PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

University of Mohamed El-Bachir El-Ibrahimi–Bordj Bou Arreridj
Faculty of Science and Technology
Department of Electronic

Thesis

Presented for the fulfillment of a MASTER'S degree

Presented to obtain

Sector: Telecommunication

Specialty:

Telecommunications System

By

Ms. Saoudi Khadidja.

Ms. Boutahar Amel.

Entitled

Study of image compression techniques based on deep learning.

Supported on:

By the evaluation committee composed of

First name & family name:	***Grade***	***Quality***	***Etablissement***
Ms. N. MELLIZI	***MAA***	***Chairman***	***Univ-BBA***
Mr. D.E. BOUDECHICHE	***MCA***	***Supervisor***	***Univ-BBA***
Mr. N. ASBAI	***MCA***	***Examiner***	***Univ-BBA***


بسم الله الرحمن الرحيم

Acknowledgments

{ وَقُلْ رَبِّ زِدْنِي عِلْمًا}

First and foremost, we would like to express our heartfelt gratitude to ourselves for the perseverance, effort, and dedication we invested throughout this project. This journey was filled with challenges, but our commitment and teamwork made it an entertaining experience.

We extend our sincere gratitude to our supervisor, Dr. Boudechiche Djamel Eddine, for his invaluable guidance, insightful feedback, and unwavering support throughout all stages of this thesis. His mentorship greatly contributed to the development and successful completion of our work.

We are truly thankful to Saoudi Rania for her continued guidance and assistance.

Our heartfelt thanks also go to Dr. Djoudi Al Ayachi, whose assistance in configuring and downloading MATLAB was instrumental to the progress of our research. His generosity with his time and expertise is deeply appreciated.

We would also like to acknowledge everyone who supported us throughout this academic journey: our families, friends, and colleagues, whose encouragement, assistance, and belief in us were invaluable to the successful completion of this work.

Dedicace

{الحمد الله حبا وشكرا وامتنان على البدء والختام}

{ و ما سلكنا البدايات إلا تيسيره وما بلغنا النهايات إلا بتوفيقه و ما حققنا الغايات إلا بفضله}

First and foremost, we would like to express our sincere gratitude to ourselves for the perseverance, dedication, and hard work that made this achievement possible.

A special note of thanks is reserved for our parents and for their unconditional love, sacrifices, and endless support, which have been the foundation of our success.

Our heartfelt thanks also go to our siblings, who have always believed in us and encouraged us throughout every step of this journey.

We would like to warmly thank our friends for their continuous encouragement, kindness, and motivation during challenging times.

We would like to extend our heartfelt thanks to our families for their unwavering support, encouragement, and patience throughout the course of this project.

To the souls of the derest personnes I have lost, my uncle Hammouche Ismail and my aunt Baya {May Allah grant him jannat al-Firduas}, “Amel.”

To all who supported, inspired, and believed in us.

Thank you all.

Abstract

This thesis explores various image compression techniques, with particular focus on approaches grounded in deep learning. It presents detailed analysis and comparison between traditional compression algorithms such as JPEG, JPEG 2000, and BPG and modern deep learning methods, including factorized and hyperprior models. While conventional techniques have been widely used for their balance of image quality and bitrate, recent advances in deep learning offer more effective and efficient alternatives. Neural network-based models, in particular, have demonstrated superior capabilities in achieving higher compression rates while maintaining perceptual image quality. This study underscores the strengths of these advanced techniques, establishing deep learning as a promising and powerful direction for the future of image compression.

Résumés

Cette mémoire explore différentes techniques de compression d'images, en mettant particulièrement l'accent sur les approches basées sur l'apprentissage profond (deep learning). Il présente une analyse approfondie et une comparaison entre les algorithmes de compression traditionnels tels que JPEG, JPEG2000 et BPG et les méthodes modernes fondées sur le deep learning, notamment les modèles factorisés et à hyperprior. Bien que les techniques classiques soient largement utilisées pour leur compromis entre qualité d'image et débit binaire, les avancées récentes en deep learning offrent des alternatives plus efficaces et performantes. Les modèles basés sur les réseaux de neurones ont démontré une capacité supérieure à atteindre des taux de compression plus élevés tout en préservant la qualité perceptuelle des images. Cette étude met en évidence les atouts de ces approches avancées, positionnant le deep learning comme une voie prometteuse et puissante pour l'avenir de la compression d'images.

ملخص

تتناول هذه الرسالة تقنيات ضغط الصور، مع التركيز بشكل خاص على الأساليب القائمة على التعلم العميق Deep Learning تقدم الدراسة تحليلاً مفصلاً ومقارنة بين الخوارزميات التقليدية لضغط الصور مثل JPEG و JPEG 2000و BPG والأساليب الحديثة المبنية على التعلم العميق، بما في ذلك النماذج العاملية (Factorized) ونماذج Hyperprior على الرغم من أن التقنيات التقليدية معروفة بتوازنها بين جودة الصورة ومعدل البت، فإن التطورات الأخيرة في مجال التعلم العميق توفر بدائل أكثر كفاءة وفعالية. لقد أثبتت النماذج المعتمدة على الشبكات العصبية قدرتها الفائقة على تحقيق معدلات ضغط أعلى مع الحفاظ على الجودة البصرية للصور. تسلط هذه الدراسة الضوء على مزايا هذه الأساليب المتقدمة، مما يجعل من التعلم العميق توجهاً واعداً وقوياً لمستقبل ضغط الصور.

List of contents

List of figures

List of tables

List of abbreviations

ANN: Artificial Neural Networks.

AVC: Advanced Video Coding.

BPG: Better Portable Graphics.

BPP Bits: Bit Per Pixel.

CNNs: Convolutional Neural Networks.

DCT: Discrete Cosine Transforms.

DL: Deep Learning.

DWT: Discrete Wavelet Transform.

GANs: Generative Adversarial Networks.

GDN: Generalized Divisive Normalization.

GPUs: Graphics Processing Units.

$\mathbf{g_a}$: Analysis Transform.

$\mathbf{g}_s$: Synthesis Transform.

HEVC: High Efficiency Video Coding.

IDCT: Inverse Discrete Cosine Transform.

IGDN: Inverse Generalized Divisive Normalization.

JPEG: Joint Photographic Experts Group.

JPEG 2000: Joint Photographic Experts Group 2000.

JVT: Joint Video Team.

MSE: Mean Squared Error.

NLP: Normalized Laplacian Pyramid.

PSNR: Peak Signal-To-Noise Ratio.

QTMT: Quad Tree Multi Type.

RD: Rate Distortion.

RGB: Red, Green, Blue.

RNNs: Recurrent Neural Networks.

SSIM: Structural Similarity Index Measure.

VAEs: Variational Autoencoders.

VTM: Versatile Video Coding Test Model.

VVC: Versatile Video Coding.

General Introduction

The modern era is characterized by significant progress in artificial intelligence. Deep learning has emerged as one of the most influential breakthroughs, offering powerful tools that require extensive research, ongoing refinement, and deep domain expertise. Its capacity to extract complex patterns and representations has led to transformative applications across a wide range of fields. In particular, the enhancement of deep learning techniques in the area of image compression has become a key focus, aiming to improve the efficiency and quality of visual data encoding and transmission.

With the rapid growth of digital media and the increasing demand for efficient data handling**,** key challenges in the field of image compression are to reconcile high visual quality with file size reduction. Indeed, increasing an image's quality often results in larger data volume, which can negatively affect the smoothness of its processing and transmission. Therefore, it is essential to develop methods capable of compressing images while preserving their quality, thus ensuring efficient delivery without compromising the visual experience.

Firstly, we examine the fundamental importance of encoding and compression algorithms in the effective management and transmission of digital data. By integrating advanced encoding methods with contemporary compression standards such as JPEG2000, H.265, and VVC, significant improvements in storage efficiency, media quality, and bandwidth optimization can be achieved.

Secondly, we highlight the transformative role of deep learning in advancing image compression, with particular emphasis on convolutional neural networks (CNNs) and variational autoencoders (VAEs) enhanced by hyperpriors. The foundational principles of neural networks and autoencoders serve as the basis for these advancements, enabling modern compression techniques to outperform traditional methods such as JPEG and JPEG2000. These innovative approaches achieve superior visual quality and greater compression efficiency by more effectively modeling spatial dependencies within latent representations.

Finally, we apply a thorough evaluation comparing traditional image compression algorithms with deep learning based models. By employing standardized performance metrics and visual quality assessments on benchmark datasets, we demonstrate that methods such as factorized

and hyperprior significantly enhance compression efficiency and reconstruction quality, particularly at lower bitrates.

Chapter 1

Coding and Image Compression Algorithms

Summary

This chapter explores about how coding and compressing algorithms are important for managing and sending transmitting data. Encoding techniques, combined with modern compression standards (**JPEG2000, H.265, VVC**), maximize storage efficiency, enhance media quality, and minimize bandwidth consumption.

Table of Contents:

1.1 Introduction

Coding algorithms are fundamental methods in computer science used to optimally transform data for various purposes, including lossless compression, encryption, and error correction [1]. Image compression enables file size reduction while preserving essential visual information. This can be either lossless or lossy, depending on whether all original data is retained. Among the best-known techniques utilized for entropy coding are Huffman coding and arithmetic coding, which leverage the statistical properties of data to reduce size or improve efficiency, sometimes taking advantage of human perception [2].

This chapter introduces fundamental concepts of coding algorithms, focusing on three essential entropy coding techniques, as well as image compression, which relies on advanced methods.

1.2 Coding Algorithms

Coding algorithms aim to optimize data representation and transmission. In this section, we explored three essential entropy coding techniques: “Huffman coding”, “arithmetic coding”, and “Golomb Rice coding”, which are used to reduce data size, improve transmission efficiency, and ensure reliability. These methods are widely applied in image compression.

1.2.1 Huffman coding

Huffman coding, proposed by David Huffman in 1952 [3], is a lossless data compression technique that assigns shorter codes to more frequent symbols and longer codes to rarer symbols, aiming to optimize storage space.

The algorithm begins by sorting symbols based on their probability of occurrence, then gradually merges the least frequent symbols into a binary tree. A unique binary code is then assigned to each symbol according to its path in the tree, with '0' assigned to the left branch and '1' to the right branch.

This coding is prefix-free, ensuring that no code is the prefix of another, allowing for unambiguous decoding. By reducing the average length of encoded words, Huffman coding enables efficient data compression [3].

The steps of the algorithm are as follows:

1. Sort the symbols in descending order according to their probability of occurrence.
2. Merge the two symbols with the lowest probabilities into new nodes and readjust the list.
3. Repeat this process until a single root node is obtained.
4. Assign binary codes by assigning '0' to the left branch and '1' to the right branch.
5. Concatenate the obtained codes at each level to define a unique code for each symbol [4].

Huffman coding generates variable-length binary codes on whole bits, enabling efficient compression by reducing the size of stored or transmitted data while ensuring the faithful restoration of the original message.

Example:

Consider source X with five symbols: "x_1, x_2, x_3, x_4, x_5", whose associated probabilities are given in the following table:

Table 1.1. Probabilities of occurrence of symbols [4].

symbols	P(x_i)
x_1	0.25
x_2	0.45
x_3	0.15
x_4	0.10
x_5	0.05

1. Construct a Huffman code for X.

Solution:

1. Figure "1" illustrates the steps for constructing Huffman code for the source X:

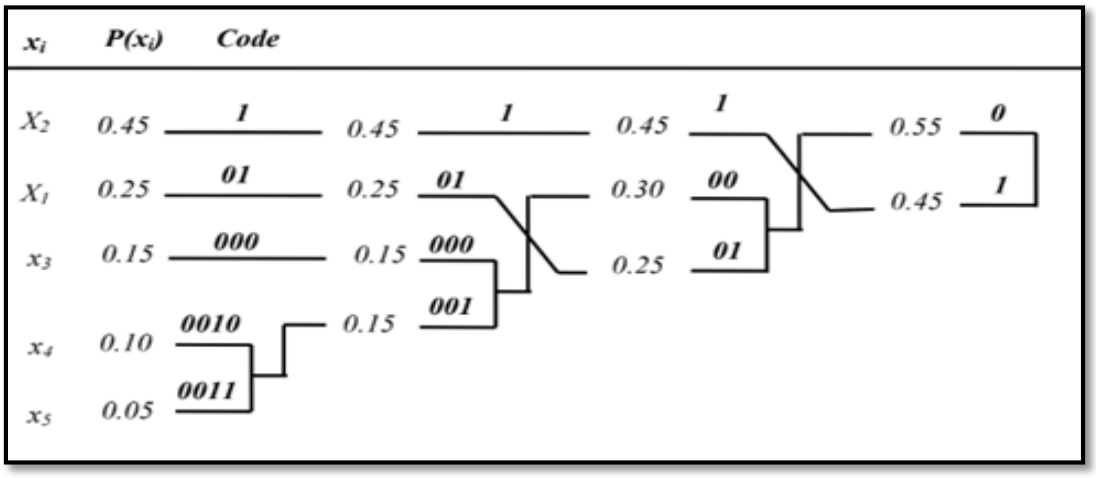

Figure 1.1. Huffman coding tree [4].

1.2.2 Arithmetic coding

Arithmetic coding is a lossless compression method that is more efficient than Huffman coding. It represents a sequence of symbols as a single number between '**0**' and '**1**', optimizing bit usage. Unlike traditional methods, it does not assign fixed code to each symbol but instead encodes an entire sequence by progressively dividing an interval into sub intervals proportional to the probabilities of the symbols [4].

This approach avoids the constraints of variable-length codes like **Huffman** and **Shannon-Fano** by allowing representation with non-integer numbers of bits. It is widely used in modern compression applications, providing more compact and efficient encoding. Instead of assigning a code to each individual symbol, arithmetic coding generates a single code for the entire sequence, refining the interval at each step based on the probabilities of the input symbols [5].

Arithmetic Coding Algorithm:

1. **Calculate probabilities**: Determine the probability of each symbol in the string to be encoded.
2. **Assign subintervals**: Assign each symbol a subinterval within [0, 1] proportional to its probability (order must be preserved for decoding).
3. **Initialize bounds**:
   - Set **lower bound = 0 and upper bound** = 1.
4. **Encoding process (loop)**:
   - Compute the interval width: **width = upper bound - lower bound**.
   - Update the lower bound:

     lower bound **= lower bound + width × (lower limit of the symbol's sub interval)**.

   - Update the upper bound:

     upper bound **= lower bound + width × (upper limit of the symbol's sub interval)**.

5. Repeat until all symbols are encoded [5].

Example:

Let X be a discrete source with an alphabet of three symbols, "**a, b, c**" whose associated occurrence probabilities are P(a) = 0.6**, P(b) = 0.2, and P(c) = 0.2.**

Given the sequence **M = 'abcaa'**.

- Encode the sequence **M** using the arithmetic coding algorithm.

Solution:

The initial intervals associated with the three symbols are shown in Table 2:

Table 1.2. Initial intervals associated with symbols **[4].**

Symbols	**A**	**b**	**c**
probability	0.6	0.2	0.2
Initial interval	[0, 0.6]	[0.6,0.8]	[0.8,1]

The flow of the arithmetic coding process is illustrated in Table 3:

Table 1.3. Progress of the arithmetic coding process of the string M='abcaa' **[4]**.

symbols	**Width**	**lower bound**	**Upper bound**
a	1	0	0.6
b	0.6	0.36	0.48
c	0.12	0.4560	0.48
a	0.024	0.4560	0.4704
a	0.0144	0.4560	0.46464

So the flattering number is between [0.4560, 0.46464].

Arithmetic coding allows for the compact representation of real number intervals in binary. To convert the interval **[0.4560, 0.46464]** into binary, two main methods can be used.

The first method involves multiplying the fractional part by **2** and extracting successive bits. This approach gives **"0.0111010010"** for **0.4560** and **"0.0111010111"** for **0.46464**. The smallest common binary number within this interval is **"0.01110101."**

Another way to obtain this representation is by using negative powers of **2**, where ($2^{-1} = 0.5. 2^{-2} = 0.25$ ... .), etc. A fractional number can thus be decomposed into sum of these powers. For example, **0.4560** can be expressed as:

0.4560=0.25+0.125+0.0625+0.015625+0.00390625+0.0009765625

Which corresponds to **0.0111010010** in binary.

In conclusion, the interval **[0.4560, 0.46464]** is represented in binary as **"0.01110101,"** which is the shortest code belonging to this range.

1.2.3 Golomb Rice Codes

Golomb Rice coding, developed by "**Solomon Wolf Golomb"** and later refined by **"Robert Floyd Rice"**, is a prefix coding method optimized for data following an exponential distribution. It is widely used in data compression. The encoding process is based on parameter M, which determines how numbers are split into two parts: the quotient, encoded using unary coding, and the remainder, encoded using binary representation. When M is the power of two, encoding is simplified. Despite its efficiency for certain distributions, Golomb-Rice coding is not always optimal for finite alphabets [6].

Golomb Rice Algorithm:

1. **Fix Parameter m**: Choose an integer value for m.
2. **Compute Quotient and Remainder**:

$$q = int\left[\frac{n}{m}\right] \tag{1.1}$$

$$r = n - qm \tag{1.2}$$

3. **Generate Code Word**: Format: "Unary Code" + "Remainder Code".
4. **Unary Code**: Encode the quotient as a sequence of q **"ones**" followed by a **"0".**
5. **Remainder Code**:
   - **If m is the power of 2**, encode the remainder in binary using $[log_2 m]$ bits.
   - **If m is not a power of 2:**
     - Compute $x = [log_2 m]$
     - If **r** < $2^{[log_2 m]} - m$ encode the remainder using **x-1** bits.
     - Otherwise, encode (r+$2^{[log_2 m]} - m$) using **x** bits.

6. Golomb Rice encoding is optimal for geometric probability distribution and is based on parameter "m" that splits each number into quotient and remainder [7].

$$p(n) = (1-p)^n \tag{1.3}$$

When

$$m = \left[-\frac{1}{log_2 p}\right] \tag{1.4}$$

Example:

For "**m=4**", calculate the quotient and the remainder, then deduce the bit stream.

Solution:

- The quotient (**q**) is encoded in unary: "**q**" times the bit "1" followed by a "0."
- The remainder **(r)** is represented using 2 binary bits (since m = 4 and $log_2 4 = 2$).

✓ **For n=0:**

Step 1: Calculating the quotient

$$q = int\left[\frac{0}{4}\right] = 0$$

Step 2: Calculating the remainder

$$r = 0 - (0 * 4) = 0$$

Step 3: Building the bitstream

Quotient in unary encoding**: 0**

Remainder encoded on 2 bits: **00**

➢ **Final bit stream: 0 00**

✓ **For n=1:**

The quotient is encoded in unary as: **0**

The remainder is encoded on 2 bits as: **01**

➢ **Final bit stream: 0 01**

And finally, this table illustrates the calculation process:

Table 1.4. Golomb rice encoding [6].

M[n]	Quotient	Remainder	Bit stream
0	0	0	0_00
1	0	1	0_01
2	0	2	0_10
3	0	3	0_11
4	1	0	10_00
5	1	1	10_01
6	1	2	10_10
7	1	3	10_11
8	2	0	110_00
9	2	1	110_01
10	2	2	110_10
11	2	3	110_11
12	3	0	1110_00
13	3	1	1110_01
14	3	2	1110_10
15	3	3	1110_11
16	4	0	11110_00

1.3 Image Compression

Traditional image compression is the cornerstone of digital media processing, enabling the reduction of file sizes without excessively sacrificing visual quality. It relies on advanced techniques that eliminate redundancies and less perceptible details. Popular formats like **JPEG**, **JPEG 2000**, **intra H.264**, **intra H.265** (BPG), and **intra H.266** (VTM) illustrate this evolution, using methods such as Discrete Cosine Transform (DCT), wavelets, or motion prediction. These standards are essential for optimizing the storage and transmission of images and videos in modern applications.

1.3.1 Definition of an image

An image is composed of a grid of small dots called pixels. Each pixel represents the smallest visible unit and contains the necessary information to display a portion of the image. In black and white images, pixels can be encoded using 1 bit: 0 for black, 1 for white. For grayscale or color images, the encoding is more complex and may use 2 to 32 bits per pixel, allowing for a wider range of shades and details. The number of bits used directly affects the quality and precision of the image [5].

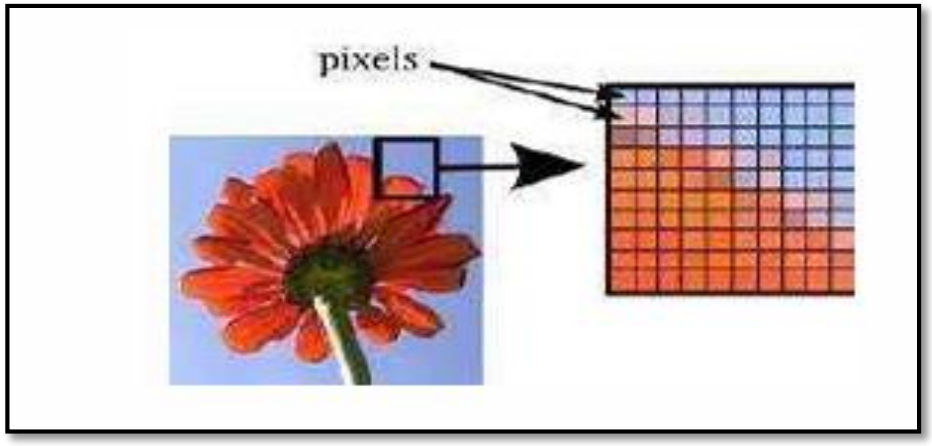

Figure1.2. Structure of an image [5].

1.3.2 General principle of image compression

The compression process is illustrated by a functional diagram, as shown in Figure 1.3:

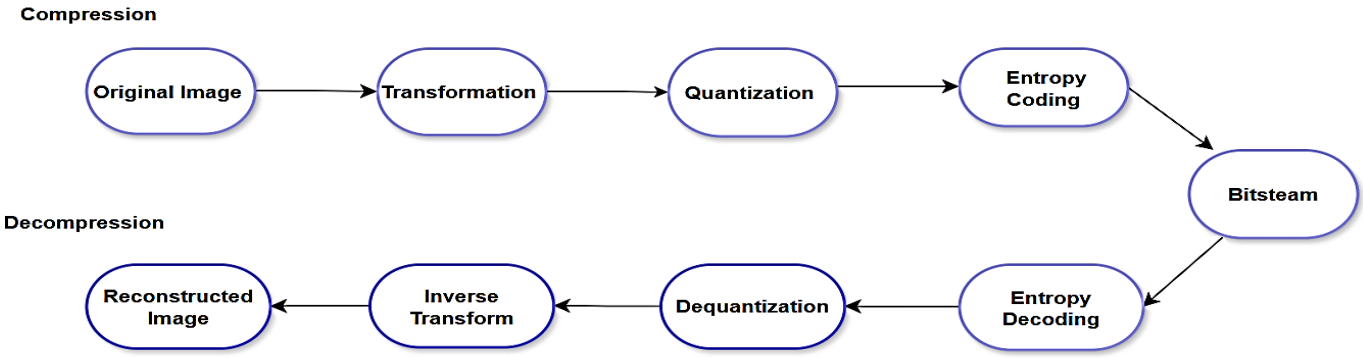

Figure1.3. Block diagram of image compression and decompression.

➢ **Compression**:

- **Transformation:**

In an image, pixels are highly correlated with their neighbors because brightness varies little from one point to another. To reduce the amount of information, a decorrelation operation is applied, transforming the pixels into a set of less dependent coefficients. This transformation is reversible [8].

- **Quantization:**

The quantization of coefficients aims to reduce the number of bits required for their representation, making it a key step in compression. It involves rounding each signal value to an integer multiple of a unit called the elementary quantum or quantization step**.** There are two types of quantization: scalar and vector, with the latter generally offering better performance [8].

- **Entropy Encoding:**

After quantization, the coefficients are encoded while meeting certain essential conditions. First, the encoding must ensure uniqueness, meaning that each distinct message has a unique code. Secondly, it must guarantee decipherability, allowing two successive codes to be distinguished without ambiguity. Various encoding techniques will be detailed later [8].

➢ **Decompression**:

Decompression is performed by reversing the steps of compression. It begins with decoding the compressed data, which allows the retrieval of the quantized symbols representing the model. In the case of lossy compression, the dequantization step is then applied to reconstruct an approximate version of the original model. However, the loss of information introduced during

quantization makes it impossible to fully recover the original data after inverse modeling [9].

1.3.3 JPEG Standard

The **JPEG** standard, created in the late 1980s by **ISO/ITU-T**, is a method for compressing still color images. It is primarily based on the Discrete Cosine Transform **(DCT)** and reduces file size while minimizing quality loss. Although it is a lossy compression, the degradation is generally acceptable for most applications [10].

JPEG Process Steps :

1. **Color conversion**: transformation from "**RGB"** to "**YCbCr"**.
2. **Image segmentation** into **8 × 8** pixel blocks.
3. **DCT application** to separate image frequency components.
4. **Quantization** to reduce coefficient precision.
5. **Entropy encoding** to further compress the data.
6. **Image reconstruction** using the Inverse Discrete Cosine Transform **(IDCT**) [10].

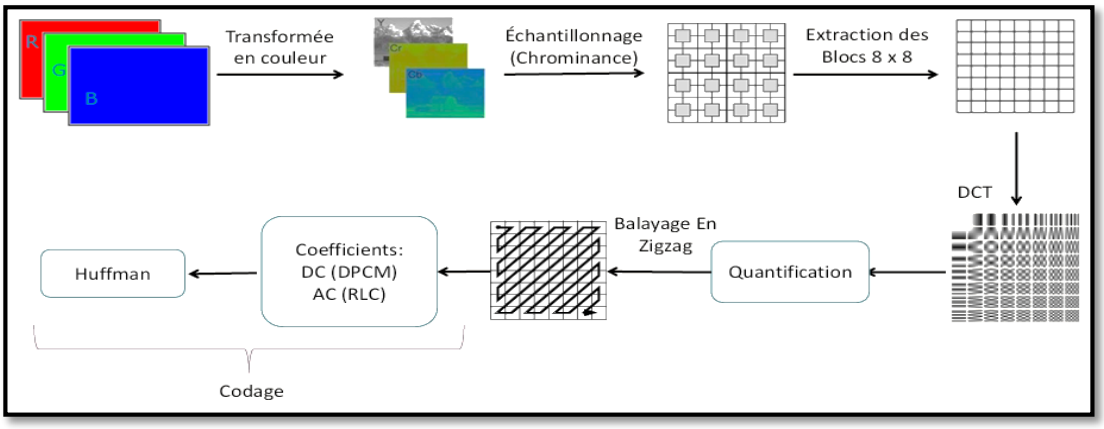

Figure 1.4. Standard JPEG block diagram [11].

The JPEG standard offers two types of compression:

1. **Lossy compression**: Uses DCT followed by quantization and entropy coding.
2. **Lossless compression**: Relies solely on entropy coding without image transformation [8].

For lossy compression, four encoders are defined:

- **Baseline encoding,** which applies DCT and Huffman coding but introduces some loss of information.

- **Three extended versions** improving efficiency through arithmetic coding or progressive image reconstruction [8].

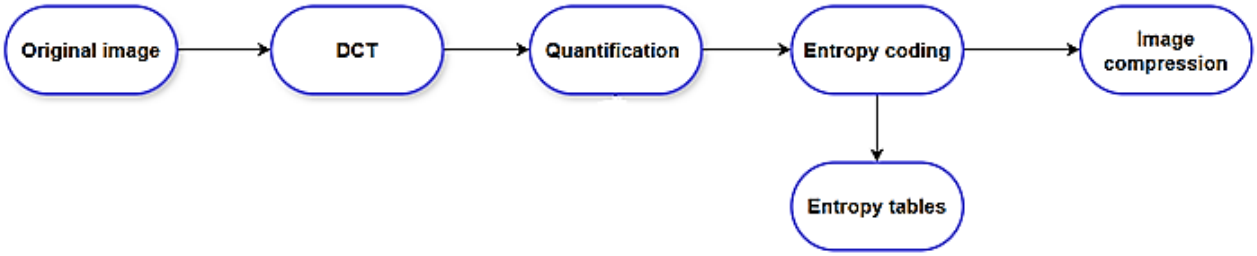

Figure 1. 6. Fundamentals of the JPEG lossy compression process.

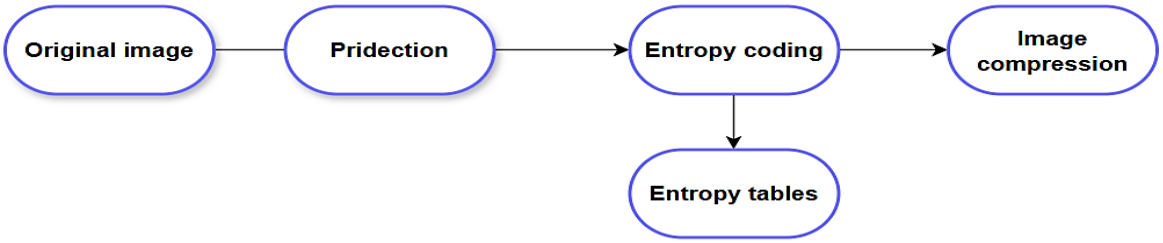

Figure 1.5. Fundamentals of the JPEG lossless compression process**.**

1.3.4 JPEG 2000

JPEG 2000 is an advanced image compression format that offers better quality and greater flexibility than traditional JPEG. It is based on Discrete Wavelet Transform **(DWT)**, enabling both **lossless** and lossy compression, along with optimized data flow management [12]**.**

Operating Principle:

JPEG 2000 works in several steps:

Discrete Wavelet Transform (DWT)

- The image is decomposed into multiple frequency sub-bands.
- Two types of filters are used:
  - **Le Gall (5,3)**: Reversible, allows for **lossless compression.**
  - **Daubechies (9,7)**: Non-reversible, provides **higher compression** but with some loss [12].

Segmentation into Code Blocks:

- Each sub-band is divided into **64×64 pixel blocks**, called **code blocks.**

- These blocks are processed independently, making selective data access easier [12].

Quantization and Encoding:

- **Scalar quantization** Adjusts data precision to optimize compression.
- **Arithmetic coding with contextual modeling ensures** efficient data compression.
- **Bit plane encoding** allowing for progressive data organization [12].

Comparison between JPEG and JPEG 2000:

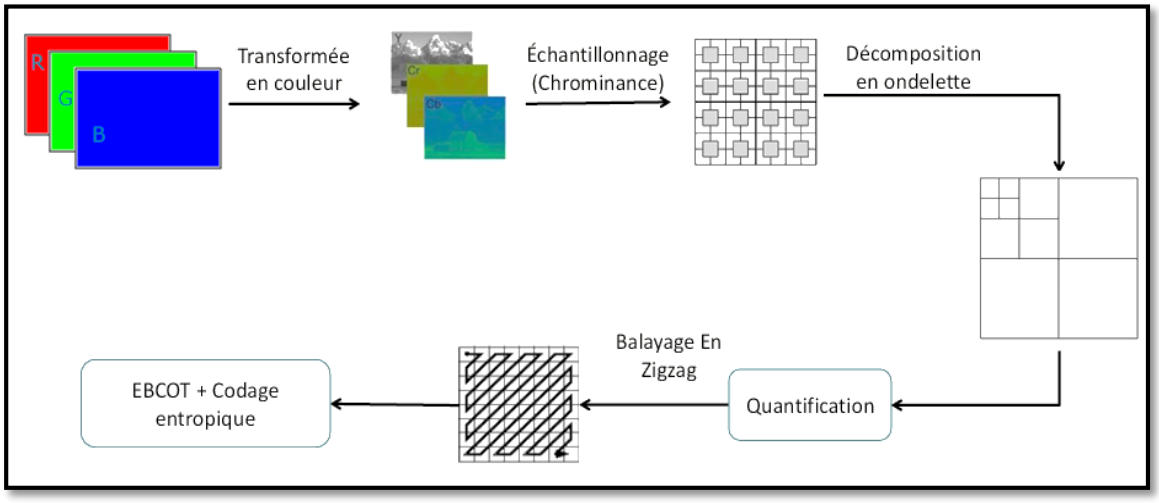

Figure 1.7. jpeg 2000 block diagram [11].

Here is a comparative table of the principles of **JPEG** and **JPEG 2000**:

Table 1.5. Comparison between JPEG and JPEG 2000.

Criterion	JPEG	JPEG 2000
Transformation method	Discrete Cosine Transform (DCT)	Discrete Wavelet Transform (DWT)
Compression type	Lossy (with limited lossless mode)	Lossy or lossless (more flexibility)
Image segmentation	8×8 pixel blocks	64×64 pixel code-blocks
Coefficient encoding	Huffman coding (less efficient)	Arithmetic coding with contextual modeling (more efficient)
Flexibility	Fixed (One quality level per file.)	Flexible (multiple quality levels in the same file).
Scalability	Limited (no support for multiple levels of detail).	Scalable (Supports different levels of detail and progressive display)

Comparative analysis of JPEG and JPEG 2000 in terms of quality and performance:

JPEG 2000 offers better image quality and more efficient compression than traditional JPEG. For example, Figure 1.8 shows an 800-pixel image may weigh 94 KB in JPEG, compared to only 32 KB in JPEG 2000, while maintaining equal or even better quality [13].

This size reduction significantly optimizes website performance by enabling faster loading times and lower bandwidth usage. However, the limited compatibility of JPEG 2000 with some browsers remains a challenge [13]**.**

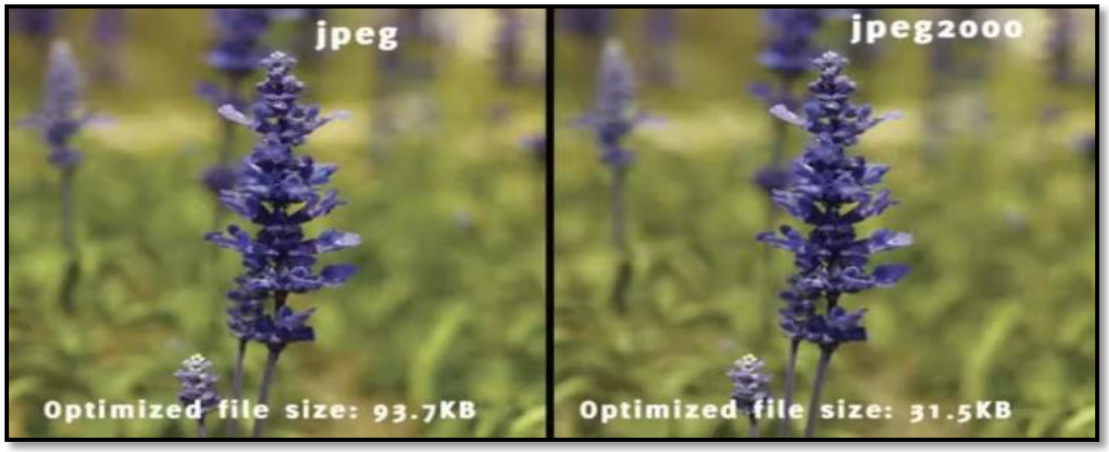

Figure 1.8. Better image with jpeg and jpeg 2000 [13].

1.3.5 H.264

H.264, or MPEG-4 Part 10 AVC, is a video compression standard developed in 2003 [14] by the Joint Video Team **(JVT)**. It offers up to **"50%"** better compression than MPEG-2 and **"30%"** better than H.263+ and MPEG-4 ASP, while maintaining optimal quality [15] [16]**.**

Adopted as a reference standard, H.264 is used in Blu-ray discs, HDTV online streaming, and mobile applications. It relies on the block-based hybrid coding, combining motion compensation and spatial transformation**,** with 16×16 pixel macro blocks for optimized compression and improved error robustness [17]**.**

H.264 uses a block-based hybrid coding scheme, combining motion compensation and transform coding**.** Images are divided into 16×16 pixel macro blocks, grouped into independent slices**.** Key features of H.264/AVC include:

- 4×4 block transformation to reduce complexity.

- Adaptive motion compensation (blocks from **16×16 to 4×4**).
- Quarter-pixel-accurate motion vectors, achieved through interpolation.
- Use of multiple reference frames for improved motion estimation.
- Directional spatial prediction optimizes intraframe coding [16].

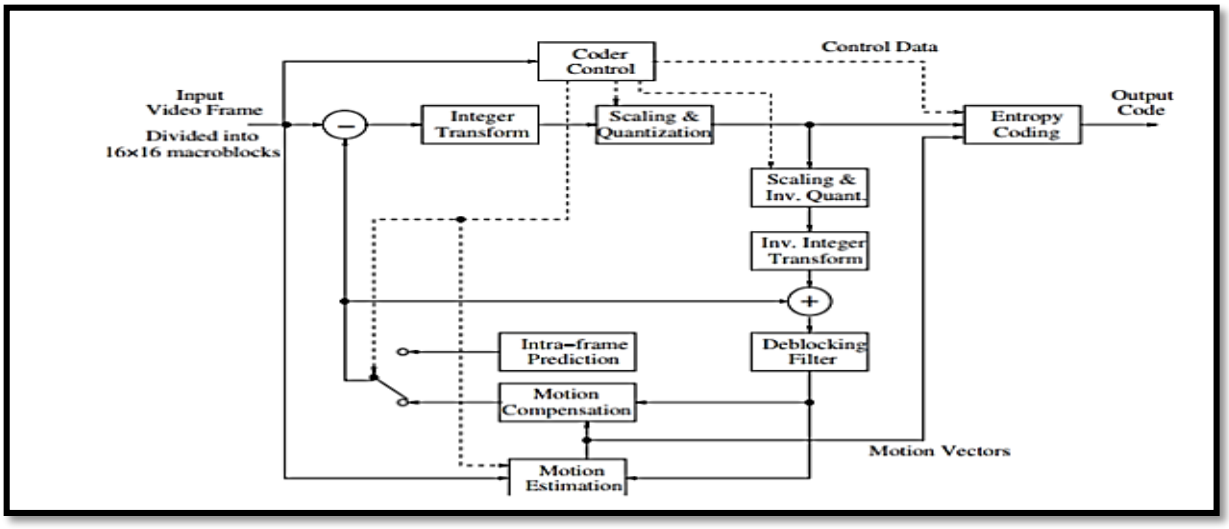

Figure 1.9. Fundamental structure of an H.264/AVC encoder [16].

H.264 Codec: Encoding and Decoding Process

- **Encoding (Compression):**
  - **Transform and Quantization**: Converts image blocks into coefficients using a transform derived from the DCT, then reduces precision (quantization) to optimize compression**.**
  - **Bitstream encoding**: Converts coefficients and prediction information into compact binary streams using variable-length coding and arithmetic coding [18]**.**
- **Decoding (Decompression):**
  - **Bitstream Decoding**: Extracts compressed data (coefficients, structure, and prediction information).
  - **Rescaling and Inverse Transform**s: Restores coefficients and reconstructs residual blocks.
  - **Image Reconstruction**: Adds reconstructed blocks to the prediction to reassemble the final image [18].

1.3.6 BPG (Intra H.265)

The BPG **(Better Portable Graphic)** format, developed in 2014 by Fabrice Bellard, uses the intra-part of the HEVC/H.265 codec and integrates an optimized 4-byte header for better compression [19]**.**

Despite its superior performance, its usage remains limited due to patents related to HEVC, particularly on the x265 encoder, making its implementation non-free [19].

BPG relies on an adaptive image partitioning into variable-size blocks.

- Larger blocks for uniform areas,
- Smaller blocks for detailed or textured regions [19].

These blocks are then subdivided into transform blocks ranging from 4×4 to 32×32 pixels for luma (luminance) and 8×8 to 32×32 pixels for chroma (chrominance**),** following a quadtree structure, optimized using the rate-distortion algorithm [19].

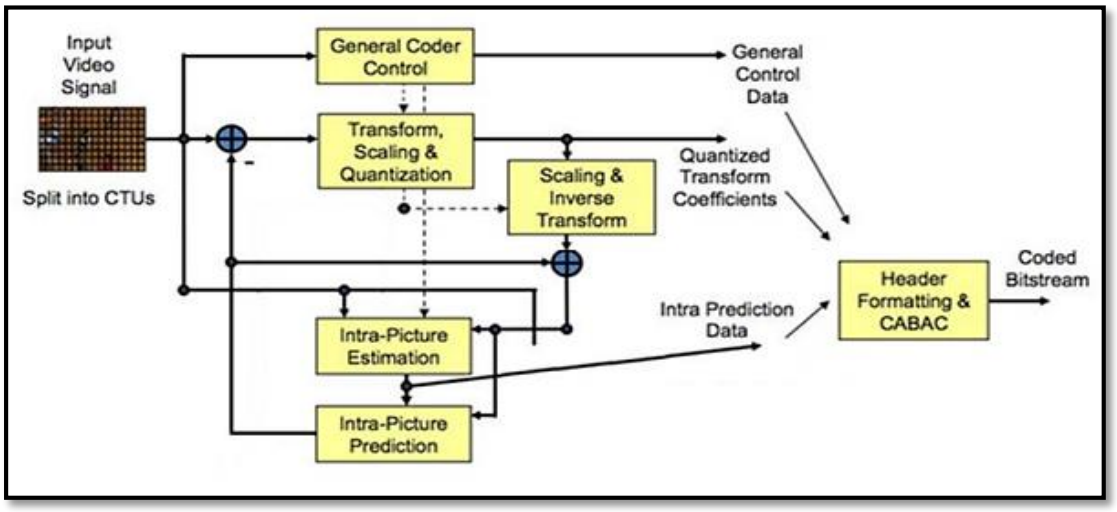

Figure 1.10. BPG encoding schema [19].

1.3.7 VTM (Intra VVC or H.266)

The Versatile Video Coding **(VVC/H.266)** is a new video compression standard developed by **JVET** (Joint Video Experts Team) and finalized in July 2020. It reduces video file sizes by 30 to 50% compared to HEVC (H.265) while maintaining the same quality [20]**.**

- **Key Improvements in VVC:**

VVC introduces several major improvements, including enhanced compression efficiency. this advancement significantly reduces storage costs and optimizes bandwidth usage

for streaming. Additionally, the format expands its compatibility by supporting resolutions up to 16K. Another key innovation is the new block partitioning structure, Quad Tree Multi Type **(QTMT),** which enables more flexible image segmentation with blocks of up to 128×128 pixels. By merging coding, prediction, and transformation units into a single element, this approach greatly simplifies the codec's architecture [20].

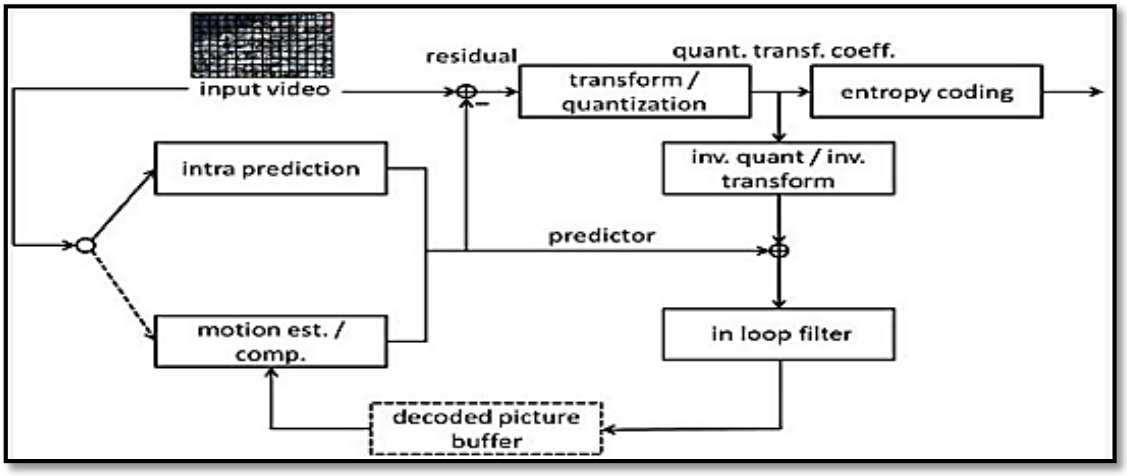

Figure 1.11.VVC codec structure [20].

- **Improved Intra Prediction**:
  - **Number of intra modes**: 67 modes, including 65 directional modes, one DC mode, and one planar mode [20].
  - **Block size**: Intra prediction is applied to blocks from 4×4 to 64×64 pixels [20].
  - **Redundancy reduction**: Uses linear model-based prediction to estimate chrominance samples from reconstructed luma samples [20].
  - **Quality improvement**: Introduces multiple reference lines to refine prediction and enhance texture reconstruction [20].

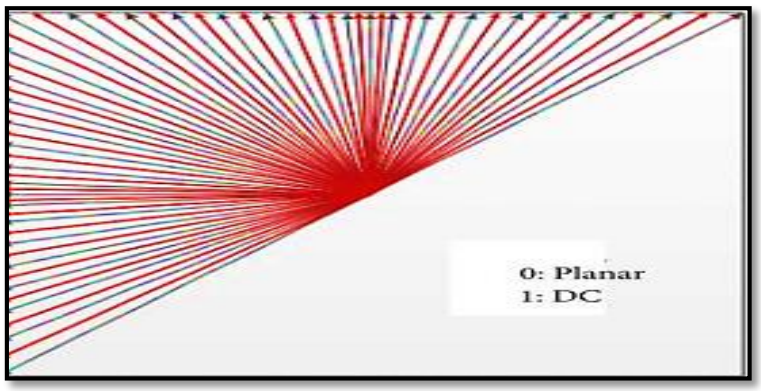

Figure1.12. Intra prediction of VVC [20].

Comparison between H.264, H.265, and H.266:

Here is a comparative table of the principles of H.264. H.265 and H.266

Table 1.6. Comparison between H.264, H.265, and H.266.

Block Diagram Element	H.264	H.265	H.266
Block Partitioning	Fixed size (16x16)	Flexible partitions (up to 64x64)	More advanced partitions (up to 128x128)
intraframe prediction	9 modes	35 modes	More than 65 modes
Transform	DCT (Discrete Cosine Transform) 4x4 and 8x8	DCT/DST up to 32x32	DCT/DST up to 64x64
Quantization	Linear	Improved with adaptive control	More refined and adaptive quantization
Entropy Coding	CABAC	Enhanced CABAC	Optimized CABAC with multi-symbols

1.4 Conclusion

This chapter highlights the importance of coding algorithms in image compression and intra prediction**,** optimizing storage and data transmission**.** Techniques such as Huffman coding, arithmetic coding, and Golomb Rice coding play a key role in reducing redundancy**,** while standards like JPEG and JPEG 2000 enhance compression efficiency and quality**.**

In the context of intra compression**,** particularly used in video**,** formats like H.264, H.265, and H.266 leverage advanced algorithms such as DCT, DWT, and adaptive partitioning, ensuring better coding efficiency. These methods enable more compact representation of images, thereby reducing storage space requirements and facilitating their transmission.

The impact of these advancements is significant in the digital field, contributing to more efficient resource management, **cost** reduction, and improved performance of communication and storage systems**.** Today, these technologies are essential in meeting the growing demands of image processing and sharing in a connected environment.

Chapter 2

Deep Learning and Deep Image Compression

Summary

This chapter presents the role of deep learning in advancing image compression, with focus on convolutional neural networks (CNNs) and variational autoencoders (VAEs) enhanced by hyperpriors. It introduces the basic principles of neural networks and autoencoders, then describes modern compression techniques that outperform traditional methods like JPEG and JPEG2000 by offering superior visual quality and greater compression efficiency, particularly through better modeling of spatial dependencies in latent representations.

Table of Contents:

2.1 Introduction

Deep learning (DL) is an advanced branch of machine learning, inspired by the human brain's ability to process information, recognize patterns, and make decisions [21]. It relies on deep neural networks to analyze complex data such as text, images, and audio by extracting high-level representations without the need for manual feature engineering. Deep learning plays a crucial role in fields like computer vision, speech recognition, and natural language processing, using structures such as convolutional neural networks **(CNNs),** recurrent neural networks **(RNNs)**, and generative adversarial networks **(GANs)**, and has become a cornerstone of modern artificial intelligence [22]**.**

This chapter explores deep learning, focusing on different architectures including artificial neural networks **(ANN),** convolutional neural networks (**CNN),** autoencoders, and methods for deep image compression.

2.2 Deep Learning

Deep learning is an advanced machine learning method that uses deep neural networks to automatically extract hierarchical features from data [23]**.** These networks process information through successive nonlinear transformations, where early layers detect simple patterns and deeper layers build complex representations. The success of deep learning is driven by increased computing power (notably through GPUs), reduced hardware costs, and major algorithmic advances [24]**.**

2.3 Artificial Neural Network

An artificial neural network (**ANN**) is inspired by the biological neural network and functions as an interconnected system of nodes, similar to neurons in the human brain. Each ANN is structured around two main components: node characteristics and learning rules. Node characteristics determine how information is processed, including the number of inputs and outputs, the weights assigned to each, and the activation function used. Learning rules guide how the weights are initially set and how they are adjusted during training. Together, these components enable the network to learn and adapt in response to data [25].

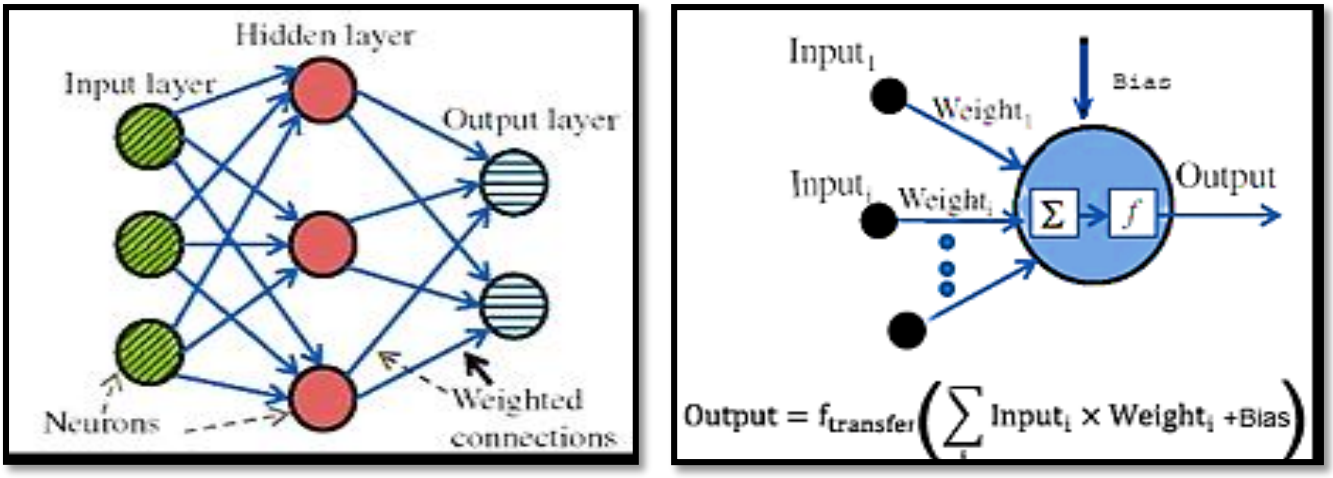

Figure 2.1. A basic feed forward neural network and neuron computation (a) artificial neuron network and (b) the perceptron.

2.3.1 Basic Neural Network Architecture (a):

A typical Artificial Neural Network (ANN) consists of three main components:

- **Input Layer:** Receives raw input data, where each neuron represents specific features. It may include preprocessing to prepare the data for the network [26].
- **Hidden Layers:** Located between the input and output, these layers apply mathematical transformations to the data using weights and activation functions. They are crucial for identifying complex patterns and learning meaningful representations [26].
- **Output Layer:** Delivers the final output or prediction of the network. The number of neurons depends on the task, such as the number of classes in classification problems [26].

2.3.2 Basic Principles of Neural Networks (b):

Neural networks rely on adjustable parameters that directly affect their training and performance. The following is an overview of key principles to enhance the effectiveness of neural networks:

- **Weight Initialization:** The method used to define the initial values of the model's weights before training (e.g., random initialization) [26].
- **Bias:** The bias in neural networks adjusts the activation function's output by shifting the input, helping the model fit the data better. It can either increase or decrease the net input depending on its sign [27].

- **Number of Hidden Layers:** This parameter is the depth of the network. More layers allow the model to capture more complex patterns but can also lead to overfitting [26].
- **Number of Neurons per Layer:** Affects the network's ability to learn complex relationships. Too many neurons can cause overfitting [26].
- **Activation Function:** Adds non-linearity to the model, which enables it to model sophisticated relationships in the data, for example, ReLU, tanh, or Leaky ReLU.

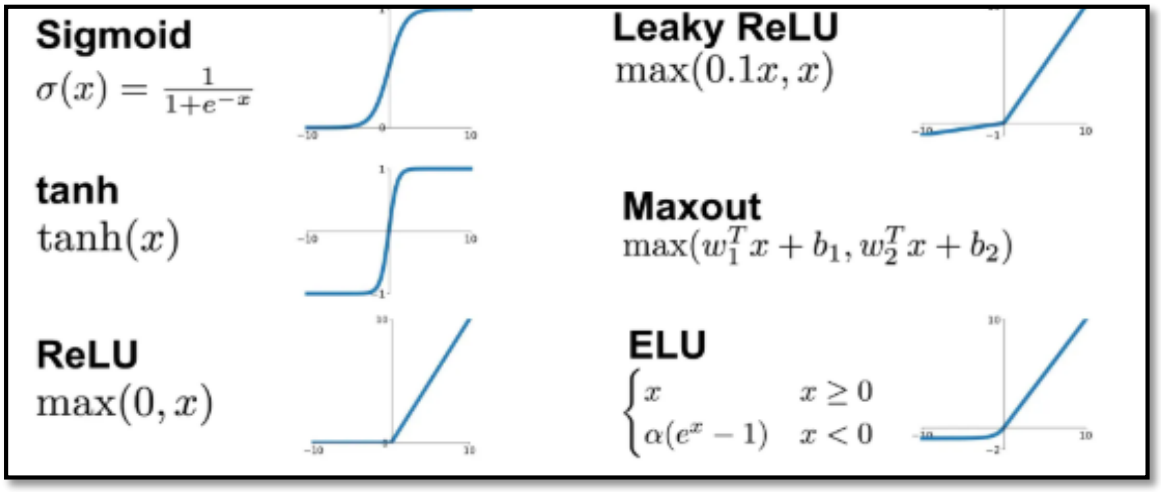

Figure 2.2. Types of activation functions in neural networks [26].

- **Loss function:** A loss function quantifies the error between the model's predictions and actual values, guiding the training process to minimize this error [26].
- **Learning Rate:** Determines the step size taken during weight updates. A rate that is too high may cause divergence, while one that is too low may slow down training (e.g., 0.1, 0.01, 0.001) [26].
- **Batch size**: Represents the number of training examples processed in one iteration of gradient descent. Typical batch sizes are 16, 32, 64, 128, 256, 512, or 1024 [26].
- **Epochs:** Refer to the number of complete passes through the entire training dataset during the training process. For example, you might train for 1, 10, 50, or 100 epochs [26].
- **Gradient Descent**: This is an iterative optimization algorithm discovered by "**Augustin-Louis Cauchy"** in the 18th century. It is used to train machine learning and deep learning models by minimizing **cost function** [28].
  - **How it works:**

It aims to reach a minimum of function by following the negative gradient. For each iteration it calculates the first derivative (slope) and updates the parameters in the opposite direction of the gradient, scaled by learning rates **(alpha)** [28].

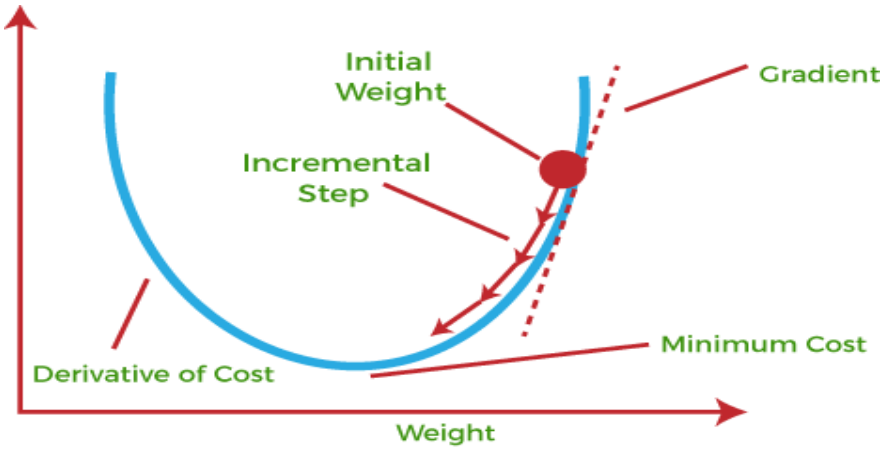

Figure 2.3. Illustration of gradient descent [28].

The **cost function** measures the average error between predictions and actual values. It guides the model's optimization by reducing this error toward the minimum, thus improving its performance [28].

2.4 The training process of a neural network

The training process of a neural network involves several key steps. First, input data is collected and preprocessed. Then, the network architecture is defined, including the choice of layers, number of neurons, activation functions, loss function, and optimization algorithms. During training, mini batches of data are passed through the network over successive iterations. In each iteration, forward propagation computes the outputs and the loss, while backpropagation adjusts the weights based on the calculated gradients to minimize the loss. This process continues until stopping criteria, such as a fixed number of epochs or early stopping, are met. Throughout training, the model's performance is evaluated on a validation set to fine-tune hyperparameters like learning rate and batch size. Finally, the trained model is assessed on a test set to evaluate its final performance [29].

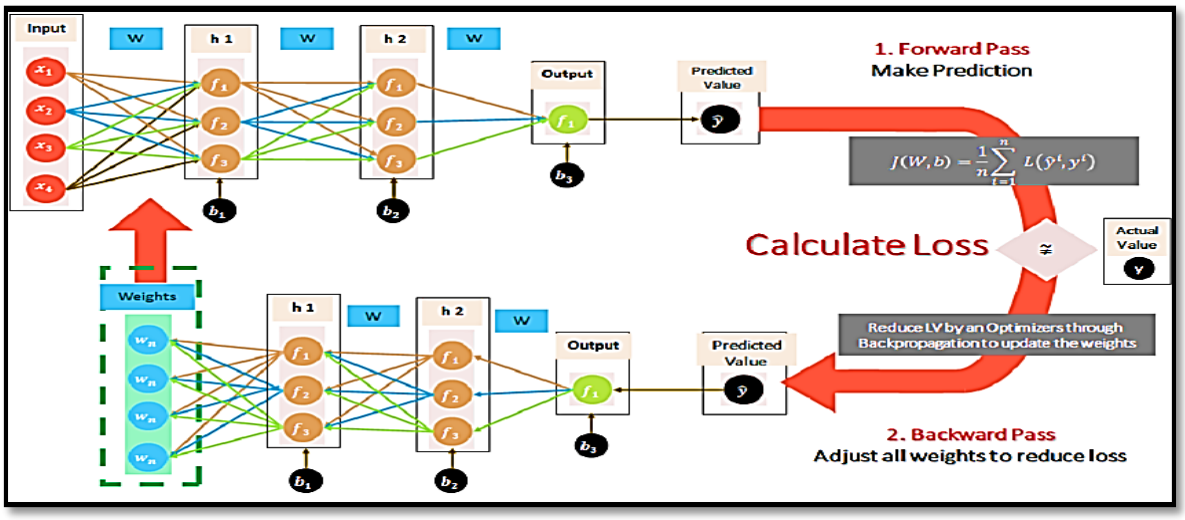

Figure 2.4. Training process of neural network [30].

2.5 Types of neural networks

1) **RNNs (Recurrent Neural Networks):** Suited for sequential data such as text or time series. They retain memory of previous inputs to model temporal dependencies [29].
2) **Generative models:** learn from unlabeled data to generate new samples by modeling probability distributions. They use loss functions to optimize their predictions through probabilistic processes. Key types include deep models (multilayer networks), autoregressive models (sequential prediction), transformers (NLP tasks), diffusion models (adding and removing noise), GANs (generator versus discriminators), and VAEs (compression and data generation) [31].
3) **CNNs (convolutional neural networks):** These are deep learning models designed for visual data analysis. Inspired by the human visual system, they automatically learn hierarchical feature representations through convolution and pooling operations, effectively capturing spatial patterns while maintaining translation invariance and reducing parameter complexity. Their efficiency and strong performance make them particularly suitable for tasks like object detection, face recognition, autonomous driving, and medical image analysis [32].

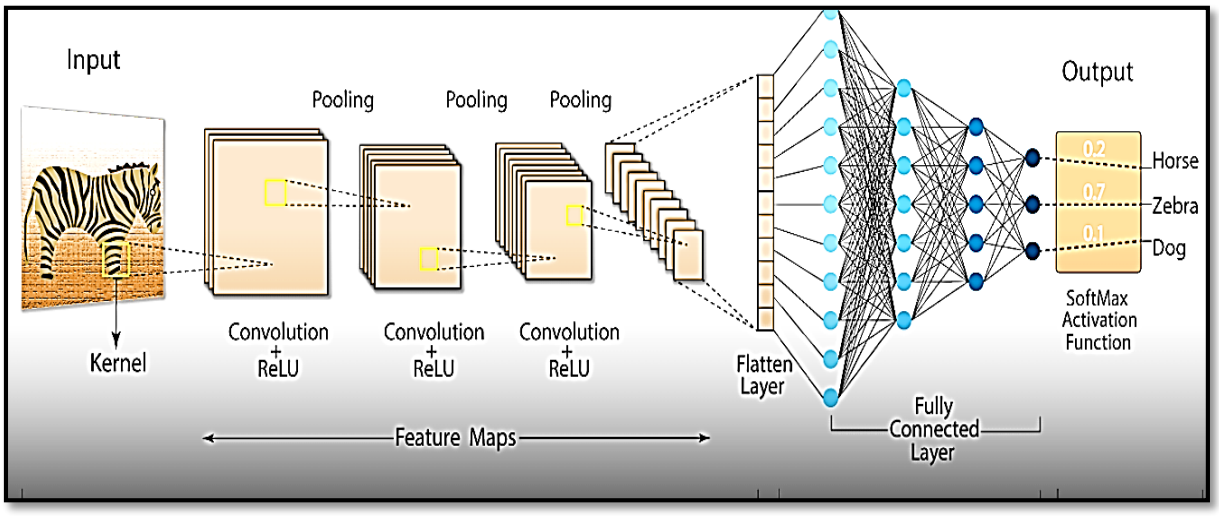

Figure 2.5. Convolutional neural networks (CNNs) [32].

- **Convolutional Layers:**

The convolution layer is the first key element of a convolutional neural network (CNN). It applies fixed-size filters (such as 3×3 or 5×5) to the image using a sliding window function to detect specific patterns such as edges, curves, or shapes. These filters move across the image, creating a new grid that highlights the detected patterns [33].

- **Pooling Layers:**

Pooling layers in CNNs compress feature maps via downsampling, boosting computational efficiency while adding robustness to small input distortions. Unlike convolutional layers, they contain no trainable parameters, only configurable hyperparameters (filter size, stride, padding). By extracting key features (**max pooling** for dominant activations, **average pooling** for smoothed values), they reduce spatial redundancy, curb overfitting, and preserve hierarchical patterns. This balance of simplification and feature retention makes pooling indispensable for efficient, high-performing CNNs [32].

- **Padding and Stride:**

Padding and stride influence how the convolution operation is performed, altering the dimensions (height and width) of the input-output vectors. **Padding** involves adding zeros around the input to ensure the output dimensions match the input, improving image analysis and preserving information at the image borders. **Stride** controls how the filter moves across the input; a larger stride reduces the output size, while a smaller stride keeps the output size larger [33].

- **Fully Connected Layers:**

A fully connected or dense layer connects every neuron in one layer to every neuron in the next. The output from the final convolutional pooling layer is flattened into a 1D vector and fed into fully connected layers, where each input links to an output via learnable weights. After feature extraction and downsampling, fully connected layers map these features to produce final outputs [33].

Finally, the softmax layer is applied to generate probability values for each possible output label, and the predicted label is the one with the highest probability score [33].

2.6 Autoencoders

Autoencoders are neural network architectures used for unsupervised learning. They are primarily used for data compression and extracting useful representations. There are variants like "**g**enerative autoencoders", which learn latent models to generate new data, or "variational autoencoders", which learn latent distributions of data to generate new data. Autoencoders can be applied to both generic and image data, often using CNNs [34].

The architecture of an autoencoder consists of two parts: the **encoder** and the **decoder**. During training, both parts are used, but for dimension reduction on new data, only the encoder is needed. The encoder transforms the input data into a lower-dimensional version, ultimately reaching the **bottleneck layer**, which contains the reduced representation of the data. Then, the decoder tries to reconstruct the original data from this representation. The goal is for the reconstructed data to be as close as possible to the input data [34].

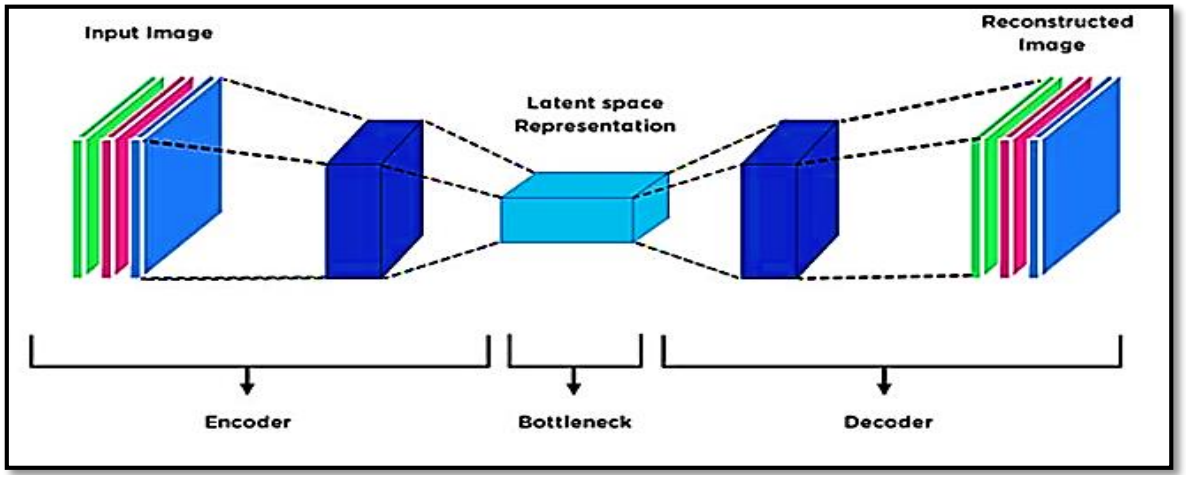

Figure 2.6. Auto-encoder architecture [35].

2.7 Deep image compression

2.7.1 Nonlinear transform codes for perceptual quality:

This section introduces an advanced framework for optimizing nonlinear transform coding through a fully differentiable and end-to-end learning approach. The primary objective is to improve the balance between compression efficiency (bitrate) and perceptual quality, moving beyond traditional fidelity metrics such as PSNR, which do not always align well with human visual perception. Instead, the framework integrates perceptual distortion measures that better capture how humans perceive image quality [36].

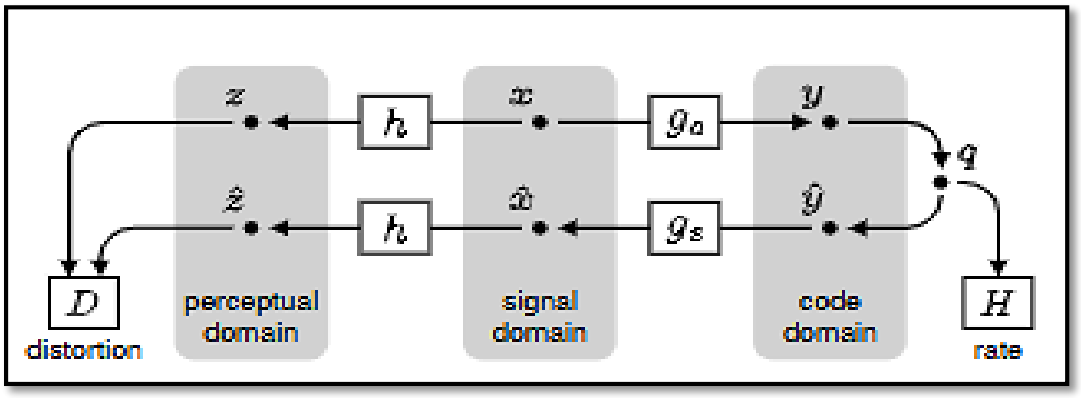

Figure 2.7. End-to-End image compression framework with perceptual distortion and entropy rate optimization [36].

The process begins with an **analysis transform (g_a),** which is a parametric and differentiable function applied to the input image to extract meaningful features for compression. These features are then quantized using scalar quantization. However, since the quantization operation (e.g., rounding) is inherently non-differentiable, it is approximated during training by adding **uniform noise** (U~ [−0.5, 0.5]). This trick enables the entire pipeline to be optimized using gradient-based methods [36].

After quantization, the compressed features are passed through **synthesis transform**(g_s), which reconstructs the image. The reconstructed output is then compared to the original using a perceptually aware metric. The overall performance is evaluated using two components: (1) the entropy of the quantized representation, which relates to the number of bits required to encode the image, and (2) a distortion metric that reflects the perceptual difference between the original and reconstructed images [36].

The overall cost function to be minimized is defined as follows:

$$L[g_a, g_s] = H[P_q] + \lambda E\|z - \hat{z}\| \quad (2.1)$$

Here, $H[P_q]$ denotes the entropy, and $\lambda E\|z - \hat{z}\|$ represents the perceptual difference between the original and reconstructed data, with λ compression rates and visual fidelity [36].

The analysis transforms uses Generalized Divisive Normalization (GDN) [37], inspired by the human visual system, which better captures local contrast and texture information. An approximate inverse of GDN is used for the synthesis step. For perceptual evaluation, the Normalized Laplacian Pyramid (NLP) [38] is used as a metric, providing a more accurate assessment of visually perceived distortions [36].

Experiments show that combining GDN with the NLP perceptual metric achieves better visual quality at similar bitrates, outperforming classical methods like the DCT, even with simpler uniform quantization. In particular, under **low bitrate conditions**, this approach preserves perceptually important details and reduces visually disturbing artifacts much better than traditional methods [36].

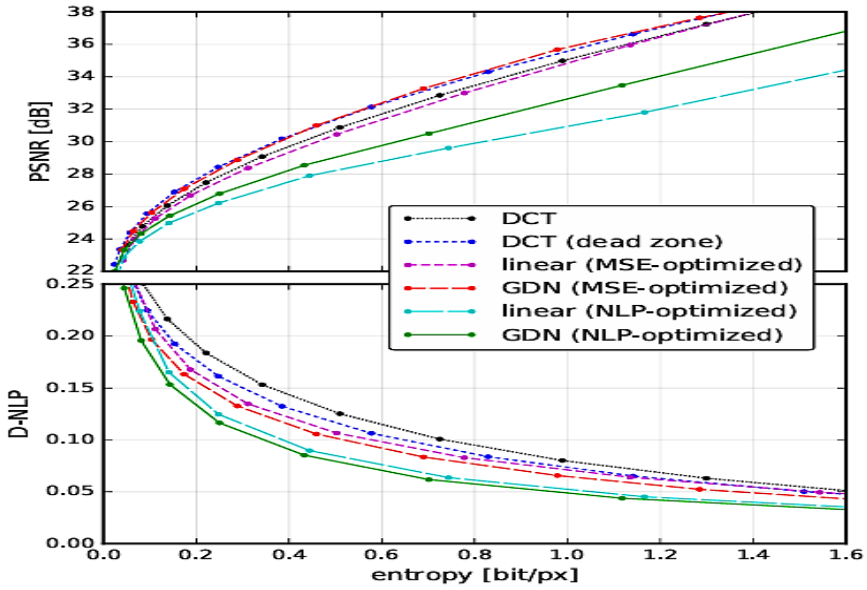

Figure 2.8. Rate-Distortion performance of transform coding methods on the Kodak image set [36].

Figure 2.8 compares different image compression methods (DCT, DCT with a dead zone, and linear GDN approaches optimized for MSE or NLP) by measuring their PSNR (quality) against entropy (compression rate). It shows that modern methods, such as GDN optimized for NLP, offer

a better quality-rate trade-off, surpassing classical techniques by preserving high perceptual quality even at low bitrates [36].

2.7.2 End-to-End Optimized Image Compression:

In [39], the authors introduce an innovative image compression method based on deep learning, optimized in an end-to-end fashion. Unlike traditional methods like JPEG or JPEG 2000, which rely on separate stages (transform, quantization, entropy coding), the proposed method integrates all these steps into a single neural network trained globally to minimize the trade-off between compression rate (bitrate) and image reconstruction quality. Using a deep autoencoder framework interpreted as a variational autoencoder (VAE) [39].

The model is built around two main components: an analysis transform that extracts latent representations from the input image and a synthesis transform that reconstructs the image from these latent codes. Both components are implemented using deep, nonlinear convolutional networks. A key feature of the architecture is the use of GDN, inspired by mechanisms in the human visual system, which helps decorrelate the filter responses and Gaussianize their distributions to improve coding efficiency [39].

To enable training through backpropagation, the quantization step, which is normally non-differentiable, is approximated during training by adding uniform noise. In its original form, quantization is defined by rounding to the nearest integer:

$$\hat{y}_i = q_i = round(y_i) \tag{2.2}$$

Which is not differentiable and therefore problematic for gradient-based learning. To overcome this, the authors replace the hard quantization with continuous relaxation during training:

$$\hat{y}_i = y_i + \Delta y \tag{2.3}$$

Where: $\Delta y = U \sim [-0.5, 0.5]$, which simulates quantization noise in a differentiable way [39].

The core optimization objective is based on the classical rate distortion trade-off, defined as

$$L = R + \lambda D \tag{2.4}$$

Where R is the estimated rate (in bits per pixel), D is the distortion (often measured by mean squared error), and λ (lambda) is a scalar controlling the balance between file size and visual quality [39].

The proposed method outperforms JPEG and competes with JPEG 2000, especially at low bitrates. Despite using MSE as the training criterion, it produces visually more natural images, as confirmed by MS-SSIM. This highlights the effectiveness of deep learning in learning compact, optimized representations for image compression [39].

These results include additional example images compressed at relatively low bit rates, providing visual insight into the qualitative nature of compression artifacts. Metrics such as MS-SSIM and PSNR are used to evaluate the fidelity of reconstructed images [39].

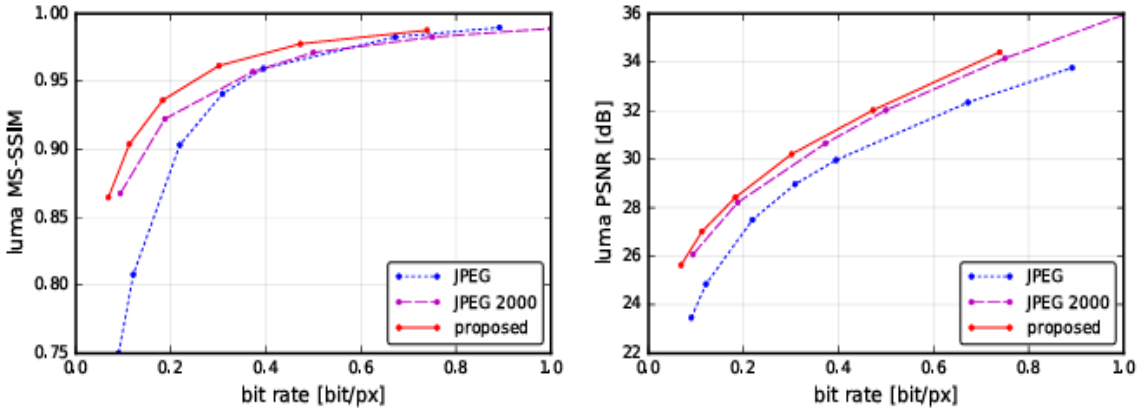

Figure 2.9. Rate–distortion curves for the luma component of images [39].

Tests show that their new compression method gives better image quality for the same file size than old codecs. In the graphs, the JPEG and JPEG 2000 curves stay far below ours across all bitrates, meaning they add more distortion at every setting. Sample images at very low bitrates back this up: JPEG and JPEG 2000 create blocky and ringing artifacts, while our method keeps edges sharp and textures clear, achieving higher visual quality without using more bits [39].

2.7.3 Variational image compression with a scale hyperprior:

Recent approaches to image compression rely on deep generative models, particularly VAEs. These methods transform an image x into latent representation y, which is then quantized into $\hat{y}$ to enable efficient compression through entropy coding. The main objective is to minimize cost function that combines the compression rate (number of bits used) and distortion [40].

- **Factorized prior model:**

The factorized prior is simplifying assumptions used in compression models based on latent representations (such as VAEs). It assumes that the latent variables $\hat{y} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n)$ are statistically independent from each other [40].

Thus, the joint distribution of the latent vector factorizes into the product of the marginal distributions of each latent variable:

$$p(\hat{y}) = \prod_i p_{\hat{y}} (\hat{y}_i) \quad (2.6)$$

To model the effect of quantization, each $\hat{y}$ is convolved with a centered uniform distribution representing quantization noise.

This simplification means the model treats each latent variable separately, ignoring any correlation or spatial dependencies between them. While this makes entropy coding and optimization easier, it limits the model's ability to capture important structural relationships (such as edges or textures) within the latent space, which can reduce compression performance [40].

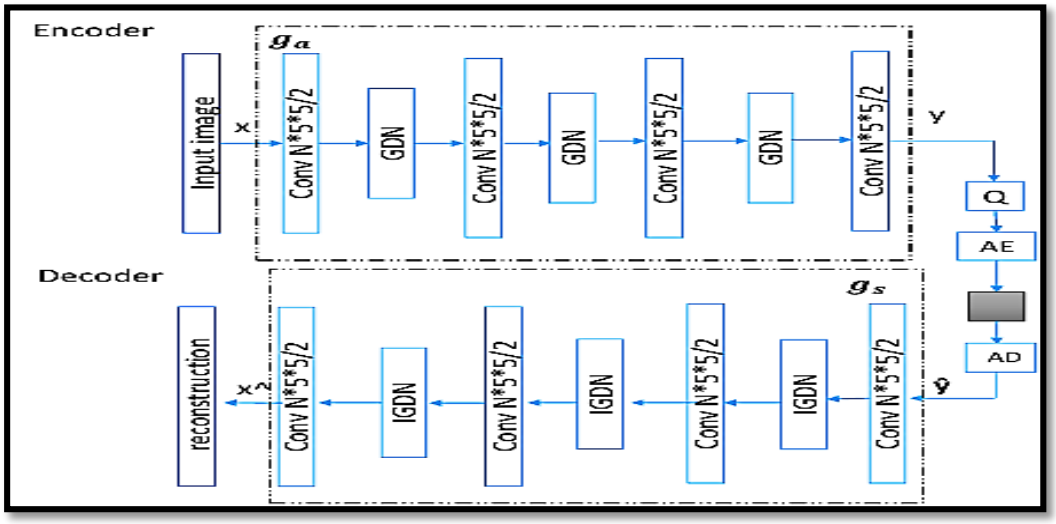

Figure 2.10. Image compression using VAE with factorized prior.

- **Encoder(g_a):**

The encoder transforms the original image into a into compressed version y by applying local convolutions (5×5 filters) and nonlinear normalization (GDN), which simulates human visual adaptation. This step extracts essential features while reducing data size [40].

- **Quantization and coding:**

The latent representation y is quantized into $\hat{y}$ to reduce numerical precision. Then, $\hat{y}$ is compressed into compact binary file using non-adaptive arithmetic coders. The model assumes that

the elements of $\hat{y}$ are independent and follow a univariate distribution that is identical across all images. This means the probabilities used for encoding are fixed and do not depend on the specific image [40].

- **Decoder (g_s):**

The decoder receives the compressed file, decompresses it, and then applies inverse operations (convolutions and IGDN) to reconstruct an image that closely approximates the original. This process optimizes visual quality while maintaining an efficient compression rate [40].

- **Hyperprior Model:**

To overcome the limitations of the factorized model, the authors introduce hyperprior**,** an additional latent vector z used to model the local scale (variance) of the latent variables y. Each $y_i$is modeled as zero-mean Gaussian with variance conditioned on z:

$$p\left({}^{y_i}/_{z}\right) = N(0, \sigma_i^2) \tag{2.7}$$

where $\sigma = h_s\ (z)$

This allows the model to represent local spatial variation more effectively. The vector z is derived from y via a function $h_a\ (y)$ then quantized and transmitted a**s** side information. At the decoder side, $\hat{z}$ is used to reconstruct σ, improving the entropy model used to decode $\hat{y}$ [40].

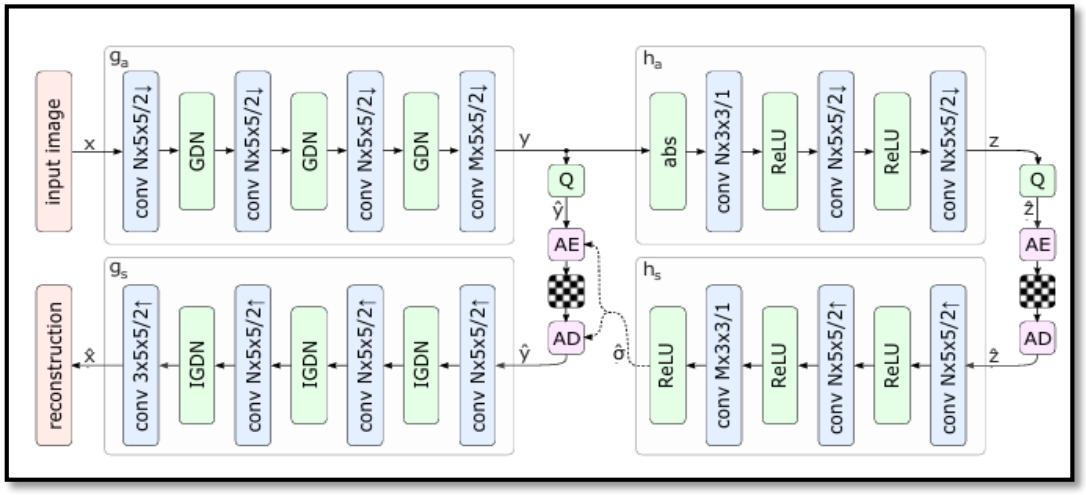

Figure 2.11. Network architecture of the hyperprior model [40].

- **Latent representation:**

The image is first transformed into latent representation y by an encoder. This representation contains the essential information needed for reconstruction, but it still needs to be efficiently compressed [40].

- **Hyperprior generation (z):**

To improve the modeling of y, the secondary network (h_a) generates an auxiliary variable z from y. This variable captures statistical information (such as local scale or variance) useful for predicting the distribution of y [40].

- **Probability estimation:**

After the latent representation y is extracted from the image, it is used to generate an auxiliary latent z through a second network (h_a). This z is then quantized into $\hat{z}$ and passed to a synthesis network (h_s), which predicts for each element of $\hat{y}$ two key statistical parameters: the mean (μ), representing the expected or central value, and the standard deviation (σ), indicating how much variation or uncertainty exists around that mean. These values define tailored probability distribution for each point in $\hat{y}$, which allows the arithmetic coder to compress the data more precisely. By adapting the encoding to the image content through these predicted distributions [40].

- **Conditional Arithmetic Coding**

The probabilities used in the arithmetic coder are no longer fixed or assumed to be the same for all elements. They are dynamically computed from $\hat{z}$, making the encoding process more accurate and efficient [40].

- **Reconstruction:**

The decoder uses $\hat{z}$ and $\hat{y}$ to reconstruct the final image, the reconstructed image has higher visual quality at the same bit rate compared to classical methods [40].

2.8 Conclusion

This chapter has provided an in-depth analysis of how deep learning has revolutionized image compression. We have seen that traditional techniques like JPEG and JPEG2000, although efficient in their time, are now being surpassed by modern approaches based on deep learning, particularly convolutional neural networks (CNNs) and variational autoencoders (VAEs) with hyperpriors. These models benefit from powerful tools such as convolution operations, pooling,

nonlinear activation functions like ReLU, and advanced optimization techniques such as the gradient descent.

By enabling end-to-end learning, deep models automatically discover the most efficient way to represent and compress images, optimizing both bitrate and visual quality. The integration of hyperpriors into VAEs, in particular, has greatly enhanced the modeling of spatial dependencies, resulting in more efficient compression and higher perceptual fidelity.

Chapter 3

Results and Discussion

Summary

This chapter presents thorough evaluation comparing traditional image compression algorithms with state of the art deep learning based models. The performance metrics like a peak signal noise ratio (PSNR) and visual quality assessments are utilized on benchmark datasets. The analysis demonstrates that methods based on deep learning (variational autoencoder) offer markedly improved compression efficiency and reconstruction quality. These findings highlight the increasing viability of deep learning approaches as powerful alternatives to conventional compression techniques.

Table of Contents:

3.1 Introduction

In this chapter, we present the results of our simulation study on image compression techniques, focusing on both traditional codecs and modern deep learning (DL) methods. The main objective is to assess the effectiveness of each approach in reducing image size while preserving visual quality.

Our evaluation covers established conventional codecs such as JPEG [41], JPEG 2000 [42], and BPG [43], alongside advanced DL models based on autoencoders, specifically the factorized VAE [40] and hyperprior VAE [40] architectures. These models were trained using Flicker 2W datasets [44] and tested on widely recognized benchmark datasets, including Kodak [45] and Tecnick RGB [46], to ensure reliability and fairness in the comparative analysis.

To objectively measure the rate distortion (RD) performance, we utilized widely accepted metrics: Peak Signal to Noise Ratio (PSNR), Structural Similarity Index (SSIM), and bitrate (bit per pixel). These metrics enable the creation of rate distortion curves, facilitating comprehensive analysis of the trade-offs between compression efficiency and image fidelity across the different techniques.

3.2 Software Material

We trained our model using the PyTorch library with GPU (RTX 3060 Ti) acceleration to improve computational efficiency. To evaluate image compression performance, we applied our custom techniques, namely the **factorized** and **hyperprior models**, and computed the PSNR, the bitrate, and the SSIM using Google Colab. In parallel, we used MATLAB 2019a to implement classical compression methods such as **JPEG, JPEG 2000,** and **BPG**. We then used Google Colab to plot and compare the performance curves of all compression techniques.

3.3 Metrics

In this work, we evaluated the performance of image compression techniques using various metrics. The first metric used is **PSNR**, which measures the difference between the original image and the compressed image. A higher PSNR value indicates better reconstruction quality [47]. The general form of PSNR is given by

$$PSNR(f,g) = 10log_{10}\left(\frac{255^2}{MSE(f,g)}\right) \tag{3.1}$$

Where **MSE** (Mean Squared Error) is defined as

$$MSE(f,g) = \frac{1}{M*N}\sum_{i=1}^{M}\sum_{j=1}^{N}(f-g)^2 \quad (3.2)$$

Here, f and g represent the pixel values of the original and reconstructed images, respectively, and M×N is the image resolution.

Additionally, bitrate represents the amount of data required to store an image after compression. It is expressed in bits per pixel (Bpp) and helps evaluate the efficiency of a compression method. An effective algorithm aims to reduce the bitrate while preserving acceptable visual quality [45].

Furthermore, **SSIM** is a metric that evaluates the similarity between two images by considering luminance, contrast, and structure. It provides a value between 0 (no similarity) and 1 (identical images). Unlike PSNR, SSIM is more consistent with human visual perception [47].
The general form of SSIM is given by:

$$SSIM(f,g) = l(f,g)c(f,g)s(f,g) \quad (3.3)$$

Where $l(f,g)$ is the luminance comparison**,** $c(f,g)$ is the contrast comparison, and $s(f,g)$ is the comparison?
The combined formula is expressed as:

$$SSIM(f,g) = \frac{(2\mu_f\mu_g+C_1)(2\sigma_{fg}+C_2)}{(\mu_f^2+\mu_g^2+C_1)(\sigma_f^2+\sigma_g^2+C_2)} \quad (3.4)$$

Here, μ_f and $\mu_g$are the means of images f and g. μ_f^2 and μ_g^2 are their variances, and σ_{fg} their covariance. Constants C_1 and C_2 help stabilize the formula and avoid division by zero.

- **Datasets:**

To evaluate the performance of various image compression techniques, we used two datasets that are well established in the scientific literature.

The first, **"Kodak,"** consists of 24 uncompressed color images with a resolution of 768×512 pixels. These images cover a variety of natural scenes, textures, and colors and are commonly used to evaluate the performance of image compression algorithms [45].

The second dataset, **"Tecnick RGB Dataset,"** includes 100 high-resolution images (1200×1200 pixels) and is known for its high level of visual detail, making it useful for evaluating compression performance under demanding quality conditions [46].

- **Training:**

On the other hand, we trained our model using the **"Flicker 2W images"** dataset, which contains around 20,000 images sourced from the Flickr platform [44]. These images offer a broad range of scenes and objects, enabling the model to generalize well across diverse visual content.

To standardize input size and reduce computational costs, all images were resized to 256×256 pixels with three RGB channels.

Meanwhile, we conducted the training over 200 epochs, with a batch size of 8. To optimize the model, we selected an optimizer with an initial learning rate of 0.0001. The momentum parameters were set to $b_1 = 0.9$ and $b_2 = 0.999$, helping stabilize weight updates. Furthermore. To speed up data loading, we saved visual samples every 200 iterations to monitor the qualitative progress of the model. For the quantization, we set the lambda value to 2048, which plays a key role in controlling the compression level and the quality of the reconstructed images.

We performed training on the model using the "**Flickr 2W images**" dataset. The training script was developed in PyTorch to monitor and reduce the loss function across epochs. This approach aimed to progressively improve the model's accuracy by minimizing the loss during each training iteration.

To evaluate the model's performance during training, we plotted the training loss against the Number of epochs as shown in Figure 3.1:

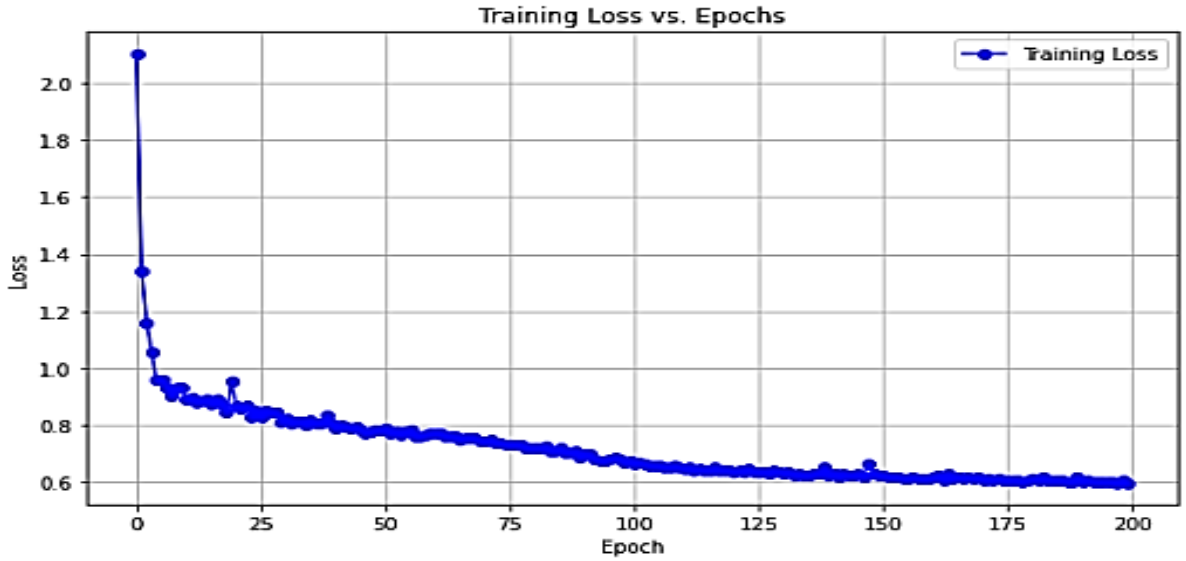

Figure 3.1. Evolution of training losses during model training.

Figure 3.1 illustrates the gradual change in the loss function value during the 200 epochs of training. A continuous decrease in loss can be observed as training progresses, indicating a gradual improvement in the model's ability to minimize errors and reconstruct images more accurately over time. This decline reflects the effectiveness of the training process and the model's stability in adapting to the training data.

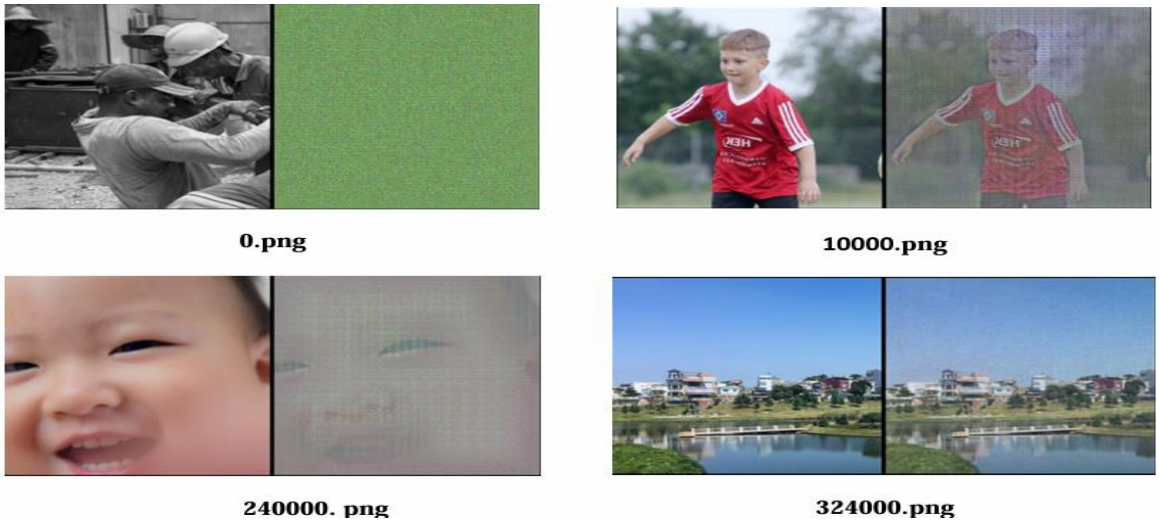

Figure 3.2. Image quality evolution across training epochs.

The figure 3.2 results illustrate the evolution of the training process, showing how the loss decreases across epochs and how image quality improves progressively throughout the optimization. Figure 3.2 shows clear improvement in visual quality as the number of training epochs increases. Early outputs are blurry or noisy, while later ones become progressively sharper and more faithful to the original.

3.4 Rate-distortion (RD) performance

The concept of RD performance is a key measure in image compression, representing the trade-off between compressed file size (bitrate) and visual quality (distortion). The objective is to compress the image while preserving image fidelity. To quantify visual quality, standard metrics such as PSNR (dB) and SSIM are used, enabling consistent comparisons between original and compressed images.

In this part, we evaluate the RD performance of different image compression techniques, chosen to represent both traditional methods (JPEG [41], JPEG2000 [42], BPG [43]) and recent DL-based models (factorized and hyperprior variational autoencoder (VAE)). Two image datasets are employed to evaluate these methods: the Kodak dataset containing 24 natural images and a custom Tecnick RGB dataset of 100 images.

For each compression method, we first calculated the **bitrate (Bpp)**, the **PSNR (dB),** and **the SSIM** values for each of the 24 images of the Kodak dataset. Then, we computed the average of these values to obtain a global measure of RD performance for each method. The resulting average values are presented in the following tables, providing concise comparison of the RD behavior of the tested approaches.

Table 3.1. The RD performance results of JPEG codec using Kodak dataset.

Quality	4	6	15	30	45	70	80	85
Bitrate (Bpp)	0.2006	0.2441	0.4257	0.6601	0.8497	1.2398	1.5718	1.8588
PSNR (dB)	22.77	24.67	28.14	30.49	31.83	33.93	35.40	36.50
SSIM	0.6295	0.7458	0.8797	0.9258	0.9435	0.9624	0.9714	0.9768

Table 3.1 shows that as the quality setting increases, the bitrate also increases, resulting in better image quality as measured by PSNR and SSIM. However, the improvement in quality becomes less noticeable at higher quality levels, indicating diminishing returns in compression performance.

Table 3.2 presents the RD performance results for the JPEG2000 codec, allowing us to compare its performance with JPEG codecs.

Table 3.2. The RD performance results of JPEG 2000 codecs using Kodak dataset.

Quality	26	27	30	33	35	37	39	40
Bitrate (Bpp)	0.1543	0.2041	0.4124	0.7308	1.0031	1.3341	1.7397	1.9766
PSNR (dB)	25.60	26.62	29.63	32.68	34.70	36.69	38.65	39.56
SSIM	0.8213	0.8489	0.9092	0.9472	0.9630	0.9740	0.9815	0.9843

Table 3.2 indicates that as the quality parameter increases, both the bitrate and the image quality metrics (PSNR and SSIM) improve significantly. Notably, the SSIM values are generally higher compared to the previous method, suggesting better preservation of structural details in the compressed images. The bitrate increases steadily, reflecting the trade-off between compression level and image fidelity.

Table 3.3 shows the RD performance of the BPG codec, providing further insight into advanced compression techniques.

Table 3.3. The RD performance results of BPG codecs using Kodak dataset.

Quality	46	36	32	31	29	26	23	21
Bitrate (Bpp)	0.0809	0.3825	0.6377	0.7175	0.8930	1.2059	1.5807	1.8914
PSNR (dB)	26.93	31.67	34.07	34.73	35.99	37.84	39.56	40.60
SSIM	0.8322	0.9324	0.9575	0.9626	0.9706	0.9793	0.9850	0.9880

Table 3.3 shows a significant increase in PSNR and SSIM as the quality parameter rises from low to medium levels. Notably, high quality is attained at moderate bitrates, demonstrating efficient compression that balances quality and bitrate better than previous methods.

Table 3.4 shows us the results of the deep image compression method [40]. This method utilizes the VAE to code and decode the image. In addition, this method utilizes the factorized techniques to calculate the probability of latent values for arithmetic coding.

Table 3.4. The RD performance results of factorized VAE codecs using Kodak dataset.

Quality	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Bitrate (Bpp)	0.1223	0.1882	0.2927	0.43971	0.6539	1.0495	1.3495	1.8040
PSNR (dB)	26.9087	28.2179	29.6819	31.2779	33.0409	35.308	37.3969	39.6213
SSIM	0.9099	0.9360	0.9551	0.9703	0.9806	0.9878	0.9921	0.9951

This model shows a gradual and consistent increase in both **PSNR** and **SSIM** with higher quality levels. Notably, it reaches high visual quality at mid-range bitrates, reflecting its strong capacity to compress efficiently while preserving fine image details and structural similarity.

Table 3.5 shows us the RD performance of the hyperprior VAE model, highlighting the effectiveness of DL-based compression compared to traditional methods.

Table 3.5. The RD performance results of hyperprior VAE codec using Kodak dataset.

Quality	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Bitrate (Bpp)	0.1311	0.2080	0.2194	0.7476	0.6671	0.9372	1.2566	1.3565
PSNR (dB)	27.5815	29.1966	30.9721	32.8381	34.5262	36.7435	38.5386	40.5568
SSIM	0.9173	0.9420	0.9610	0.9748	0.9835	0.9892	0.9928	0.9954

The hyperprior VAE model [40] demonstrates impressive performance, achieving high **PSNR** and **SSIM** values even at low bitrates. Compared to other methods, it maintains excellent quality with low bitrate, indicating its effectiveness in preserving both pixel fidelity and structural details during compression.

Now, we draw the curves of the RD performance of various image compression methods on the Kodak and Tecnick_RGB_OR_1200x1200 datasets. The curves include **PSNR vs. BPP**and **SSIM vs. BPP**, allowing us to compare the compression performance of different codecs at varying compression rates.

The curves of RD performance are built using the numerical data from the previous tables. They help to better visualize the RD performance of each compression method by showing how image quality (PSNR and SSIM) changes with the bitrate. This visual representation makes it easier to compare the efficiency of the methods.

3.4.1 Kodak dataset:

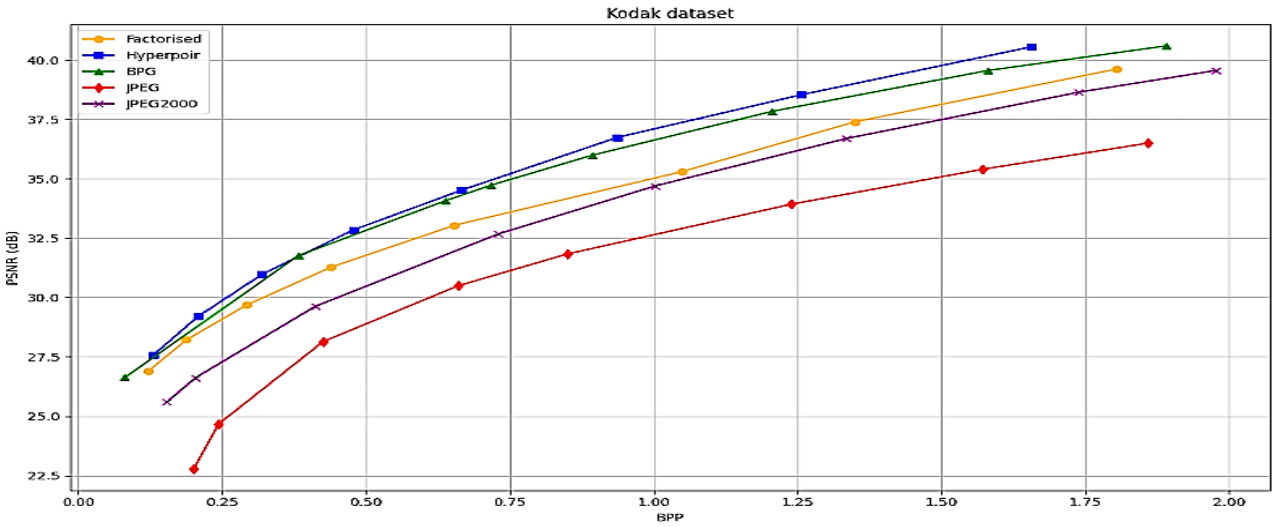

Figure 3.3. The RD performance of different image compression methods on Kodak datasets using PSNR metric.

Figure 3.3 shows us the performance of different compression methods on the Kodak dataset. DL models, especially hyperprior and factorized VAE, outperform traditional codecs by delivering better image quality at similar bit rates. The hyperprior VAE achieves the highest PSNR, showing strong reconstruction even at low compression rates.

BPG closely follows and outperforms traditional methods, demonstrating that it remains an effective alternative. Classic approaches such as JPEG2000 and especially JPEG lag behind, with significantly lower PSNR values, particularly at low Bpp rates.

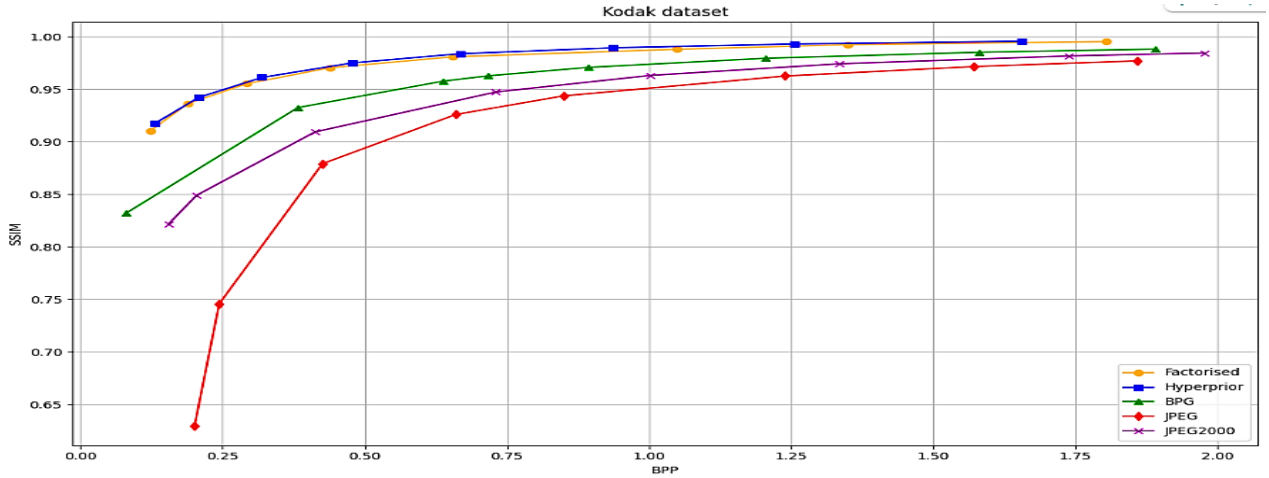

Figure 3.4. The RD performance of different image compression methods on Kodak dataset using SSIM metric.

Figure 3.4 shows us the performance of image compression methods when we use the SSIM metric. Figure 3.4 shows that DL methods effectively preserve visual structure during compression. Hyperprior VAE leads with the highest SSIM values, whereas traditional codecs like JPEG and JPEG2000 experience notable drops at low bitrates, reflecting weaker structural retention.

3.4.2 Tecnick_RGB_OR_1200x1200:

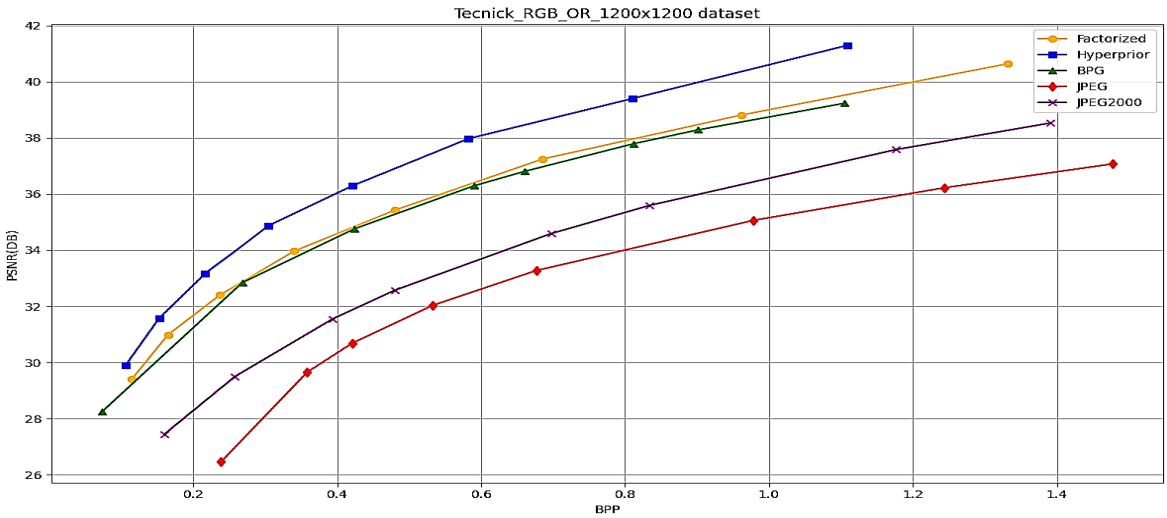

Figure 3.5. The RD performance of different image compression methods on Tecnick _RGB dataset using PSNR metric**.**

Figure 3.5 shows a comparison of image compression performance across different codecs. The evaluation based on PSNR versus Bpp shows that hyperprior VAE consistently achieves the highest PSNR across all bitrates, outperforming other codecs, especially at low rates. The factorized VAE follows closely, outperforming BPG, which remains competitive but less effective. JPEG 2000 performs better than JPEG but falls short of BPG's efficiency, while JPEG shows the weakest performance overall.

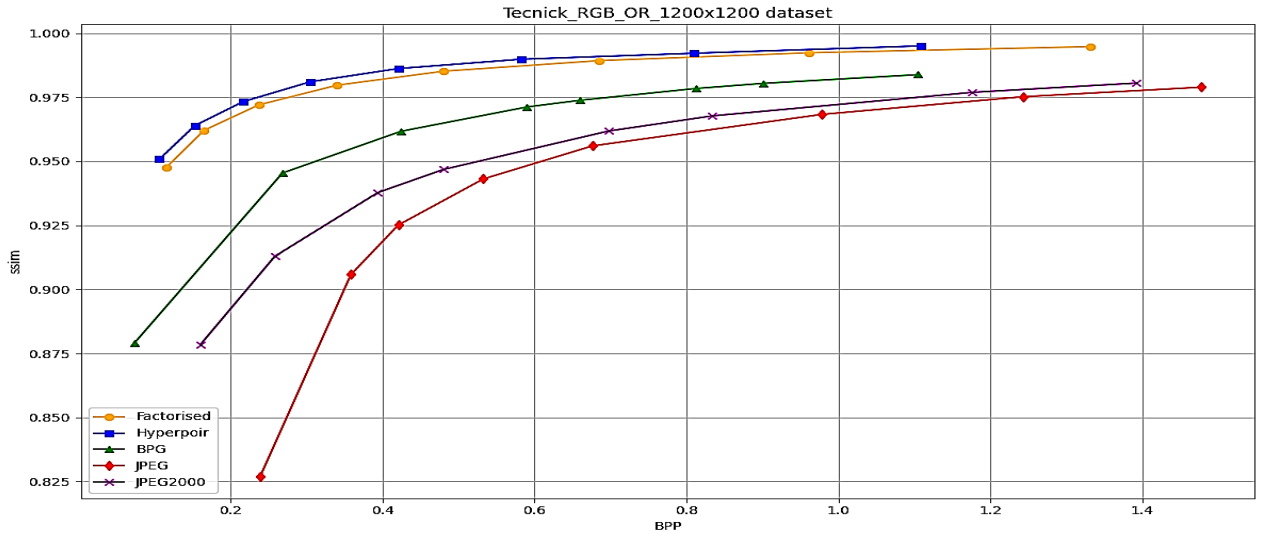

Figure 3. 6. The RD performance of different image compression methods on Tecnick _RGB using SSIM metric**.**

Figure 3.6 shows a comparison of image compression performance across different codecs. The RD performance is evaluated based on the SSIM against **Bpp.** Figure 3.6 shows us that hyperprior VAE and factorized VAE offer significantly better image quality (SSIM) at the same compression rate compared to traditional methods. The hyperprior VAE is the most effective, closely followed by the factorized VAE. Traditional methods like JPEG and JPEG2000 are considerably less efficient, while BPG falls somewhere between the DL-based method and the JPEG codec. This highlights the effectiveness of modern approaches to image compression.

3.5 Visual Performance

While objective metrics such as PSNR (dB) and SSIM provide valuable insights into the compression quality, they do not always reflect the human visual perception. Therefore, visual evaluation is essential to assess how compression affects the perceptual quality of the images.

In this section, we present a visual comparison using representative images from the Kodak dataset. The same image has been compressed using different techniques, allowing us to observe how each method preserves visual details, textures, and overall clarity. This qualitative analysis complements the numerical results and highlights perceptual differences that may not be evident through metrics alone.

Figure 3.7. Visual quality comparison between different image compression methods

To assess the visual quality of compressed images more effectively, we selected sample images from the Kodak dataset and zoomed into specific regions. This allows us to closely examine fine details and understand how each compression method affects the image's structure and clarity.

From the visual inspection of these enlarged areas, we observe that deep learning based compression methods, such as hyperprior VAE and factorized VAE, preserve the original image quality significantly better. The details remain sharp, and the structures are clearly visible even at high compression levels.

BPG, although not based on DL, performs notably well. It maintains more image detail than traditional methods and produces fewer artifacts, placing it between classical and learned approaches in terms of quality.

In contrast, traditional methods like JPEG and JPEG2000 tend to introduce noticeable artifacts, blurring, and loss of detail, especially under strong compression.

This visual comparison highlights the superiority of learned compression techniques, with BPG as a strong non-deep alternative, in maintaining both visual fidelity and structural consistency.

3.6 Discussion

- **Comparison between hyperprior VAE and factorized VAE:**

The Hyperprior VAE outperforms the Factorized VAE in terms of compression efficiency and visual quality. By leveraging additional side information for entropy modeling, the Hyperprior.

model achieves higher PSNR and SSIM values, especially at low bitrates. While both models are effective, the Hyperprior VAE demonstrates superior reconstruction accuracy and reduced artifacts, making it the more advanced of the two.

- **Comparison between traditional and DL-based compression techniques:**

Traditional compression methods like JPEG and JPEG2000 often cause visual artifacts such as blocking and blurring, particularly at low bitrates or with complex images, due to their reliance on non-adaptive algorithms. In contrast, DL-based models such as factorized VAE and hyperprior VAE learn adaptive representations from data, enabling them to better preserve image details and structure. Although requiring more computation during training, these models achieve significantly higher image quality and compression efficiency, marking a clear advantage over traditional codecs.

- **Objective metrics (PSNR vs. SSIM):**

Traditional techniques such as JPEG and JPEG2000 demonstrate limited ability to preserve image fidelity, especially at low compression rates. Their PSNR and SSIM values are generally lower, indicating more distortion and structural loss.

In contrast, DL-based methods like factorized VAE and hyperprior VAE achieve significantly better performance, maintaining higher image quality across a range of bitrates. These models consistently show stronger structural similarity and lower distortion levels.

- **Visual quality assessment:**

Visual inspection supports the quantitative results. DL methods provide sharper edges, clearer textures, and fewer artifacts compared to traditional codecs. In detailed image regions, they preserve content that is often degraded by classical methods.

Among traditional codecs, BPG offers a better trade-off between quality and compression than JPEG or JPEG2000, but it still does not reach the perceptual quality of deep learning approaches.

- **Training efficiency:**

The DL models were trained on a large and diverse dataset, with optimized hyperparameters and a well-structured learning schedule. The training process demonstrated a steady decrease in loss, indicating effective learning of compression patterns and a solid balance between data size and visual quality.

3.7 Conclusion

This study offers a thorough comparison between traditional and DL-based image compression techniques, supported by objective metrics and visual evaluations. The results clearly show that DL methods provide superior performance in terms of compression efficiency and perceptual quality, especially at lower bitrates.

Models such as hyperprior VAE and factorized VAE outperform traditional codecs in maintaining image fidelity and structural consistency. These findings are further supported by stable training behavior and effective generalizations, made possible through the use of a large and diverse dataset and well-chosen training parameters.

Although traditional methods remain relevant due to their simplicity and wide adoption, their limitations become apparent when compared to learned models. The consistent performance of deep compression techniques across different datasets confirms their growing potential as a new standard in the field.

General Conclusion

In this study, we explored and systematically compared the effectiveness of various image compression techniques, focusing on both traditional algorithms (JPEG, JPEG2000, BPG) and modern deep learning based approaches (factorized and hyperprior). We evaluated their performance in minimizing storage requirements while preserving visual quality, using objective metrics such as PSNR, SSIM, and bitrate.

Our findings clearly demonstrated that deep learning models significantly outperformed classical methods, especially at lower bitrates. The factorized and hyperprior architectures consistently delivered superior image quality and structural fidelity, as evidenced by their strong performance across both the Kodak and Tecnick benchmark datasets.

In addition to quantitative metrics, visual assessments confirmed that deep learning methods better preserved fine image details and introduced fewer artifacts compared to traditional codecs. While BPG emerged as a strong non-deep learning alternative offering better quality than traditional techniques, it still fell short of the perceptual quality achieved by learned models.

In conclusion, this study underscores the increasing potential of compression techniques based on deep learning as a robust and efficient alternative to traditional methods, positioning them as a promising foundation for the next generation of image compression standards. By leveraging advanced neural architectures such as the factorized and hyperprior models, these techniques not only achieve superior rate distortion performance but also demonstrate remarkable adaptability across diverse image datasets. As computational resources become more accessible, the integration of such models into real-world applications is expected to accelerate, offering enhanced visual quality, reduced storage demands, and improved transmission efficiency across various digital platforms.

References

[1] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Wiley-Interscience, 2006.

[2] A. J. Qasim, R. Din, and F. Q. A. Alyousuf, "Review on techniques and file formats of image compression," *Bull. Electr. Eng. Inform.*, vol. 9, no. 2, pp. 602–610, Apr. 2020, doi: 10.11591/eei.v9i2.2085.

[3] V. Beaudoin, "D´eveloppement de nouvelles techniques de compression de donn´ees sans perte," Ph.D. dissertation, Université Laval, 2009.

[4] A. Hacine Gharbi, "Chapitre2_codage de source," Département Télécommunications Université de Bordj Bou Arréridj, 2023.

[5] S. Benhizia and I. Aroub, "Compression d'images numériques avec et sans perte," Doctoral dissertation, Université Kasdi Merbah Ouargla, 2021.

[6] S. Ramasamy *et al.*, "Hardware optimization for effective switching power reduction during data compression in GOLOMB rice coding," *PLoS One*, 2024.

[7] H. Sather, "Golomb codes: An efficient coding scheme for integers," *Experience Stack*, Apr. 23, 2022. [Online]. Available: https://experiencestack.co/golomb-codes-d189eba4a64d

[8] M. Benabdellah, "Outils de compression et de crypto-compression : Applications aux images fixes et vidéo," Université Mohammed V–Agdal, Faculté des Sciences, Rabat, 2007.

[9] C. Sibade, Mohamed. Akil, Laurent. Perroton, and S. Barizien, "Stratégie d'Application de Traitement d'Image sur des Flux Compressés," 2003.

[10] A. M. Raid, W. M. Khedr, M. A. El-Dosuky, and W. Ahmed, "JPEG Image Compression Using Discrete Cosine Transform—A Survey," May 2014.

[11] "Compression image JPEG, Chapitre 4," Université Badji-Mokhtar Annaba, Faculté des Sciences de l'Ingéniorat, Département d'Informatique, S3, /2025 2024.

[12] D. Santa-Cruz, R. Grosbois, and T. Ebrahimi), "JPEG 2000 Performance Evaluation and Assessment," *Signal Processing: Image Communication*, Jan. 2002, doi: 10.1016/S0923-5965(01)00025-X.

[13] B. Lorincz, "ImageEngine Introduces JP2, JPEG2000 for Apple Devices," ImageEngine Blog. [Online]. Available: https://imageengine.io/imageengine-introduces-jp2-jpeg2000-for-apple-devices/

[14] M. M. Fleck, D. Forsyth, and C. Bregler, “Finding Naked People,” European Conference on Computer Vision (ECCV), pp. 593–602, 1996.

[15] C.-C. Chang and S.-Y. Lee, “Retrieval of Similar Pictures on Pictorial Databases,” *Pattern Recognition*, 1991.

[16] S. Paek *et al.*, “Integration of Visual and Text-Based Approaches for the Content Labeling and Classification of Photographs,” *Proceedings of the ACM SIGIR’99 Workshop on Multimedia Indexing and Retrieval*, 1999.

[17] Z.-N. Li, M. S. Drew, and J. Liu, *Fundamentals of Multimedia (2^{e} édition)*, Springer. Cham, Suisse, 2014.

[18] “An Overview of H.264 Advanced Video Coding.” [Online]. Available: https://www.vcodex.com/an-overview-of-h264-advanced-video-coding

[19] H. Sébastien, “Compression de contenus visuels pour transmission mobile sur réseaux de très bas débit,” Institut Polytechnique de Paris, 2020.

[20] M. Boumachta, “Study and Analysis of Multiview Video Coding,” Université Badji Mokhtar-Annaba, 2020.

[21] DataCamp, “How to Learn Deep Learning in 2025: A Complete Guide,” Feb. 29, 2024. [Online]. Available: Deep Learning Guide 2025

[22] P. Wang, E. Fan, and P. Wang, “Comparative analysis of image classification algorithms based on traditional machine learning and deep learning,” 2021.

[23] L. Zhang, S. Wang, and B. Liu, “Deep Learning for Sentiment Analysis.” National Science Foundation (NSF) and Huawei Technologies Co. Ltd., 2017.

[24] J. Markoff, “Scientists See Promise in Deep Learning Programs,” *New York Times*, Nov. 23, 2012.

[25] J. Zou, Y. Han, and S. S. So, “Overview of Artificial Neural Networks,” Artificial Neural Networks: Methods and Applications., 2009.

[26] A. Sharma, “Explain the basic architecture of a Neural Network, model training and key hyper-parameters,” AIML.com. [Online]. Available: https://aiml.com/what-is-the-basic-architecture-of-an-artificial-neural-network-ann/

[27] A. Zayegh and N. Al Bassam, “Neural Network Principles and Applications,” IntechOpen., 2018. [Online]. Available: https://doi.org/10.5772/intechopen.80416

[28] Tpoint Tech, “Gradient Descent in Machine Learning,” *Tpoint Tech*, Apr. 10, 2024. [Online]. Available: https://www.tpointtech.com/gradient-descent-in-machine-learning

[29] API4AI, “Mastering Deep Learning: Key Concepts and Its Impact on Image Processing,” *Medium*, Jul. 11, 2024. [Online]. Available: https://medium.com/@API4AI/mastering-deep-learning-key-concepts-and-its-impact-on-image-processing-1dc6d7ac0999

[30] S. M. Al-Selwi, M. F. Hassan, S. J. Abdulkadir, and A. Muneer, “LSTM Inefficiency in Long-Term Dependencies Regression Problems,” May 2023.

[31] Belcic, “Qu’est-ce qu’un modèle génératif ?,” *IBM*, Jan. 11, 2024. [Online]. Available: https://www.ibm.com/fr-fr/think/topics/generative-model

[32] S. Swapna, “Convolutional Neural Network | Deep Learning,” *Developers Breach*, Aug. 21, 2020. [Online]. Available: https://developersbreach.com/convolution-neural-network-deep-learning/

[33] DataCamp, “Introduction to Convolutional Neural Networks (CNNs),” *DataCamp*. [Online]. Available: https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns

[34] D. Bergmann and C. Stryker, “Qu’est-ce qu’un auto-encodeur ?,” *IBM*. [Online]. Available: https://www.ibm.com/fr-fr/think/topics/autoencoder

[35] R. Ritwek, “Auto-Encoders for Computer Vision: An Endless World of Possibilities.” [Online]. Available: https://www.analyticsvidhya.com/blog/2021/01/auto-encoders-for-computer-vision-an-endless-world-of-possibilities/

[36] J. Ballé, V. Laparra, and E. P. Simoncelli, “End-to-end optimization of nonlinear transform codes for perceptual quality,” Oct. 17, 2016, *arXiv*: arXiv:1607.05006. doi: 10.48550/arXiv.1607.05006.

[37] J. Ballé, V. Laparra, and E. P. Simoncelli, “Density Modeling of Images using a Generalized Normalization Transformation,” Feb. 29, 2016, *arXiv*: arXiv:1511.06281. doi: 10.48550/arXiv.1511.06281.

[38] V. Laparra, J. Ballé, A. Berardino, and E. P. Simoncelli, “Perceptual image quality assessment using a normalized Laplacian pyramid,” *Proceedings of SPIE, Human Vision and Electronic Imaging XXI*, 2016.

[39] J. Ballé, V. Laparra, and E. P. Simoncelli, “End-to-End Optimized Image Compression,” Mar. 03, 2017. [Online]. Available: https://arxiv.org/abs/1703.00395

[40] J. Ballé, S. J. Hwang, D. Minnen, N. Johnston, and . Singh, “Variational Image Compression with a Scale Hyperprior,” May 01, 2018.

[41] M. Al-Ani and F. H. Awad, “The JPEG Image Compression Algorithm,” May 2013. [Online]. Available: https://www.researchgate.net/publication/268523100

[42] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek, “An Overview of JPEG-2000,” 2000. [Online]. Available: https://doi.org/10.1109/DCC.2000.838192

[43] F. Bellard, “Better Portable Graphics,” 2014. [Online]. Available: https://bellard.org/bpg/

[44] Flickr, “Flickr – Photos tagged with ‘flicker.’” [Online]. Available: https://www.flickr.com/photos/tags/flicker/

[45] S. Miller, “Kodak Lossless True Color Image Suite.” [Online]. Available: https://r0k.us/graphics/kodak/

[46] TestImages Project, “SAMPLING 8BIT RGB 1200x1200.” [Online]. Available: https://sourceforge.net/projects/testimages/files/SAMPLING/8BIT/RGB/SAMPLING_8BIT_RGB_1200x1200.tar.bz2/download

[47] A. Horé and D. Ziou, “Image Quality Metrics: PSNR vs. SSIM,” *20e Conférence internationale sur la reconnaissance des formes (ICPR 2010)*, 2010. [Online]. Available: https://projet.liris.cnrs.fr/imagine/pub/proceedings/ICPR-2010/data/4109c366.pdf