto, P. Hanhart, P. Korshunov, and T. Ebrahimi, "Ultra-Eye: UHD and HD Images Eye Tracking Dataset," in *Sixth International Workshop on Quality of Multimedia Experience (QoMEX)*, (Singapore), September 2014. Available from `http://mmspg.epfl.ch/ultra-eye`. 98, 141

[122] L. Jin, J. Y. Lin, S. Hu, H. Wang, P. Wang, I. Katsavounidis, A. Aaron, and C.-C. J. Kuo, "MCL-JCI Dataset." Available from `http://mcl.usc.edu/mcl-jci-dataset/`. 98, 141

[123] Lina Jin, J. Y. Lin, S. Hu, H. Wang, P. Wang, I. Katsavounidis, A. Aaron, and C.-C. J. Kuo, "Statistical Study on Perceived JPEG Image Quality via MCL-JCI Dataset Construction and Analysis," in *Electronic Imaging (2016), the Society for Imaging Science and Technology (IS&T)*, 2016. 98, 141

[124] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 1132–1140, IEEE Computer Society, 2017. 98, 141

[125] JPEG-AI Call for Evidence - IEEE MMSP2020 Challenge (Dataset), 2020. Available from `https://jpegai.github.io/test_images/`. 99

[126] U. Sara, M. Akter, and M. S. Uddin, "Image quality assessment through fsim, ssim, mse and psnr—a comparative study," *Journal of Computer and Communications*, vol. 7, no. 3, pp. 8–18, 2019. 111

[127] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, vol. 2, pp. 1398–1402, Ieee, 2003. 111

[128] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778, IEEE Computer Society, 2016. 115

[129] M. H. Hyun, B. Lee, and M. Kim, "A frame-level constant bit-rate control using recursive bayesian estimation for versatile video coding," *IEEE Access*, vol. 8, pp. 227255–227269, 2020. 118

[130] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015. 138

[131] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948. 158, 159

[132] R. M. Gray, *Entropy and Information Theory*. Springer Publishing Company, Incorporated, 2 ed., 2011. 158, 159, 160, 161, 163