RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE POLYTECHNIQUE

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

EURECOM
Sophia Antipolis

Département d'Électronique

End-of-study project dissertation

for obtaining the State Engineer's degree in Electronic

The impact of the new image compression scheme JPEG AI on image analysis tasks

---

TEMMAR Mohamed Riadh

Under the supervision of: **Dr. BERRANI Sid-Ahmed** ENP
Pr. DUGELAY Jean-Luc EURECOM

Presented publicly 21/06/2023

Jury's composition:

President:	Pr.ADNANE Mourad	ENP
Supervisor:	Dr.BERRANI Sid-Ahmed	ENP
Examinator:	Dr.BOUADJENEK Nesrine	ENP

ENP 2023

Électronique / ENP, 10 Rue des Frères OUDEK, El Harrach 16200 Alger Algérie.

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE POLYTECHNIQUE

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

EURECOM
Sophia Antipolis

Département d'Électronique

End-of-study project dissertation

for obtaining the State Engineer's degree in Electronic

The impact of the new image compression scheme JPEG AI on image analysis tasks

---

TEMMAR Mohamed Riadh

Under the supervision of: **Dr. BERRANI Sid-Ahmed** ENP
Pr. DUGELAY Jean-Luc EURECOM

Presented publicly 21/06/2023

Jury's composition:

President:	Pr.ADNANE Mourad	ENP
Supervisor:	Dr.BERRANI Sid-Ahmed	ENP
Examinator:	Dr.BOUADJENEK Nesrine	ENP

ENP 2023

Électronique / ENP, 10 Rue des Frères OUDEK, El Harrach 16200 Alger Algérie.

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE POLYTECHNIQUE

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

EURECOM
Sophia Antipolis

Département d'Électronique

Mémoire de projet de fin d'études

Pour l'obtention du diplôme d'ingénieur d'état en Électronique

---

The impact of the new image compression scheme JPEG AI on image analysis tasks

---

TEMMAR Mohamed Riadh

Sous la direction de: **Dr. BERRANI Sid-Ahmed** ENP
Pr. DUGELAY Jean-Luc EURECOM

Présenté et soutenu publiquement le 21/06/2023

Composition du jury :

Président :	Pr.ADNANE Mourad	ENP
Promotrice :	Dr.BERRANI Sid-Ahmed	ENP
Examinatrice :	Dr.BOUADJENEK Nesrine	ENP

ENP 2023

Électronique / ENP, 10 Rue des Frères OUDEK, El Harrach 16200 Alger Algérie.

ملخص

تلعب ضغط الصور دورًا حيويًا في تخزين ونقل الوسائط الرقمية. بالإضافة إلى الأساليب التقليدية للضغط، هناك تطورات حديثة في تقنيات الذكاء الاصطناعي. تم تصميم هذه الطرق بهدف تحقيق أهداف محددة، مثل إعادة إنشاء الصور بشكل محسن أو استخدام التمثيلات الكامنة لمهام رؤية الحاسوب. في هذه الدراسة، نستكشف التباينات بين هذه الأطراف التشفيرية التي تعتمد على الذكاء الاصطناعي وفقًا لأهدافها من خلال معالجة مشكلة تصنيف. بعد ذلك، نركز على إنشاء ضاغط صور محسن قادر على أداء ثلاث مهام : ضغط الصور ورؤية الحاسوب ومعالجة الصور. على وجه التحديد، قمنا باختيار التعرف على الوجوه وتضاعف الدقة كمهام ثانوية إلى جانب ضغط الصور.
الكلمات المفتاحية : الذكاء الاصطناعي، ضغط الصور، التعرف على الوجوه، معالجة الصور، رؤية الحاسوب

Résumé

La compression d'images joue un rôle essentiel dans le stockage et la transmission des médias numériques. En plus des méthodes de compression traditionnelles, il y a eu récemment des avancées dans les techniques basées sur l'intelligence artificielle (IA). Ces méthodes sont conçues dans le but d'atteindre des objectifs spécifiques, tels que la reconstruction optimisée d'images ou l'utilisation de représentations latentes pour des tâches de vision par ordinateur. Dans cette étude, nous explorons les variations entre ces codecs basés sur l'IA en nous attaquant à un problème de classification. Ensuite, nous nous concentrons sur la création d'un compresseur d'images amélioré capable d'effectuer trois tâches: la compression d'images, la vision par ordinateur et le traitement d'images. Plus précisément, nous avons choisi la reconnaissance faciale et le doublement de la résolution comme tâches secondaires, en plus de la compression d'images.
Mots clés: Intelligence artificielle, Compression d'images, Reconnaissance faciale, Traitement d'images, Vision par ordinateur.

Abstract

Image compression plays a vital role in storing and transmitting digital media. In addition to traditional compression methods, there have been recent advancements in AI-based techniques. These methods are designed with specific objectives in mind, such as optimized image reconstruction or utilizing latent representations for computer vision tasks. In this study, we explore the variations among these AI-based codecs based on their objectives by tackling a classification problem. following that we focuses on creating an enhanced image compressor capable of performing three tasks: image compression, computer vision, and image processing. Specifically, we chose face recognition and resolution doubling as secondary tasks alongside image compression.
Keywords: Artificial intelligence, Image compression, face recognition, image processing, computer vision.

Acknowledgements

I would also like to extend my heartfelt appreciation to my supervisors, Sid-Ahmed BERRANI and Jean-Luc DUGELAY, for their continuous guidance and support throughout the entire journey of this thesis. Their availability and invaluable advice played a crucial role in the successful completion of this work.

I would like to dedicate this achievement to my parents, my grandmother, and my two aunts who have been unwavering in their support and assistance throughout my educational journey. During challenging times, they stood by my side, offering all the support and encouragement.

Additionally, I would like to express my gratitude to my friends: Samy, Anes, Idriss, Maroua, Romeila, Faten, Chaeloula and all my classmates ELN match and all the members of VIC for being an integral part of this three-year journey. Their presence and camaraderie have enriched my academic experience. and I would like also to thank TopG.

I am also thankful to Mr. Mourad ADNANE, for agreeing to chair my thesis jury, and to Ms. Nesrine BOUADJENEK, my examiner, for showing interest in my work.

Mohamed Riadh.

Contents

List of Tables

List of Figures

List of acronyms

- **ANN** : Artificial Neural Network
- **CNN** : Convolutional Neural Network
- **CPU** : Central Processing Unit
- **GPU** : Graphics Processing Unit
- **DNN** : Deep Neural Network
- **AE** : Auto-Encoders
- **VAE** : Variational auto-encoders
- **DCT** : Discrete cosine transform
- **GAN** : Generative adversarial network
- **SSIM** : Structural Similarity Index
- **MSSIM** : Multi-Scale Structural Similarity Index

Chapter 1

Introduction

1.1 Introduction

Performing inference of deep learning models that process images on embedded devices is a difficult task due to limited computational resources. One popular alternative is to conduct model inference in the cloud, necessitating the transmission of images from the embedded device. To reduce latency over low-bandwidth networks, image compression techniques are commonly employed in cloud-based architectures. This study firstly investigate how neuronal image compression is effected when it is optimized for machine vision tasks compared to reconstruction only. Then it introduces a new JPEG AI model that can fulfill the 3 requirements, i.e introducing a model that is capable of compressing and decompressing, preform computer vision task and image processing.

1.2 Research objective

In our research, our primary objective is to explore and analyze the behavior of self-pretrained encoders obtained from state-of-the-art models. We specifically investigate how the encoder's latent representation varies when it is trained for compression alone or for both compression and computer vision tasks. Our aim is to quantify the extent of change in these latent representations. In the context of this thesis, we select classification as the computer vision task. To accomplish this, we simultaneously optimize the encoder and classifier together. Additionally, we conduct a parallel optimization where we solely focus on optimizing the classifier. Upon obtaining the results, we observe an intriguing finding. Despite deliberately choosing two extreme scenarios, we find that the latent representations exhibit a remarkable degree of similarity, as measured by cosine similarity. This finding serves as a motivation to develop a new framework adapted to JPEG AI, which possesses the ability to perform three tasks concurrently: functioning as a CODEC in itself, enhancing image resolution by doubling its resolution, and adapting the latent representation to facial recognition.

1.3 Thesis outline

To begin the thesis, we will start by exploring some fundamental concepts. We will delve into the widely known jpeg compressor and the classic loss for neural image compression. Following this, we will delve into deep learning principles, including generative adversarial neural networks and variational auto-encoders. We will also explain some information theory concepts, such as Shannon's entropy, which plays a crucial role in applying lossless encodings. In the subsequent

chapter, we will examine state-of-the-art models that are essential for this work. Next, we will conduct our initial experiment, which involves training an encoder and classifier for loss accuracy. Simultaneously, we will train a classifier exclusively for loss accuracy. We will then compare the outcomes of both approaches. Moving forward, in the following chapter, we will develop a model that fulfills the three requirements of JPEG AI. Specifically, we will focus on augmenting resolution and adapting latent representation for face recognition. This section will showcase relevant results along with corresponding metrics. Concluding the thesis, we will summarize our findings and present a general conclusion. Additionally, we will outline potential future work that can be explored in this field.

Chapter 2

Fundamental tools and related works

2.1 Introduction

In this opening chapter, we will embark upon an extensive exploration of multiple preliminary concepts that are essential for the reader to acquire a comprehensive understanding of the subject matter presented in this thesis. We will start by delving into the foundational principles underlying classic image compression, allowing us to establish a solid groundwork for our subsequent discussions. Subsequently, we will shift towards neural networks and information theory, elucidating the synergistic relationship between these domains and shedding light on pertinent topics such as Generative Adversarial Networks (GANs) variational auto-encoders and Shannon's entropy. Moreover, we will investigate diverse metrics that are commonly employed in the realm of image processing. Among these metrics, we will place notable emphasis on the Structural Similarity Index (SSIM), a pivotal tool that plays an indispensable role in the evaluation of image quality and performance of image CODECs. By examining these metrics and AI notions, we aim to provide a comprehensive overview of the intricate mechanisms and quantitative measures employed in the domain of image compression, thereby enabling the reader to develop a nuanced understanding of the subject matter at hand.

2.2 Image compression

Image compression refers to the process of compressing digital images to decrease their storage or transmission costs and also the computational bandwidth when treating them.
Unlike generic data compression techniques used for other forms of digital data, image compression algorithms can leverage both visual perception and statistical characteristics of the image data to achieve more effective results.
The well known image compressor JPEG [1] and JPEG2000 [2] were created and optimized for photographic images in opposition to PNG and other compression for another type of images, JPEG is more successful compared to its predecessor since it could take a lot of color into consideration and a better quality compression ratio. Through history image compression for photographic images uses lossy and lossless compression techniques such as quantification of DCT coefficients but also huffman encodings. In recent years new approaches and algorithms are being developed using machine learning algorithms. They are known for JPEG AI, those algorithms showed much improvement in terms of quality compression ratio but on the other side have more computational cost.

2.3 JPEG Joint Photographic Experts Group

JPEG is a lossy compression algorithm used for photographic images that can be modified to achieve different reconstruction qualities with different compression ratios.

It can achieve a 1:10 compression ratio with almost invisible quality distortion. It was introduced as a norm in 1992 [1] by the committee active by the same name. The most common variant of JPEG encoding is JFIF (JPEG File Interchange Format) which consists of 5 steps that are the following:

1 - Color Space Conversion: The first step in the JPEG algorithm is to convert the image from its original color space (usually RGB or CMYK) to the YCbCr color space. This conversion that can be visualized in figure 2.1 separates the brightness (Y) and color information (Cb and Cr) of the image, which allows for more efficient compression.

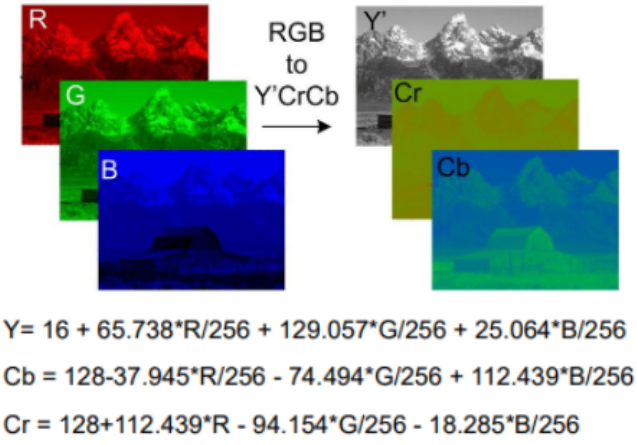

Figure 2.1: Color conversion process from RGB to YCrCb.

3 - Discrete Cosine Transform (DCT): The DCT is a mathematical transformation that converts the pixel data from the spatial domain to the frequency domain [3]. In the spatial domain, the pixel values represent the image as it appears to the human eye. In the frequency domain, the pixel values represent the image as a set of frequency components. The DCT is applied to small blocks of the image data, typically 8x8 pixels in size. Each block of pixel data is transformed into a set of DCT coefficients, which represent the contribution of each frequency component to the block.

4 - Quantization: Quantization in the equation 2.1 is a process that reduces the precision of the DCT coefficients by dividing them by a set of quantization values. The quantization values are determined based on the desired level of compression, with higher values resulting in more lossy compression.

$$B_{j,k} = round(\frac{G_{j,k}}{Q_{j,k}}) \tag{2.1}$$

5 -Huffman Encoding: Huffman encoding [4] is a lossless compression technique that assigns shorter codes to more frequently occurring values. In the JPEG algorithm, Huffman encoding is applied to the quantized DCT coefficient.

In order to decompress we use the inverse path, i.e multiply by Q matrix applying inverse DCT, and so on, until we get the reconstructed image.

2.4 AI-based image compression (JPEG AI)

In recent years image compression algorithms that use deep learning have gained more attention since they surpassed anchors algorithms performance for very low bitrates. The aim behind JPEG AI is to make image compressors do additional tasks, such as computer vision ones (Classification, Detection, ...) and also preprocessing (ex: Denoising). One of the advantages of JPEG AI is that it can be customized for specific applications. For example, the compression algorithm can be optimized for different types of images, such as photographs, graphics, or text. This makes it possible to create compressed images that are tailored to the specific needs of a particular application, but they also can take advantage of the compressed representation directly for machine vision tasks since it hold the necessary information to reconstruct the original image, which will be the main focus of the study. Those algorithms use different architectures to play the role of encoders and decoders such as GANs, AEs, VAEs. then applying lossless encodings to the compact representation in order to decrease the bit per pixel metric. The classic loss function for those models is divided into two parts, the reconstruction loss and the entropy loss explained in the following subsection.

2.4.1 Rate-Distortion loss:

The aim of lossy image compression has traditionally been to strike a balance between reducing the expected length of the bit-stream that represents the input image, x, and minimizing the expected distortion between the image and its reconstructed version, $\hat{x}$. This problem is commonly referred to as the rate-distortion optimization problem in information theory literature [5]. In the domain of learned image compression, this objective can be expressed as a multi-task loss:

$$\mathcal{R}(\mathbf{Q}(f(x))) + \lambda \cdot \mathcal{D}(g(\mathbf{Q}(f(x))), x) \tag{2.2}$$

where R represents the rate or bit-stream length, and D represents the distortion or reconstruction fidelity. The non-linear transform coding strategy [6] involves using an encoder to generate a discrete latent representation, denoted as z, from an input image x. This representation is obtained through a non-linear function, specifically $z = f(x)$. The real-valued representation z is then quantized using a quantization operator Q to obtain $\hat{z}$. The image decoder $g(\hat{z})$ is responsible for generating the reconstructed input, denoted as $\hat{z}$, based on the quantized representation $\hat{z}$. It is important to note that in order to encode the representation $\hat{z}$ into a bit-stream using a lossless compression algorithm like arithmetic coding [7], $\hat{z}$ must belong to a finite set. This requirement ensures that the encoding process can be performed without any loss of information.

2.5 Deep Learning

Deep learning is a subfield of machine learning that uses artificial neural networks to model and solve complex problems. It is inspired by the structure and function of the human brain and is designed to process large amounts of data to identify patterns, classify information, and make predictions. It can be modeled by ANNs (artificial neural networks), the basic building block of an artificial neural network is the neuron, which receives input signals, processes them using an activation function, and generates an output signal. Neurons are organized into layers, with each layer receiving input from the previous layer and producing output for the next layer. The input layer receives the initial data, and the output layer produces the final predictions or decisions. Between the input and output layers, there can be one or more hidden layers that perform intermediate processing.

Deep learning models can also be seen as a complex function that is defined by a set of parameters. The reason it is referred to as a "deep" model is that it typically comprises multiple layers, where the output of each layer serves as the input for the subsequent layer. Additionally, it is called a "learning" model since the values of the set of parameters are not predetermined but instead learned from data automatically through the use of the back-propagation algorithm.

We can also see it as an optimization problem where we try to minimize the loss function which is a distance between the value predicted by the model and the real value. The way to minimize a multidimensional function is to use the gradient operator (where the name gradient descent

comes from) in an iterative form where at each step we get closer to the minimal value of the loss by changing the parameters in the right way.

2.5.1 Neural networks

Deep learning involves correcting errors based on the importance of the elements that actually contributed to those errors. In the case of neural networks, the synaptic weights that contribute to generating a significant error will be modified more significantly than the weights that generated a minor error. We can see in the following figure the evolution of deep learning:

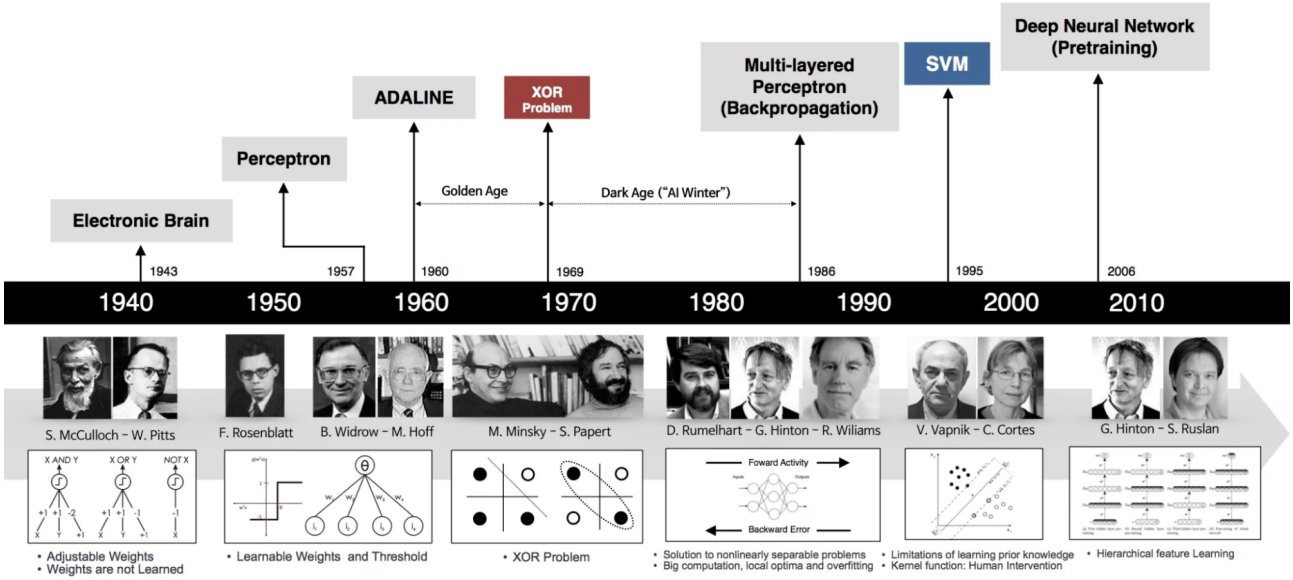

Figure 2.2: An in-depth exploration of the evolution and advancements of neural networks over time.

2.5.2 Perceptron

A perceptron neuron (shown in figure 2.3) is a type of artificial neuron that is used in artificial neural networks. It was developed in the 1950s by Frank Rosenblatt, and is a simple algorithm for supervised learning of binary classifiers. It takes in multiple input values, and each input is assigned a weight then calculates a weighted sum of the inputs, adds a bias term, and applies an activation function to produce an output. which will be the final output of the network. [8], and can be expressed like the following:

$$\hat{y} = \sum_{i=1}^{n} x_i \omega_i + b \tag{2.3}$$

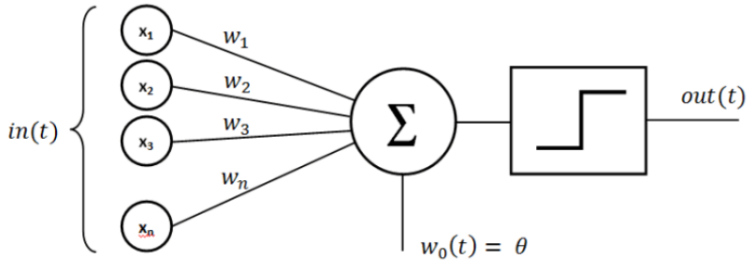

Figure 2.3: Perceptron algorithm.

2.5.3 Multilayer perceptron MLP

The multilayer perceptron shown in figure 2.4 an extended version of the single-layer perceptron. It is composed of an input layer, several hidden layers, and an output layer, where each neuron in a layer is connected to all neurons in the adjacent layer, thus building a fully connected network of at least three layers. The number of neurons in the input layer defines the dimension of the input feature vector, while the number of neurons in the output layer defines the number of classes. The number of neurons in the hidden layers is considered a design question. The fewer hidden neurons, the more difficult it is for the model to learn complex decision boundaries. On the other hand, the more hidden layers there are, the less generalized the model becomes, and it specializes.

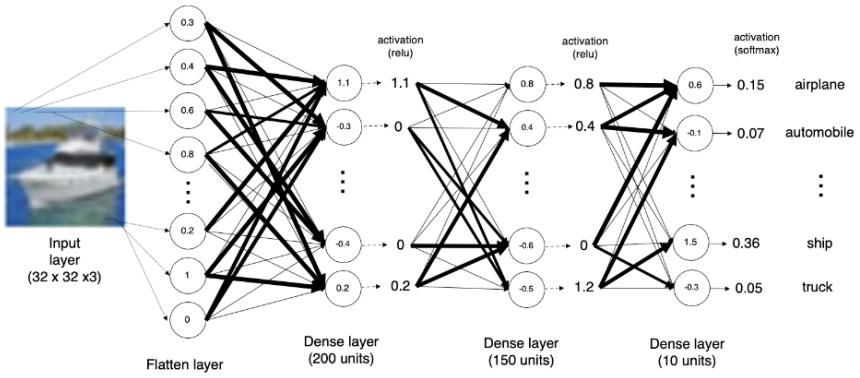

Figure 2.4: Multilayer perceptron algorithm.

MLPs are not designed to handle spatial information and are generally used for tasks that do not require spatial processing, such as predicting numerical outputs from a set of input features. In contrast, CNNs are designed to efficiently process spatial information, making them better suited for tasks such as image classification and object detection.

2.5.4 Backpropagation algorithm

It is a process for computing the gradient of the loss function with respect to the model's weights, which can then be used to update the weights to minimize the loss.

2.5.5 Gradient decent

The standard approach to train a neural network involves using gradient descent [9]. This method relies on a loss function $l(\theta)$ that quantifies how well the network's weights and biases θ fit the training data. A higher loss indicates poor performance and more errors, while a lower loss suggests better performance. To minimize the loss, the network parameters are adjusted iteratively. In the gradient descent algorithm, the update rule is formally expressed as:

$$\theta_{t+1} = \theta_t - L_r \nabla l(\theta) \tag{2.4}$$

This equation involves taking the gradient ∇ of the loss function l, which indicates the direction in parameter space that increases the loss. To decrease the loss, the update is performed in the opposite direction, denoted by ∇l. The step size of the update is determined by the learning rate l_r. It is typically a small fraction that ensures the model optimizes its parameters in smaller steps, preventing overshooting of loss minima. This helps the learning algorithm converge steadily towards reducing the error, rather than oscillating around a local minimum. The gradient descent method forms the foundation of many modern deep learning techniques and is widely effective in practice.

2.5.5.1 Stochastic gradient decent

Stochastic gradient descent (SGD) is an iterative optimization algorithm used in machine learning to train models. Unlike traditional gradient descent, which updates model parameters based on the gradients computed over the entire training dataset, SGD updates the parameters using a randomly selected subset, known as a mini-batch, at each iteration. This stochastic sampling introduces noise and variability into the updates, but it also provides computational efficiency, particularly for large datasets. By updating the parameters more frequently, SGD can converge faster and potentially escape local minima. However, the noise introduced by stochastic sampling can make the convergence path less smooth. To address this, variations of SGD, such

as mini-batch SGD and momentum-based methods, are commonly used, striking a balance between efficiency and stability.

2.5.6 Optimization

In machine learning, optimization refers to the process of finding the best set of model parameters or variables that minimize or maximize an objective function. The objective function represents the goal or criterion that the model aims to optimize, such as reducing the error or maximizing the accuracy of predictions. Optimization algorithms in machine learning iteratively adjust the model parameters based on the information obtained from the training data. These algorithms search for the optimal values of the parameters by evaluating the objective function and updating the parameters in a way that moves closer to the optimal solution. The choice of optimization algorithm depends on various factors, such as the nature of the objective function, the size of the dataset, and the complexity of the model. Common optimization algorithms used in machine learning include gradient descent, stochastic gradient descent, Adam, and LBFGS, among others. The goal of optimization in machine learning is to find the best model parameters that generalize well to unseen data, thereby improving the model's performance and predictive accuracy.

2.5.7 Transfer learning

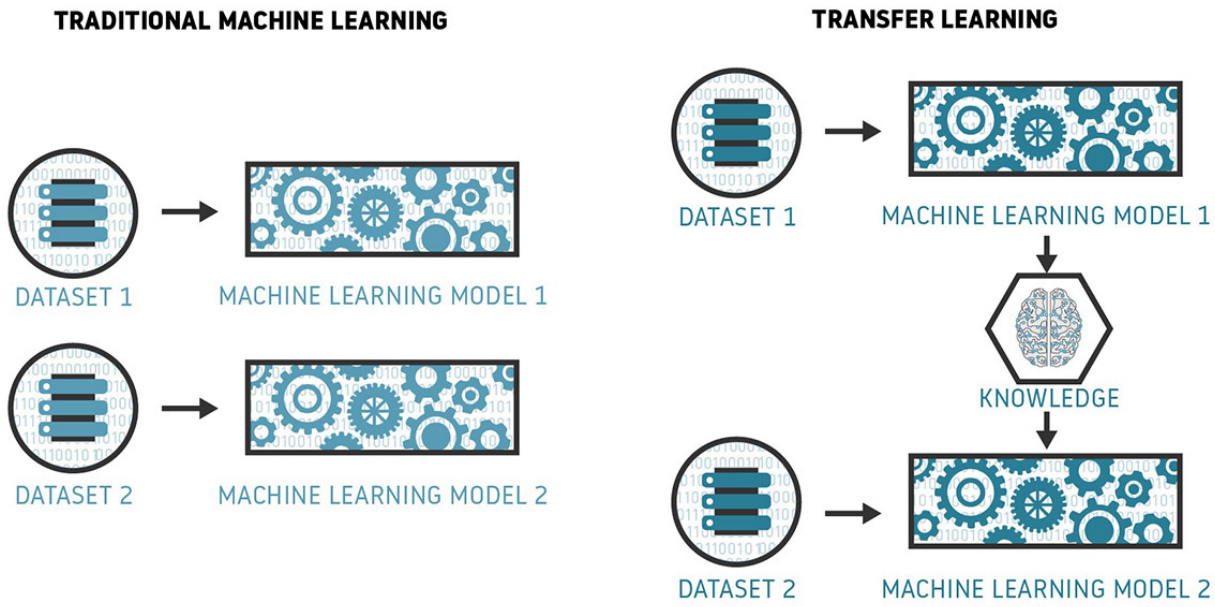

Figure 2.5: Transfer learning scheme.

Transfer learning is a machine learning technique that leverages knowledge gained from one task

to improve performance on another related task it can be seen in figure 2.5. Rather than starting from scratch, transfer learning utilizes pre-trained models or learned representations from a source task and applies them to a target task. By leveraging the learned features, relationships, and patterns captured by the source task, transfer learning can accelerate and enhance the learning process on the target task, especially when the target task has limited labeled data. This approach allows models to generalize better and achieve improved performance in situations where data availability is limited. Transfer learning has been widely adopted in various domains, such as computer vision and natural language processing, enabling the development of robust and effective models with less effort and data.

2.5.8 Convolutional neural networks

Convolutional Neural Networks (CNNs or ConvNets) are an extension of MLPs that effectively overcome some of their shortcomings. They are designed to automatically extract features from input data. They were initially inspired by the discovery made by Hubel and Wiesel in 1962, where they noticed that specific patterns stimulated activity in specific parts of the brains of cats and monkeys.

The first use of these networks was made by Fukushima with the self-organizing map (SOM) for feature extraction using unsupervised learning. In the field of pattern recognition, significant development was made by LeCun et al. where they proposed an architecture, called LeNet, for recognizing handwritten digits.

The CNN eliminates the need for manual feature extraction. It extracts features and assigns memorizable weights and biases describing various aspects of the image to differentiate them from each other. Technically, each input image passes through two blocks of layers. The first block contains the convolutional layers, which are responsible for generating the features, and the second block is the classification block, which contains fully connected layers, similar in architecture to the MLP. The following figure visualize the CNN as feature extractor:

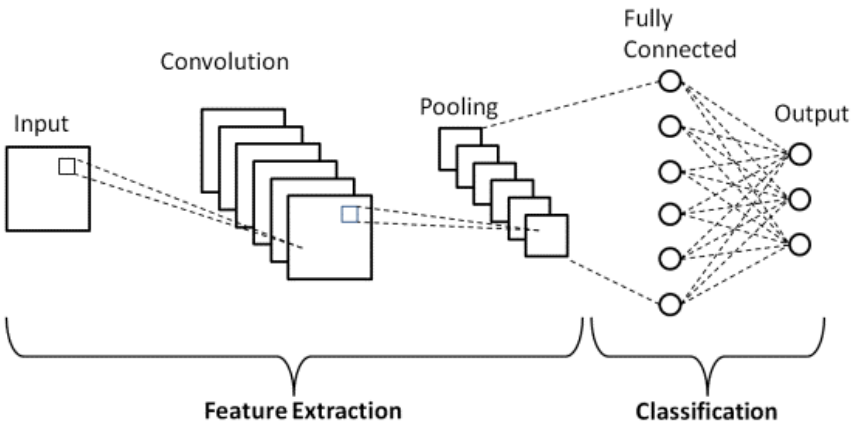

Figure 2.6: CNN With fully connected neural network.

2.5.8.1 Feature extraction

This block can play the role of feature extractor without doing it manually but also as a compressor for the information in the context of image compression, since it extracts the necessary information to reconstruct the original image with the right training. Every layer in this block is considered as a feature map which can be used according to the needs, the more we get deeper into the blocks the more the features get specific for a given task. The way those features are extracted is using the convolution matrix which will be detailed in the next paragraph. [10]

Convolutions:

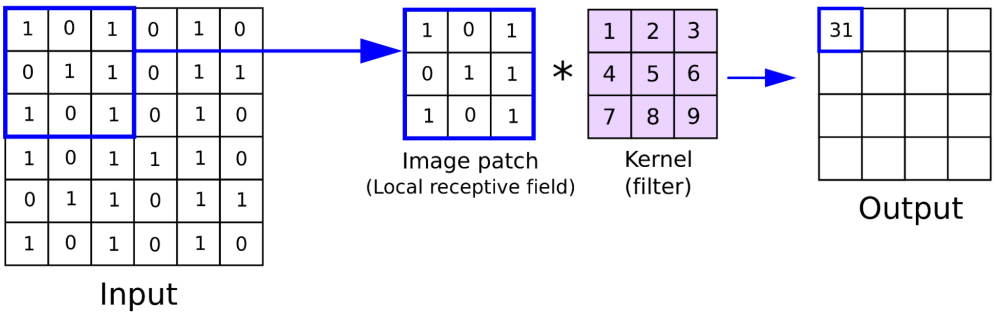

Figure 2.7: Convolution process for one dimensional matrice.

In the case of figure 2.7 we have two dimensional convolution with a 3x3 kernel the formula for the convolution for a matrix $x(a, b)$ and a kernel is the following:

$$x_{ij} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} k_{ab} f_{(i+a)(j+b)} \tag{2.5}$$

And the kernel coefficients are the weights that must be trained in order to extract the right information (ex: edge detection). And it's not limited to only one channel or one kernel, we can use multiple of those at the same time and we will get different feature maps at the same layer.

Padding:

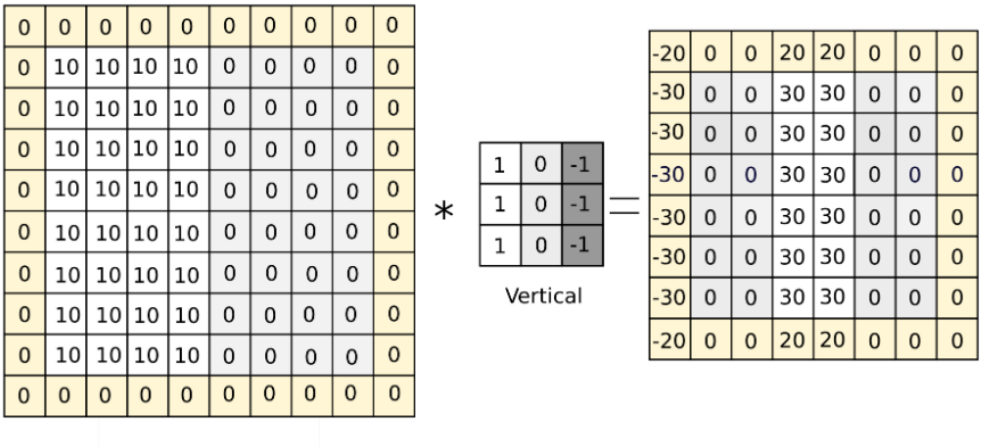

Figure 2.8: An example of same convolution with padding $p = 1$.

If we apply convolution directly to an image of size HxH with a kernel of WxW we will get an image of size $(H - W + 1) \times (H - W + 1)$ which will reduce the size of the output, and we will lose the information contained in the edges on the input. In order to overcome this we will use padding (can be seen in figure 2.8) which consists of adding additional lines and columns to the original matrix whether with zeros or another value.

Pooling: A downsampling operation shown in figure 2.9 that reduces the spatial dimensions of the input feature maps. It is typically applied after convolutional layers to capture and preserve the most important features while discarding unnecessary details.

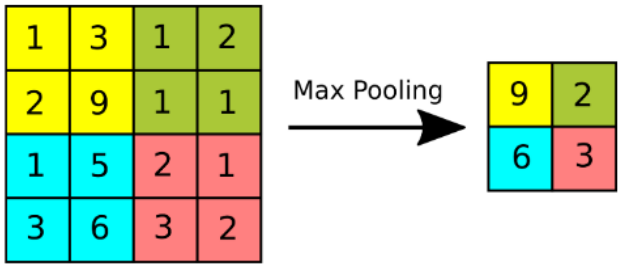

Figure 2.9: Pooling Layer 4:2.

2.5.9 Generative adversial networks

A novel method for training a generative model is the introduction of Generative Adversarial Networks (GANs) [11]. These networks consist of two adversarial models, a generative model (G) that captures the data distribution and a discriminator model (D) that estimates the likelihood that a given sample came from the training data rather than G. Both G and D can be non-linear mapping functions, such as multi-layer perceptrons. The following figure shown in depth the architecture of GANs:

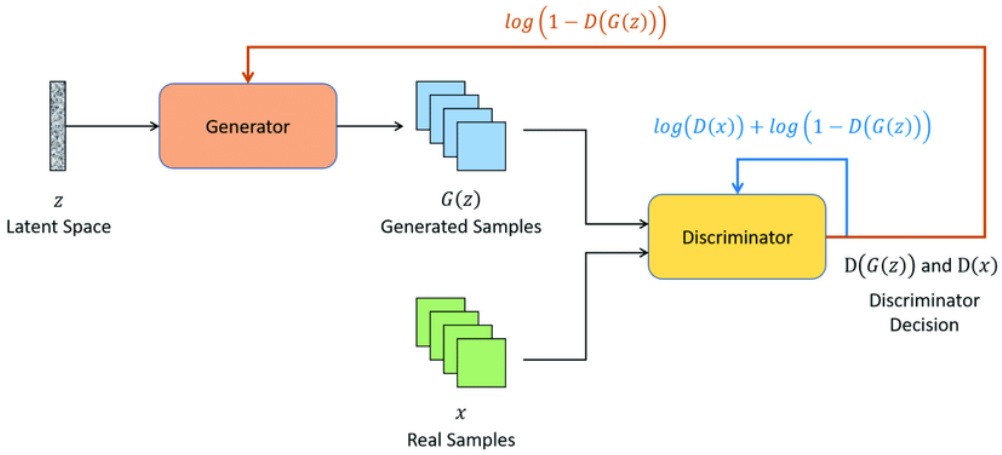

Figure 2.10: Generative adversial networks architecture and their respective losses.

To generate a distribution p_g over data x, the generator constructs a mapping function from a prior noise distribution $p_z(z)$ to data space as $G(z; \theta g)$. On the other hand, the discriminator $D(x; \theta d)$ outputs a single scalar that represents the probability that x came from the training data rather than p_g. The two models, G and D, are trained simultaneously through a two-player min-max game with value function $V(G, D)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{datq}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \tag{2.6}$$

G's parameters are adjusted to minimize $\log(1 - D(G(z)))$, while D's parameters are adjusted to minimize $\log D(X)$ [12].

2.5.9.1 Conditional GANs

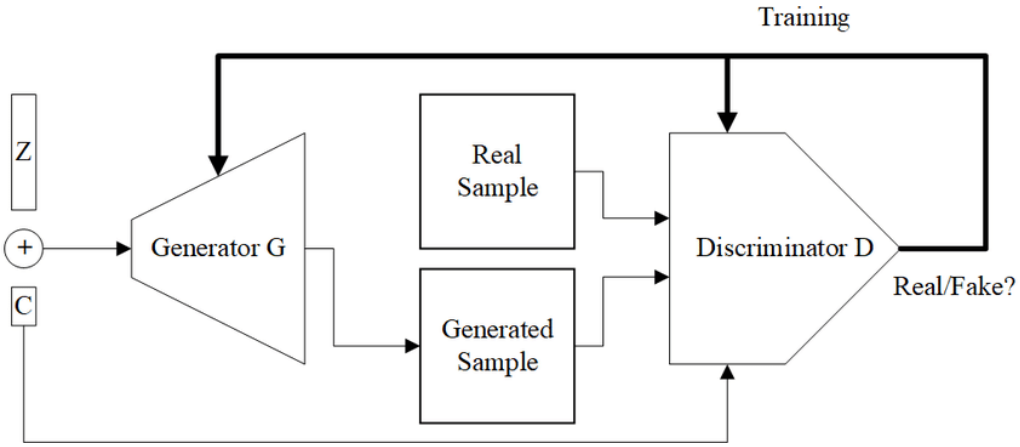

Figure 2.11: Conditional Generative neural networks and their training process.

By incorporating auxiliary information y, such as class labels or data from other modalities, generative adversarial nets can be transformed into a conditional model [13]. This is achieved by introducing y as an additional input layer to both the generator and discriminator. The generator combines the prior input noise $p_z(z)$ and y to form a joint hidden representation, while the discriminator takes in x and y as inputs and applies a discriminative function using a multi-layer perceptron (MLP). The trainging processes is visualized in figure 2.11.

The adversarial training framework provides significant flexibility in how the hidden representation is composed. The objective function for this two-player minimax game is expressed as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}\,(\boldsymbol{x})}[\log D(\boldsymbol{x} \mid \boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z} \mid \boldsymbol{y})))] \tag{2.7}$$

2.5.10 Autoencoders

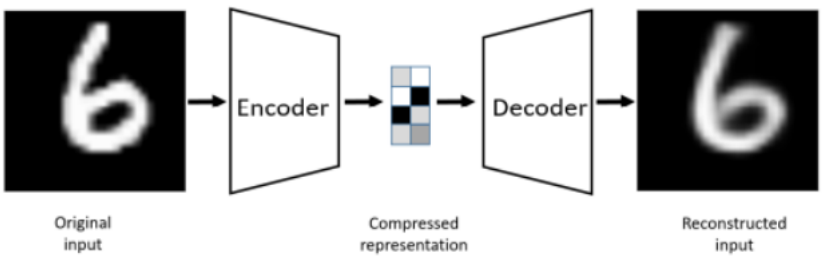

Figure 2.12: Autoencoders architecture containing encoder network on the left and decoder network on the right for digit reconstruction.

Autoencoders (shown in figure 2.12) are a type of neural network architecture that can be used for unsupervised learning, dimensionality reduction, and data compression [14]. An auto-encoder consists of an encoder network that maps an input to a latent space representation, and a decoder network that maps the latent space representation back to the original input. During training, the auto-encoder learns to minimize the reconstruction error between the input and the output:

$$argmin_{A,B} E[\Delta x, B(A(x)] \tag{2.8}$$

In the context of image compression, its error is a distance between the original image and its reconstruction for example using MS SSIM since it's not very significant to compare pixel per pixel using MSE.

2.5.11 Variational Autoencoders

When considering the connection between auto-encoders and content generation, a common question arises: "What role do auto-encoders play in generating new content?" Once an auto-encoder is trained, it consists of two main components: an encoder and a decoder. However, there is no straightforward method to produce entirely new content using the trained auto-encoder. Instead of directly mapping the input data to the latent space, the VAE introduces a probabilistic approach [15]. It assumes that each point in the latent space is generated from a probability distribution. The distribution is typically assumed to be a multivariate Gaussian distribution with a mean and variance. We can see in the figure below the VAE model:

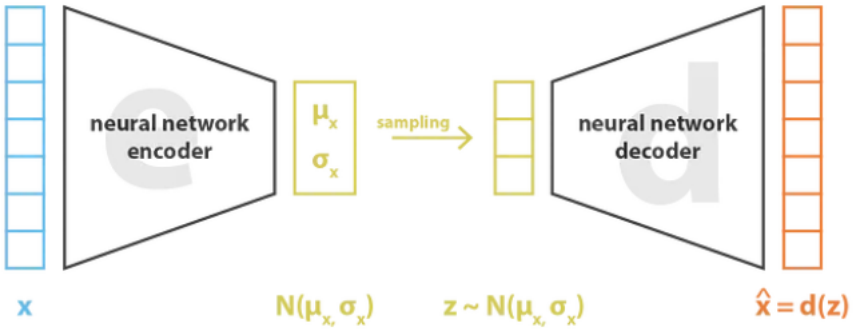

Figure 2.13: Variational auto-encoder and its loss where x is the input and d(z) is the output.

2.5.11.1 Loss function

The loss function of a variational auto-encoder has one additional term, so it contains two terms:

$$Loss = ||\hat{x} - x||^2 + KL[N(\mu_x, \sigma_x), N(0, l)] \tag{2.9}$$

- Reconstruction Loss: The reconstruction loss encourages the VAE to generate outputs that closely resemble the original input. It measures the discrepancy between the reconstructed output and the original input. In VAEs, the reconstruction loss is typically based on the concept of maximizing the likelihood of generating the original input given the learned latent representation. This is achieved by assuming that the reconstruction follows a probabilistic distribution.

- Regularization Term: The regularization term, also known as the Kullback-Leibler (KL) divergence, is used to ensure that the learned latent representation follows a specific prior distribution, typically a standard Gaussian distribution. The regularization term encourages the VAE to learn a latent space that is smooth and continuous, making it easier to generate new samples by sampling from this distribution.

2.6 Metrics for evaluating image compression quality

In this section, we will explore various metrics utilized for assessing the quality of image compression and employing them as loss functions during training. It is crucial to emphasize that apart from the metrics we will discuss, there are additional information theory-based measurements such as Shannon's entropy and KL divergence that can be referenced in [5].

2.6.1 Bits per pixel

The term "bits per pixel" (**bpp**) is a measure of the amount of bits used to represent each pixel in a digital image, its calculated as:

$$bpp = \frac{N_{bits}}{N_{pixels}} \tag{2.10}$$

In image compression, reducing the number of bits per pixel can significantly reduce the size of the image file, making it easier to transmit and store. However, reducing the bpp too much can result in a loss of image quality and detail, as the image is represented with less precision.

The appropriate bpp value for image compression depends on the specific application and the desired level of image quality. Different image compression techniques, such as JPEG or PNG, use different bpp values and algorithms to achieve the desired level of compression while preserving image quality as much as possible.

2.6.2 MS-SSIM

MS-SSIM (Multi-Scale Structural Similarity Index) is a widely used objective image quality assessment method that is based on measuring the structural similarity between two images. It is an extension of the original SSIM index [16], which is a full reference metric that compares the perceived image quality with the original or reference image.

The MS-SSIM index is calculated by comparing the luminance, contrast, and structure of two images at different scales. It uses a multi-scale decomposition of the images into a set of structural information maps, which are then compared to generate the final similarity index. The equation for MS-SSIM can be written as:

$$MSSSIM(x,y) = \left[\prod_{i=1}^{N} L_i^{\alpha_i}(x,y) \times C_i^{\beta_i}(x,y) \times S_i^{\gamma_i}(x,y)\right]^{\theta} \tag{2.11}$$

where:

x and y: are the two images being compared.
N: the number of scales used in the decomposition.
$l(i)$, $c(i)$, and $s(i)$: are the luminance, contrast, and structure similarity measures at scale i, respectively.
α_i, β_i, and γ_i: are the weighting factors for the luminance, contrast, and structure measures at scale i, respectively.
θ: a constant that controls the overall sensitivity of the index.

The MS-SSIM index ranges from 0 to 1, where 1 indicates perfect similarity between the two images.[17]

2.6.3 Peak to noise ration PSNR

Peak Signal-to-Noise Ratio (PSNR) is a commonly used metric to measure the quality of image compression algorithms [18]. It quantifies the difference between the original image and the compressed image by evaluating the signal-to-noise ratio in decibels. PSNR is calculated using the following formula:

$$PSNR = 20.\log_{1\,0}(MAX) - 10.\log_{1\,0}(MSE) \tag{2.12}$$

where:
- MAX is the maximum pixel value of the image (e.g., 255 for an 8-bit image).
- MSE (Mean Squared Error) is the average squared difference between the original and compressed images.

A higher PSNR value indicates a smaller difference between the original and compressed images, and thus, a higher quality compression. Typically, a higher PSNR is desired, indicating less loss of image quality during compression. However, it's worth noting that PSNR is just one metric to evaluate image compression quality. Other metrics, such as Structural Similarity Index (SSIM) or Perceptual Quality Index (PQI), may provide a more comprehensive assessment of perceptual image quality, taking into account factors like human visual perception.

2.7 Conclusion

In conclusion, this opening chapter of the thesis embarks on an extensive exploration of preliminary concepts essential for a comprehensive understanding of this thesis. The chapter begins by establishing a solid foundation in classic image compression principles. It then delves into the background of neural networks, discussing topics such as variational auto-encoders. The chapter also investigates various metrics commonly used in image processing, with a particular focus on the Multi Scale Structural Similarity Index (MS-SSIM), a crucial tool for evaluating image quality and codec performance. Through a thorough examination of these metrics and AI concepts, the chapter aims to provide a nuanced overview of the intricate mechanisms and quantitative measures employed in the domain of image compression, empowering readers to develop a nuanced understanding of the subject matter. Within the forthcoming chapter entitled "AI-based image compression", we shall investigate an array of methodologies that seamlessly align with our predefined objectives.

Chapter 3

AI-based image compression techniques

3.1 Introduction

In the upcoming chapter, we will delve into an exploration of some methodologies for image compression using deep learning. These methodologies can be classified into two distinct categories: the first focuses on achieving optimal results in terms of reconstruction and the compression ratio while maintaining quality, besides the second category that dedicates the latent representation to computer vision tasks, particularly face recognition, which we consider as our secondary objective. Throughout those investigations, they ensured that the aspect of reconstruction is given due consideration, underscoring its significance in their analysis.

3.2 State of the art diagram

Since in our work we are trying to unite the computer vision and image compression we divided the state of the art into two part, then later we will use the necessary part of each of the following models, we can summerize those techniques in the following figure:

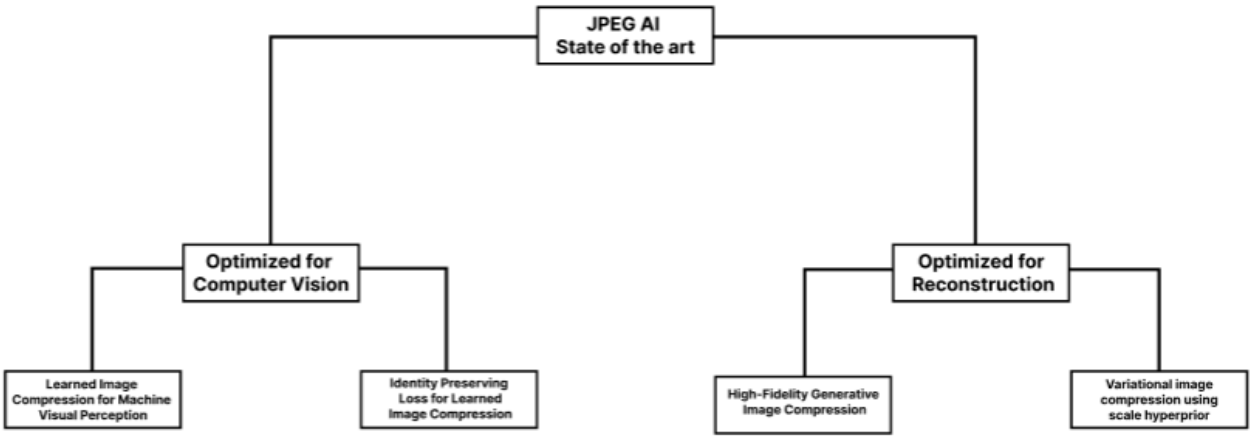

Figure 3.1: State of the art diagram of JPEG AI compressors.

3.3 Variational image compression using scale hyperprior

3.3.1 Introduction

Ballé et al. [19] present a comprehensive approach to image compression using variational auto-encoders, incorporating a hyperprior to effectively capture spatial dependencies within the latent representation. This hyperprior leverages side information, a concept widely used in modern image codecs but largely unexplored in image compression with artificial neural net-

works (ANNs). In contrast to existing auto-encoder compression methods, their model simultaneously trains a sophisticated prior alongside the underlying auto-encoder. Their experiments demonstrate that this model achieves state-of-the-art image compression results in terms of visual quality, as measured by the popular MS-SSIM index. Additionally, it outperforms previously published ANN-based methods in rate-distortion performance when evaluated using the more traditional metric based on squared error (PSNR). They also provide a qualitative comparison of models trained for different distortion metrics.

3.3.2 Compression with variational models

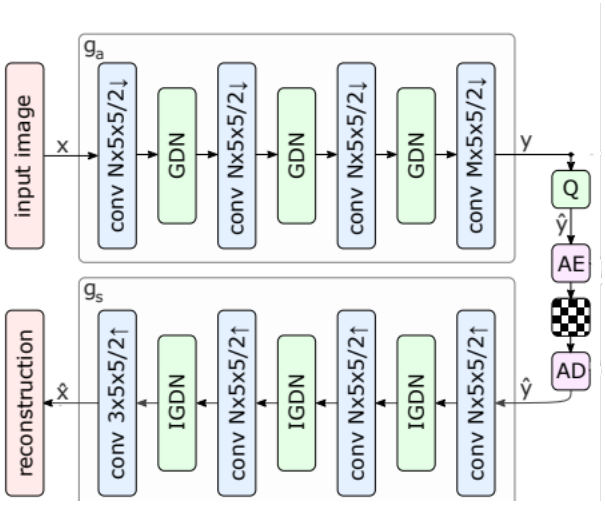

Figure 3.2: Transform coding using ANNs.

In the transform coding approach (shown in figure 3.2) to image compression described in [20] the process involves an encoder and a decoder. The encoder takes an image vector, denoted as x, and applies a parametric analysis transform :

$$g_a(x; \phi_g)$$

With the parameters ϕ_g . This transform converts the image into a latent representation, denoted as y. The resulting y is then quantized to produce $\hat{y}$. Quantization involves approximating the continuous values of y with a limited number of discrete values. Since $\hat{y}$ is now a discrete-valued representation, it can be compressed without any loss of information using entropy coding techniques like arithmetic coding in [21]. Entropy coding assigns shorter codes to more frequently occurring values, resulting in an efficient representation of the data. The compressed representation is then transmitted as a sequence of bits. On the decoder side, the compressed signal is received and processed to recover $\hat{y}$. The decoder applies a parametric synthesis transform, $g_s(\hat{y}; \theta_g)$, with the parameter θ_g, to reconstruct the image vector $\hat{x}$. The

synthesis transform essentially reverses the analysis transform, allowing the decoder to recreate an approximation of the original image.

It's important to note that quantization, the process of mapping continuous values to discrete levels, introduces some error in the compressed representation. This error is tolerated in lossy compression, where a certain amount of quality loss is acceptable in exchange for achieving higher compression ratios. The trade-off between bit rate of the compressed representation and the distortion error introduced by quantization gives rise to a rate–distortion optimization problem. The optimization problem is formally expressed as a variational auto-encoder, introduced by Kingma and Welling in [15]. This approach combines a probabilistic generative model of the image with an approximate inference model, The generative model is associated with the synthesis transform, which reconstructs an image from its latent representation. On the other hand, the inference model is linked to the analysis transform, which infers the latent representation from the source image. In the context of variational inference, the objective is to approximate the intractable true posterior distribution $p_{\tilde{y}|x}(\tilde{y}|x)$ with a parameterized variational density $q(\tilde{q}|x)$ by minimizing the expectation of their Kullback–Leibler (KL) divergence over the data distribution p_x:

$$\mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}}} D_{\mathrm{KL}}\left[q \| p_{\tilde{\boldsymbol{y}}|\boldsymbol{x}}\right] = \mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}}} \mathbb{E}_{\tilde{\boldsymbol{y}}\sim q}\left[\log q(\tilde{\boldsymbol{y}} \mid \boldsymbol{x}) - \underbrace{\log p_{\boldsymbol{x}|\tilde{\boldsymbol{y}}}(\boldsymbol{x} \mid \tilde{\boldsymbol{y}})}_{\text{weighted distortion}} - \underbrace{-\log p_{\tilde{\boldsymbol{y}}}(\tilde{\boldsymbol{y}})}_{\text{rate}}] + const.\right] \quad (3.1)$$

3.3.3 Introduction of a scale hyperprior

Figure 3.3: Left: an image from the Kodak dataset Right: visualization of a subset of the latent representation y of that image.

The figure figure 3.3 clearly illustrates the presence of substantial spatial relationships among the elements of $\hat{y}$. Remarkably, their magnitudes seem to be interconnected in a spatial manner. To address dependencies among a group of target variables, a commonly employed approach is to introduce latent variables. These latent variables are conditioned upon to assume the

independence of the target variables in [22]. In the proposal, they augment the model by incorporating an extra set of random variables, denoted as $\tilde{z}$, to capture the spatial dependencies. The extended model is depicted in the following figure:

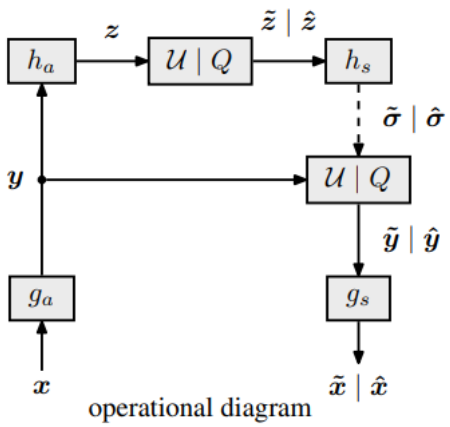

Figure 3.4: Compression model with extended hyperprior.

The model represents each element, $\hat{y}_i$, as a Gaussian distribution with a mean of zero and a unique standard deviation,σ_i. These standard deviations are determined by applying a parametric transformation, h_s to the variable $\tilde{z}$. Similar to before, they convolve each Gaussian density with a standard uniform distribution:

$$p_{\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{z}}}\left(\tilde{\boldsymbol{y}} \mid \tilde{\boldsymbol{z}}, \boldsymbol{\theta}_h\right) = \prod_i \left(\mathcal{N}\left(0, \tilde{\sigma}_i^2\right) * \mathcal{U}\left(-\frac{1}{2}, \frac{1}{2}\right)\right)\left(\tilde{y}_i\right)$$

$$\text{with} \quad = \mathrm{h}_s\left(\tilde{\boldsymbol{z}}; \boldsymbol{\theta}_h\right) \quad (3.2)$$

To expand the inference model, they incorporate an additional parametric transformation, h_a, onto the existing output, y. This combination creates a unified and factorized variational posterior. In simpler terms, they enhance the model by adding h_a to y, resulting in a joint and factorized variational posterior, as follows:

$$q\left(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}} \mid \boldsymbol{x}, \boldsymbol{\phi}_g, \boldsymbol{\phi}_h\right) = \prod_i \mathcal{U}\left(\tilde{y}_i \mid y_i - \frac{1}{2}, y_i + \frac{1}{2}\right) \cdot \prod_j \mathcal{U}\left(\tilde{z}_j \mid z_j - \frac{1}{2}, z_j + \frac{1}{2}\right)$$
$$\text{with } \boldsymbol{y} = g_a\left(\boldsymbol{x}; \boldsymbol{\phi}_g\right), \boldsymbol{z} = h_a\left(\boldsymbol{y}; \boldsymbol{\phi}_h\right). \quad (3.3)$$

This follows the intuition that the responses y should be sufficient to estimate the spatial distribution of the standard deviations. As they have no prior beliefs about the hyperprior,

they model $\tilde{z}$ using the non-parametric, fully factorized density model previously used for y:

$$p_{\tilde{\boldsymbol{z}}|\boldsymbol{\psi}}(\tilde{\boldsymbol{z}} \mid \boldsymbol{\psi}) = \prod_i \left(p_{z_i|\boldsymbol{\psi}^{(i)}} \left(\boldsymbol{\psi}^{(i)} \right) * \mathcal{U}\left(-\frac{1}{2}, \frac{1}{2} \right) \right) (\tilde{z}_i) \tag{3.4}$$

The loss function of this model works out to be:

$$\mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}}} D_{\mathrm{KL}} \left[q \| p_{\tilde{\boldsymbol{y}},\tilde{\boldsymbol{z}}|\boldsymbol{x}} \right] = \mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}}} \mathbb{E}_{\tilde{\boldsymbol{y}},\tilde{\boldsymbol{z}}\sim q} \left[\log q(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}} \mid \boldsymbol{x}) - \log p_{\boldsymbol{x}|\tilde{\boldsymbol{y}}}(\boldsymbol{x} \mid \tilde{\boldsymbol{y}}) - \log p_{\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{z}}}(\tilde{\boldsymbol{y}} \mid \tilde{\boldsymbol{z}}) - \log p_{\boldsymbol{z}}(\tilde{\boldsymbol{z}}) \right]. \\ + \text{const.} \tag{3.5}$$

The overall architecture of the model will be:

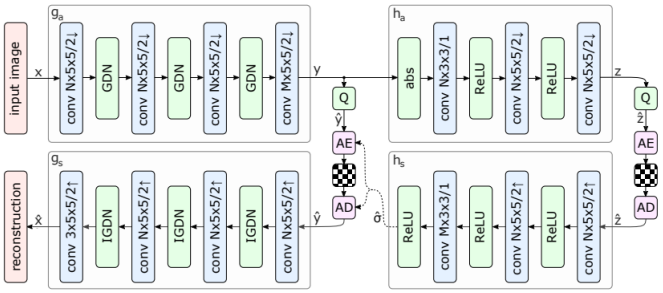

Figure 3.5: Network architecture of the hyperprior model.

3.4 High-Fidelity Generative Image Compression

3.4.1 Introduction

Mentzer et al, conducted an extensive study on combining Generative Adversarial Networks (GANs) and learned compression techniques to develop an advanced generative lossy compression system [23]. Their investigation focused on various aspects, including normalization layers, generator and discriminator architectures, training strategies, and perceptual losses. Unlike previous works, their approach achieved visually appealing reconstructions that closely resembled the input, operated effectively at different bitrates, and could be applied to high-resolution images.

To bridge the gap between rate-distortion-perception theory and practical implementation, the researchers evaluated their approach through quantitative analysis using different perceptual

metrics and a user study. The results of the study demonstrated that their method outperformed previous approaches even when those approaches utilized more than twice the bitrate. In summary, the researchers successfully developed a state-of-the-art generative lossy compression system that achieved visually pleasing reconstructions, worked across various bitrates, and surpassed previous methods in terms of user preference.

3.4.2 Losses

Lets the loss of neuronal image compression which is modeled with an auto-encoder consisting of an encoder E and a decoder G be:

$$\mathcal{L}_{EG} = \mathbb{E}_{x\sim p_X} \left[\lambda r(y) + d\left(x, x'\right)\right] \tag{3.6}$$

where:
- $r(y)$ is the rate error: $r(y) = -log(P(y))$.
- $d(x, x')$ is the distortion error in this case the SSIM MSSIM.

For the Conditional GAN loss the objective is to create a generator G that takes into account a condition s and transforms samples y from a predetermined distribution p_Y to $pX|S$. Additionally, a discriminator D is used to determine the probability of an input (x, s) being a sample from $pX|S$ or from the output of G. The aim is for G to deceive D by generating samples that appear real, meaning they are from $pX|S$. By keeping s constant, they can optimize the "non-saturating" loss:

$$\mathcal{L}_G = \mathbb{E}_{y\sim p_Y}[-\log(D(G(y, s), s)]$$

$$\mathrm{L}_D = \mathbb{E}_{y\sim p_Y}\Big[-\log(1 - D(G(y, s), s)] + \mathbb{E}_{x\sim p_{X|s}}[-\log(D(x, s))] \tag{3.7}$$

3.4.3 Formulation and optimization

The augmentation of a neural image compression formulation using a conditional GAN combines various equations and learns networks E, G, P, and D. The process involves using $y = E(x)$ and $s = y$. In addition, a "perceptual distortion" d_P is incorporated, which

is inspired by the work of [24] and employs a VGG-based loss (LPIPS) [25] for improved training. d_P is considered a distortion and is combined with MSE to create the overall distortion$d = kMMSE + k_P d_P$, where k_M and k_P are hyper-parameters. By adjusting hyper-parameters and , the trade-off between the different terms can be controlled, resulting in the final outcome:

$$\begin{aligned} \mathcal{L}_{EGP} &= \mathbb{E}_{x\sim p_X}\left[\lambda r(y) + d\left(x, x'\right) - \beta \log\left(D\left(x', y\right)\right)\right], \\ \mathcal{L}_D &= \mathbb{E}_{x\sim p_X}\left[-\log\left(1 - D\left(x', y\right)\right)\right] + \mathbb{E}_{x\sim p_X}[-\log(D(x, y))]. \end{aligned} \tag{3.8}$$

3.4.4 Architecture

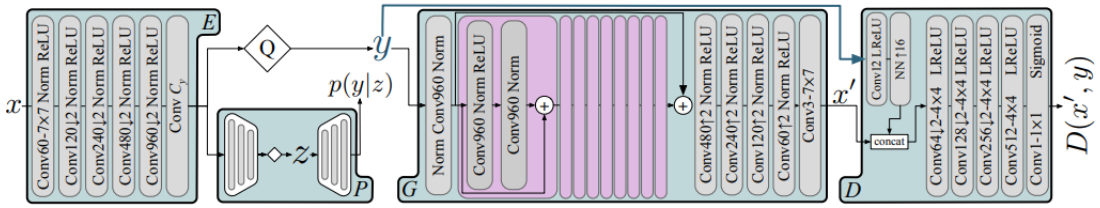

Figure 3.6: HiFi Architecture where E: is the encoder to compress input image. P: is the probability model trained to calculate entropy. G: Conditional generator to decompress y. D: Descriminator to be trained mutually with G.

The architecture shown in figure 3.6 consists of four components: encoder E, generator G, discriminator D, and probability model P. For the probability model P, we utilize a hyper-prior model proposed in a previous work [19]. This model incorporates side information z to represent the distribution of y and involves simulating quantization using uniform noise $U(-1/2, 1/2)$ in the hyper-encoder and when estimating $p(y|z)$. However, when y is fed to the generator G, rounding is used instead of noise, ensuring that G encounters the same quantization noise during both training and inference. The components E, G, and D are based on other references, although there are some notable differences in the discriminator and normalization layers.

3.5 Learned Image Compression for Machine Visual Perception

3.5.1 Introduction

The main objectives of this thesis [26] were threefold. Firstly, the aim has been to develop an advanced compression algorithm using neural networks that outperforms JPEG when evaluated against standard image compression metrics. This algorithm should generate a compressible representation. Secondly, the goal has been to investigate whether it is feasible to influence this compressible representation in order to improve performance in downstream tasks such as image classification, object detection, and semantic segmentation. Lastly, the objective has been to determine whether this compressible representation provides an advantage in densely labelled tasks, specifically semantic segmentation, as well as in the few-shot setting. Their hypothesis suggests that it does, as the compressible representation is expected to encapsulate all relevant image information within a smaller data structure, facilitating the extraction of useful information for downstream tasks.

3.5.2 Rate-Distortion-Utility perspective

The representation z has various applications, including encoding it into a bit stream for storage or transmission, as well as direct utilization for visual tasks without reconstructing the input image. This expands the conventional rate-distortion trade-off and raises the question of the representation's utility for machine perception tasks, rather than solely focusing on perceptual quality compared to the original image. They put forward the hypothesis that for any desired rate (R) and distortion level (D), there are multiple quantized latent representations:

$$\hat{z} = Q(f(x)) \tag{3.9}$$

that exhibit significant performance differences in subsequent machine vision tasks. In other words, different representations can achieve the same rate-distortion trade-off, but some may be more suitable for specific semantic tasks than others. In light of this, They propose that the optimization objective function should not only consider the classical rate-distortion paradigm but also incorporate a measure of the representation's usefulness for concrete machine perception tasks:

$$\mathcal{R}(\mathbf{Q}(f(x)), x) + \lambda_d \cdot \mathcal{D}(g(\mathbf{Q}(f(x))), x) + \lambda_u \cdot \mathcal{U}(\mathbf{Q}(f(x))) \tag{3.10}$$

The utility function $\mathcal{U}()$ represents a pragmatically defined metric for machine perceptual quality. Unlike traditional metrics such as squared error that characterize human perceptual quality, machine vision perceptual quality can be defined more pragmatically by considering a multi-task loss:

$$\mathcal{U}(\mathbf{Q}(f(x))) = \sum_{t \in \mathcal{T}} \lambda_t \left[L_t \left(h_t(f(x)), y_t \right) \right] \tag{3.11}$$

Where L_t denote a loss function specific to a particular task t, which measures its performance based on labels y_t. The functions h_t represent task-specific decoders. They hypothesize that optimizing learned compression models for rate R, distortion D, and utility U, will yield encoders $f(x)$ that can generate quantized representations $\hat{z}$ with low bit-per-pixel values.

3.5.3 Architecture

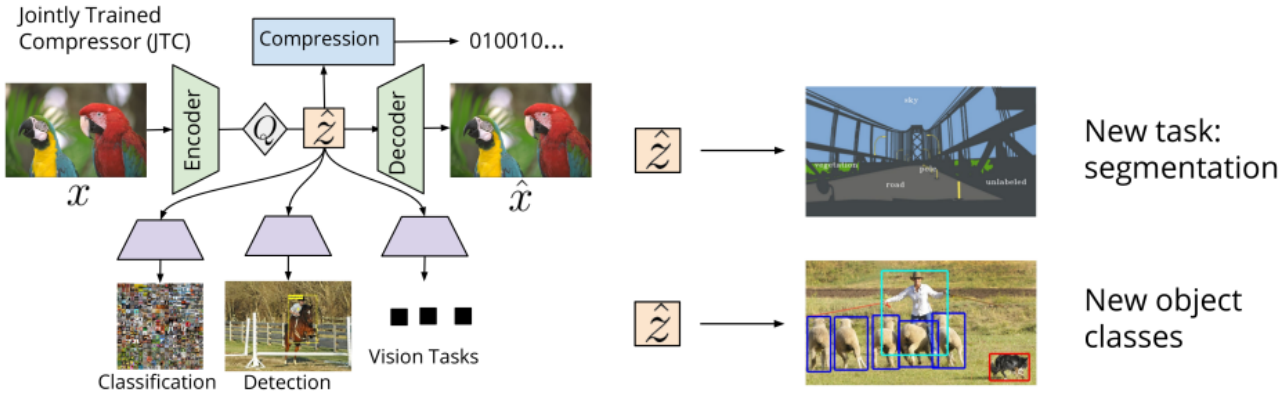

Figure 3.7: Learned Image Compression for Machine Visual Perception model. Right: Global model containing Encoder and Decoder and ANNs for different tasks. Left: Different tasked trained in this methode.

Their strategy for learning compressible image representations for machine perception that can be seen in figure 3.7 can be summarized as follows. They adopt a holistic approach by training a model to simultaneously compress images and perform essential vision tasks using the compressed representations. This integrated training methodology results in notable improvements compared to conventional methods like JPEG compression or simplistic learned deep

compression. Notably, when training new models for specific tasks directly from their learned compressible representations, their approach demonstrates significant advantages.

3.6 Identity Preserving Loss for Learned Image Compression

3.6.1 Introduction

This study introduces an end-to-end image compression framework that learns specific features related to the domain, enabling higher compression ratios compared to standard HEVC/JPEG techniques while preserving accuracy in downstream tasks such as recognition. Their framework eliminates the need for fine-tuning the downstream task, allowing them to use any off-the-shelf model without retraining. For their research, they focus on faces as the application domain due to the abundance of available datasets and readily accessible recognition models. They propose a novel loss function called Identity Preserving Reconstruction (IPR), which achieves Bits-Per-Pixel (BPP) values approximately 38% and 42% of the CRF-23 HEVC compression for low-resolution (LFW) and high-resolution (CelebA-HQ) datasets, respectively. This compression ratio is achieved by teaching the model to retain domain-specific features (e.g., facial features) while sacrificing background details. Additionally, their proposed compression model produces reconstructed images that remain robust to changes in downstream model architectures. They demonstrate comparable recognition performance on the LFW dataset using an unseen recognition model while maintaining a lower BPP value of around 38% of CRF-23 HEVC compression.

3.6.2 Framewrork

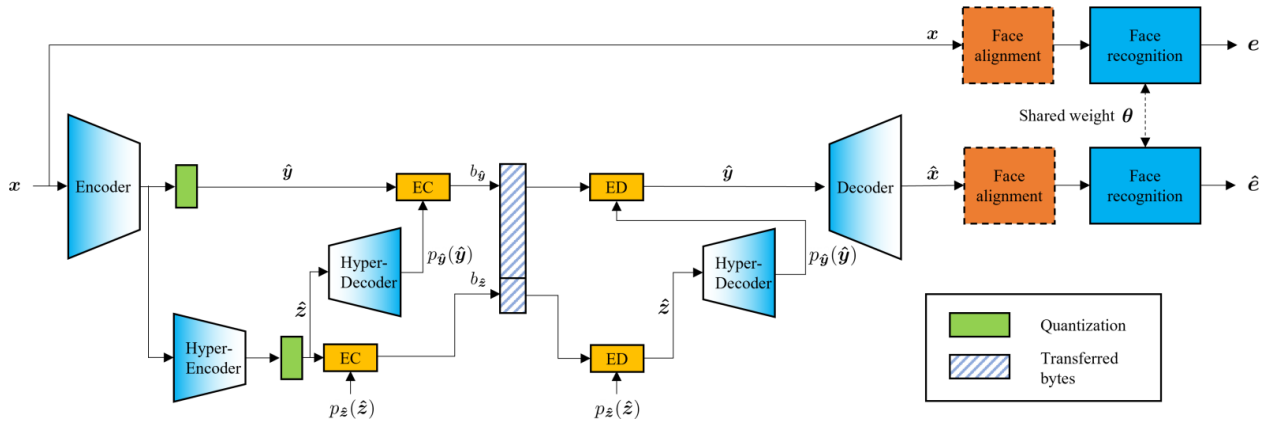

Figure 3.8: Proposed framework containing the Codec's architecture on the left and the face recognition pretrained model on the right.

They present the proposed framework (in figure 3.8). Initially, the input images (x) undergo encoding and quantization, resulting in a discrete latent representation ($\hat{y}$). To determine the entropy model, we adopt the hyper-prior method [19]. The hyper encoder-decoder generates a discrete hyper-latent representation$\hat{z}$ and predicts the distribution of $\hat{y}$ using the entropy model $p_{\hat{y}(\hat{y})}$. To achieve lossless compression, they employ an entropy encoder (EC) and an entropy decoder (ED). The EC utilizes $p_{\hat{y}(\hat{y})}$ to encode $\hat{y}$ into transferable bytes $b_{\hat{y}}$, while a separate entropy model$p_{\hat{z}(\hat{z})}$ with a factorized prior distribution encodes $\hat{z}$ into $\_\hat{z}$. The combination of $\_\hat{y}$ and $\_\hat{z}$ forms the payload that requires transmission over the network. Upon transmission, the ED converts these bytes back into discrete latent and hyper-latent representations. Finally, the decoder reconstructs the images (x). Traditional compression methods optimize both reconstruction loss to maintain image quality and bit-rate loss to reduce file size where:

$$\mathcal{L}_{\mathrm{R}} = \mathcal{L}_{\mathrm{rec}} + \lambda_{\mathrm{rate}}\, \mathcal{L}_{\mathrm{rate}} \tag{3.12}$$

$$\mathcal{L}_{\mathrm{rec}} = \mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}}}[d(\boldsymbol{x}, \hat{\boldsymbol{x}})] \tag{3.13}$$

$$\mathcal{L}_{\mathrm{rate}} = \mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}}}\left[-\log\left(p_{\hat{\boldsymbol{y}}}(\hat{\boldsymbol{y}})\right) - \log\left(p_{\hat{\boldsymbol{z}}}(\hat{\boldsymbol{z}})\right)\right] \tag{3.14}$$

where $\mathcal{L}_{\mathrm{rec}}$ is the reconstruction loss, $\mathcal{L}_{\mathrm{rate}}$ is the bit-rate loss, $d(\boldsymbol{x}, \hat{\boldsymbol{x}})$ is the distortion function, and λ_{rate} is the weight for bit-rate loss.the total loss function $\mathcal{L}_{\mathrm{R}}$ is for Reconstruction-only loss. When selecting the MSE function as the reconstruction loss, $d(\boldsymbol{x}, \hat{\boldsymbol{x}})$ is defined as:

$$d(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2 \tag{3.15}$$

3.6.2.1 Identity Preserving Reconstruction Loss

The objective is to enhance the compression ratio while maintaining the recognition performance. To achieve this, they employ the CurricularFace [27] pre-trained model. However, they refrain from fine-tuning the model weights and instead keep them frozen. To ensure compatibility between the compression model and the recognition model, they develop the Identity Preserving (IP) loss function. This loss function incorporates both the bit-rate loss and identity loss, aiming to minimize the discrepancy between e and $\hat{e}$ (as depicted in figure 3.8) for identity recognition while achieving a smaller file size. where:

$$\mathcal{L}_{\text{IP}} = \lambda_{\text{rate}} \mathcal{L}_{\text{rate}} + \lambda_{\text{id}} \mathcal{L}_{\text{id}} \tag{3.16}$$

$$\mathcal{L}_{\text{id}} = \mathbb{E}_{\boldsymbol{x} \sim p_{\boldsymbol{x}}} \left[1 - \frac{\boldsymbol{e} \cdot \hat{\boldsymbol{e}}}{\|\boldsymbol{e}\| \|\hat{\boldsymbol{e}}\|} \right] \tag{3.17}$$

where $\mathcal{L}_{\text{id}}$ is the identity loss. Since $\mathcal{L}_{\text{IP}}$ only optimizes recognition performance, they further derive Identity Preserving Reconstruction (IPR) loss function to jointly optimize image quality and recognition performance:

$$\mathcal{L}_{\text{IPR}} = \mathcal{L}_{\text{rec}} + \lambda_{\text{rate}} \mathcal{L}_{\text{rate}} + \lambda_{\text{id}} \mathcal{L}_{\text{id}} \tag{3.18}$$

Using face alignment module A and a face recognition model R with shared weights θ, we derive the embeddings by analyzing both decompressed images and their corresponding original images, where:

$$\boldsymbol{e} = R_{\boldsymbol{\theta}}(A(\boldsymbol{x})) \quad \text{and} \quad \hat{\boldsymbol{e}} = R_{\boldsymbol{\theta}}(A(\hat{\boldsymbol{x}})) \tag{3.19}$$

To conclude, in this paper, the authors integrated three loss functions. The first loss function is dedicated to reconstruction, aiming to ensure accurate image restoration. The second loss function focuses on minimizing entropy to reduce the bits per pixel (bpp), thereby enhancing compression efficiency. Lastly, the third term of the loss function is designed to preserve facial features and maintain identity representation in the compressed images.

3.7 Conclusion

In this chapter, we examined the current methods for neural image compression, focusing on two specific approaches: [23] and [19] that use the scale hyper prior method and the trained probability model in order to minimize the latent representation while applying lossless encodings. These methods aim to optimize compression while preserving a high-quality reconstruction. Additionally, we explored two other techniques that trained their codecs for face recognition, image detection, and segmentation. One of these techniques directly utilized a latent representation, while the other trained the reconstruction process to maintain the same facial feature vector. In the upcoming chapter, we will delve into a detailed analysis of how latent representation varies under different training criteria while aiming to achieve accurate classification using the architecture of [23].

Chapter 4

Making use of the latent representation for image classification

4.1 Introduction / Motivation

The motivation behind choosing a computer vision task is to explore the potential of compressed representations in classification problems beyond their traditional use for reconstruction purposes. By investigating the compressed representation, we aimed to understand the extent to which it preserves valuable information that can be harnessed for effective classification. Our objective for this part is to analyze the transformation of this representation across two extreme ends of the spectrum. While examining the compressed representation at these extremes, we sought to gain insights into how it evolves and adapts to different requirements, and whether it retains useful information that can aid in computer vision tasks. This investigation would help us assess the overall utility and robustness of compressed representations in the context of computer vision tasks, potentially unlocking new possibilities for efficient and effective computer vision systems.

Further more, we noticed that all the work done before always train the decoder and the encoder simultaneously. So in order to implement JPEG AI on a larger scale its more adequate to have a fixed decoder on different devices. So for the following we took the the decoder in [23] and fixed its weights than we train the encoder with the same architecture for different losses which will be discussed in the Framework section.

4.2 Experimental conditions

4.2.1 Material resources

During the training process we used 3 GPUs (**one** RTX 3070 8 Go and **two** RTX 2080Ti 12Go) as main resources some training were done on the first one and others in the two 2080s.

4.2.1.1 GeForce RTX 3070

The NVIDIA GeForce RTX 3070 with 8GB of VRAM is a graphics processing unit (GPU) that offers excellent performance for machine learning trainings. With its Ampere architecture and advanced tensor cores, the RTX 3070 delivers good computational power and accelerated AI capabilities. The 8GB GDDR6 memory allows for efficient data handling and faster model training times. The card boasts a boost clock speed of up to 1.73 GHz, enabling quick processing of complex machine learning algorithms. It supports hardware-accelerated ray tracing, which enhances visualizations and simulations, providing realistic and immersive experiences. The RTX 3070 is equipped with 5888 CUDA cores, ensuring high parallel processing capabilities, which is crucial for training deep learning models. Additionally, the card supports NVIDIA's CUDA and cuDNN libraries, as well as popular machine learning frameworks like TensorFlow and PyTorch, making it compatible with a wide range of software tools. Overall, the RTX 3070 8GB is a robust option for machine learning trainings, offering exceptional performance, memory capacity, and compatibility with popular frameworks and libraries.

4.2.1.2 GeForce RTX2080Ti

The NVIDIA RTX 2080 Ti 12GB is a GPU designed for high-performance computing tasks such as training AI models. It features a 12GB GDDR6 video memory, which provides ample space for storing large datasets and model parameters. With its 4352 CUDA cores, the GPU delivers powerful parallel processing capabilities, enabling efficient training and inference operations. The RTX 2080 Ti also supports real-time ray tracing, allowing AI models to generate more realistic graphics and visualizations. Its memory bandwidth of 616 GB/s facilitates fast data transfers and computations, contributing to quicker training times for AI algorithms. Furthermore, the GPU supports multiple monitors and offers various connectivity options for seamless integration into AI development workflows. Overall, the NVIDIA RTX 2080 Ti 12GB provides the computational power and memory capacity necessary for training sophisticated AI models.

4.2.1.3 AMD Ryzen 5 5600H

The AMD Ryzen 5 5600H is a laptop processor from AMD's Ryzen 5000 series, based on the Zen 3 architecture. Here are its specifications:

- CPU Cores and Threads: The Ryzen 5 5600H has 6 CPU cores and 12 threads. This means it can handle up to 6 simultaneous tasks with the support of multi-threading technology.
- Base Clock Speed: The base clock speed of the Ryzen 5 5600H is 3.3 GHz. This is the frequency at which the processor operates under normal conditions.
- Max Boost Clock Speed: The processor can dynamically increase its clock speed for better performance when needed. The Ryzen 5 5600H can reach a maximum boost clock speed of 4.2 GHz.
- Cache: It features a total of 16MB of L3 cache, which helps improve the processor's efficiency by providing faster access to frequently used data.
- Memory Support: The processor supports DDR4 memory with a maximum speed of 3200 MHz. The specific memory configuration and maximum capacity may vary depending on the laptop's design and the manufacturer's specifications.
- PCIe Version: The Ryzen 5 5600H supports PCIe Gen 3.0, which allows for high-speed data transfer to compatible devices such as NVMe solid-state drives and dedicated graphics cards.

4.2.2 Dataset

4.2.2.1 Cifar-10

CIFAR-10 [28] is a widely used benchmark dataset in the field of computer vision and machine learning. It consists of 60,000 color images, each measuring 32x32 pixels, divided into ten classes, with 6,000 images per class. The ten classes include common objects such as airplanes, automobiles, birds, cats, dogs, frogs, horses, ships, trucks, and a miscellaneous category. CIFAR-10 serves as a valuable resource for researchers and practitioners to develop and evaluate image classification algorithms. It has become a standard dataset for testing the performance and generalization capabilities of various deep learning models, providing a challenging task due to the small image size and the complexity of differentiating between similar-looking objects.

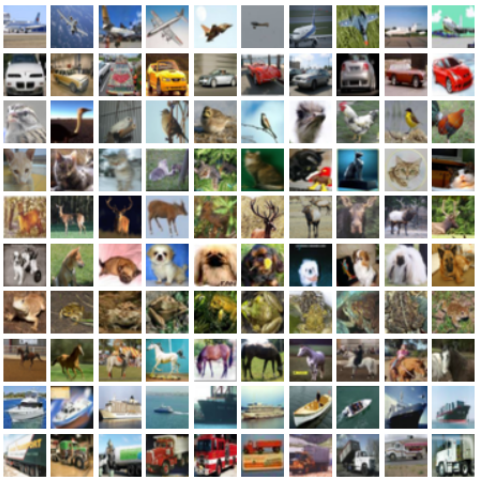

Figure 4.1: CIFAR-10 images.

With its broad applications and widespread usage, CIFAR-10 has played a significant role in advancing the field of computer vision. We can see some samples in figure 4.3

Classes:

AIrplane	automobile	horse	frog	deer
dog	bird	cat	ship	truck

Table 4.1: CIFAR-10 Classes.

4.2.2.2 LFW Labled faces in the wild

The Labeled Faces in the Wild (LFW) dataset [29] is a widely used benchmark in the field of face recognition. It is a comprehensive collection of facial images gathered from the internet, capturing a diverse range of individuals under various conditions. The dataset consists of over 13,000 labeled images, encompassing more than 5,000 unique individuals. Each image is aligned and cropped to focus on the face, ensuring consistency across the dataset. LFW has become a standard evaluation resource for face recognition algorithms, allowing researchers to compare and assess the performance of different methods. Its large size, diverse set of subjects, and real-world variations make it a valuable resource for advancing the field of face recognition and understanding the challenges associated with real-world face detection and identification tasks. We can visualize one of LFW classes in the following figure:

Figure 4.2: LFW face images of the same class.

4.3 Frameworks

In order to investigate how the latent representation varies when optimized on different objectives, we employed two specific protocols. These protocols were designed to explore and analyze the changes that occur within the latent representation based on the optimization objectives employed. The purpose is to gain insights into how different optimization goals affect the resulting latent representation. We used to protocols that are the following:

4.3.1 Framewrork 1: Encoder & Classifier joint training

4.3.1.1 Architecture

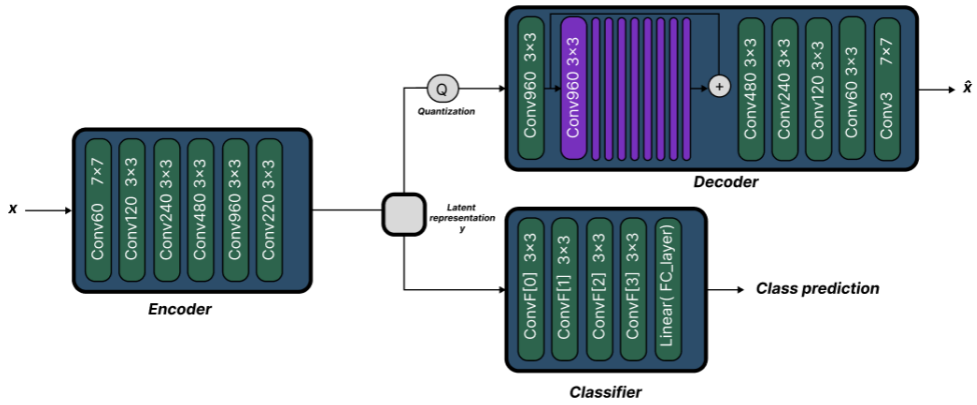

Figure 4.3: Protocol 1's architecture containing Encoder E on the left and Decoder/Classifier on the right.

For the current architecture seen in figure 4.3 we have our Encoder's (based on [23]) input x which is resized CIFAR-10 images, with the size of 64x64x3 (instead of 32x32x3). We will get the latent representation $y = E(x)$ with the size (220,4,4). This representation will be used for two tasks:

- **Reconstruction:** Before the reconstruction of the original the latent representation

needs to be quantified in order to limit the number of symbols to be coded later by the lossless encoding such as Huffman, and in order to optimize this process we will need to decrease the entropy of this source, but this will not be the focus of this thesis. After the quantization part the quantified latent representation will be introduced in the Decoder that contains convolution layers and residual blocks.

- **Classification** For the classification part we wont need the latent representation to be quantified, we will use it directly as an input to the classifier model which contains variable convolution layers and a fully connected layer which will be followed by a $Softmax$ function.

Encoder (Trainable)

- **Pre-padding**: The input data is padded using reflection padding with a width of 3 pixels on each side. Reflection padding means that the values on the boundary of the input are reflected to create the padding.

- **Asymmetric padding:** After the pre-padding, asymmetric padding is applied to the input. This padding adds an extra row of padding on the top and bottom and an extra column of padding on the left and right sides. Post-padding:

- **Post-padding:** After the convolutional layers, the output is again padded using reflection padding with a width of 1 pixel on each side.

- **Convolutional Block 1:** This block consists of a reflection padding layer, followed by a 2D convolutional layer with 3 input channels, 60 output channels, and a kernel size of 7x7. The output is then passed through a ChannelNorm2D layer and a ReLU activation function.

- **Convolutional Block 2:** This block also starts with a reflection padding layer, followed by a 2D convolutional layer with 60 input channels, 120 output channels, a kernel size of 3x3, and a stride of 2x2 (which downsamples the spatial dimensions by a factor of 2). The output is then passed through a ChannelNorm2D layer and a ReLU activation function.

- **Convolutional Block 3, 4, and 5:** These blocks follow a similar structure to Block 2 but with different input and output channel dimensions. Each block includes a reflection padding layer, a 2D convolutional layer, a ChannelNorm2D layer, and a ReLU activation function. The spatial dimensions are downsampled by a factor of 2 in each block.

- **Convolutional Block Output:** This block consists of a reflection padding layer followed by a 2D convolutional layer. It has 960 input channels and 220 output channels, with a kernel size of 3x3 and a stride of 1x1. This block provides the final output of the encoder.

Classifier (Trainable)

- **Convolutional Block 1:** It takes an input with 220 channels and applies 100 filters of size 2x2.

- **Batch Normalization layer:** It normalizes the output of the previous convolutional layer by subtracting the mean and dividing by the standard deviation computed over a batch of samples.

- Same layer are multiplied with different channels (as shown in the figure **3.3**). Untill the FNN to classify all the 10 classes.

Decoder (Frozen weights)

- **Initial convolutional block:** The first part of the generator network consists of a series of operations. It includes:
  Channel normalization (ChannelNorm2D): This operation normalizes the input channels of the feature map.
  Reflection padding: Another reflection padding is applied.
  Convolutional layer: A 2D convolution (Conv2d) is applied with a kernel size of 3x3 and a stride of 1x1, transforming the input from 220 channels to 960 channels.
  Channel normalization: The output channels are normalized again.

- **Residual blocks:** The core of the generator consists of multiple residual blocks (ResidualBlock). Each residual block performs the following operations:
  Reflection padding.
  Convolutional layer (conv1): A 2D convolution is applied with a kernel size of 3x3 and a stride of 1x1.
  Convolutional layer (conv2): Another 2D convolution is applied with the same parameters.
  Channel normalization (norm1 and norm2): Normalization is applied after each convolutional layer.
  The output of the second convolutional layer is added to the input of the residual block, creating a residual connection.

These residual blocks allow the network to learn residual or incremental changes, enabling it to preserve important features from the input image while introducing necessary modifications.

- **Upsampling blocks:** After the series of residual blocks, the generator includes a set of upsampling blocks. Each upsampling block performs the following operations: Convolutional transpose: This operation upsamples the feature map by a factor of 2 using a kernel size of 3x3, stride of 2x2, and padding of 1x1. The output channels decrease as the spatial dimensions increase.
  Channel normalization.
  Rectified Linear Unit (ReLU): This activation function introduces non-linearity into the network.
  These upsampling blocks help to restore the spatial resolution of the feature map while reducing the number of channels.

- Output convolutional block: Finally, the output convolutional block produces the desired output image. It includes:
  Reflection padding.
  Convolutional layer: A 2D convolution is applied with a kernel size of 7x7 and a stride of 1x1, converting the feature map from 60 channels to 3 channels, corresponding to the RGB color channels of the output image.

4.3.1.2 Loss function

As mentioned before the quality compression ratio will not be the main focus of this thesis so the loss function will not contain the entropy of the encoder source $r(y)$ where:

$$r(Y) = \mathbb{E}_{xp_x}[log(p(Y))] \tag{4.1}$$

So we will focus on the reconstruction part. For a first step we will use the Mean squarred error as a reconstruction metric to evaluate the distortion between the original image and its reconstruction:

$$D(x, \hat{x}) = \mathbb{E}[(x_i - \hat{x}_i)^2] \tag{4.2}$$

For the classification loss, since it is a multi-class problematic we will use the cross entropy, where:

$$C(o,c) = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{4.3}$$

So the global loss for this model will be:

$$L = D(x,\hat{x}) + \lambda C(o,c) = \mathbb{E}[(x_i - \hat{x}_i)^2] + \lambda \mathbb{E}[-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})] \tag{4.4}$$

Were λ varies according to the importance of the desired objective (whether giving importance to the reconstruction or to the classification).

4.3.1.3 Training

In this specific training process, the decoder's [23] weights are fixed. It means that the parameters of the decoder, which determine how it transforms the latent representation back into the reconstructed data, remain unchanged throughout the training. On the other hand, the encoder and the classifier are trained using a loss function that is mentioned in the preceding section. By minimizing the loss function, the model aims to improve its performance in terms of both reconstruction and classification.

For the training and validation splits, CIFAR-10 contains 50000 labeled images for training and 10000 images for testing. And it will be the same split that we will use.

We will also leave $\lambda = 1$ to give both importance to image reconstruction and image classification

Data augmentation

The main idea behind data augmentation is to introduce variations in the training data that are similar to the variations that the model is expected to encounter during actual deployment or inference. And we used it for this training since we faced an overfitting beforehand. here we can visualize how data augmentation work:

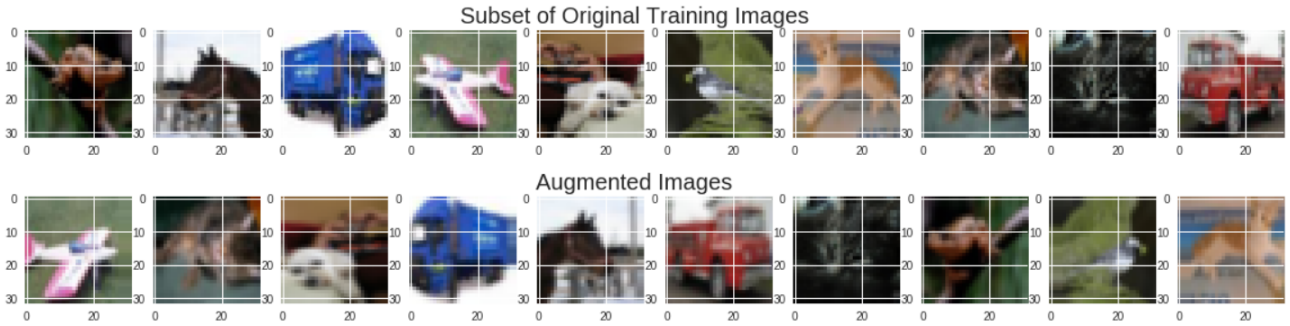

Figure 4.4: Data augmentation CIFAR-10.

Regularization: Weight decay

The purpose of weight decay is to control the complexity of the model by discouraging large weight values. When the weights are allowed to grow without bounds, it can lead to overfitting, where the model becomes too specialized to the training data and performs poorly on new, unseen data. And it is added during the training with a value equal to: **1e-05**.

Training

After conducting an extensive grid search to determine the optimal number of filters for each layer, we proceeded to carry out multiple training iterations using different filter configurations. Through this process, it has been revealed that the following configuration yielded the most favorable outcomes:

Operation	channels	activation
$conv2x2$	500	relu
$conv2x2$	420	relu
$conv2x2$	250	relu
$conv2x2$	170	relu
$FClayer$	10	softmax

Table 4.2: Architecture of the Classifier used in this frame work.

Prompting us to proceed with it. Consequently, we obtained the subsequent training results while incorporating a variable learning rate and SGD (stochastic gradient decent) optimizer in the following figure:

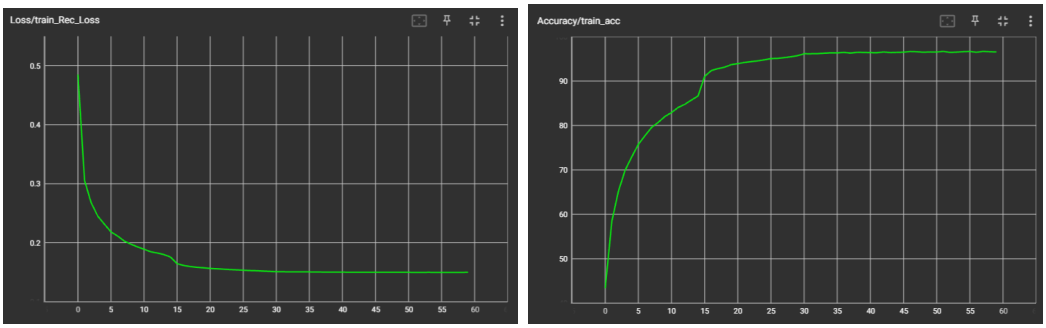

Figure 4.5: Left: Training loss by cross entropy. Right: Train accuracy. For 60 epochs.

The validation set plots are depicted in the figure presented below:

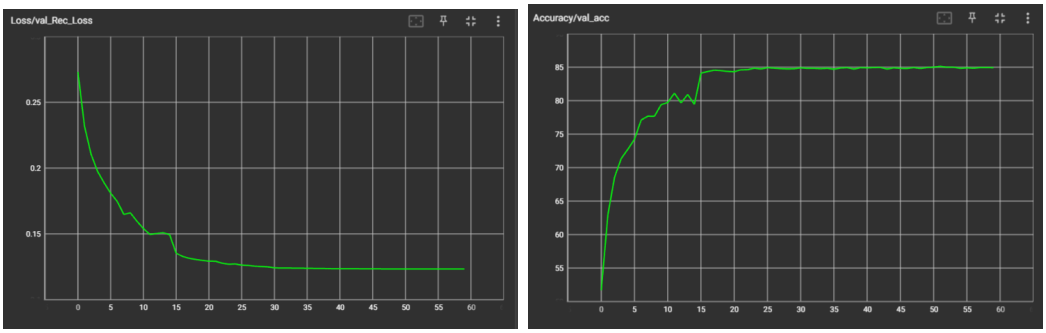

Figure 4.6: Left: Test loss by cross entropy. Right: Test accuracy.

4.3.1.4 Results

In the subsequent results provided below in table 4.6, we conducted a comparative analysis of our classifier with several recent models that have been trained with the CIFAR dataset. That are the following:

- **Hybrid (PiN)**: Vision Xformers: Efficient Attention for Image Classification [30]
- **Vision Nystromformer (VIN):** Vision Xformers: Efficient Attention for Image Classification [30]
- **SmoothNetV1:** SmoothNets: Optimizing CNN architecture design for differentially private deep learning [31]

Classifier	Params	Test Accuracy (%)	Reconstruction loss (MSE)
Our Classifier	1.8M	**85.12**	0.12 (Where the MSE loss= 0.249)
Hybrid (PiN) (2021)	0.99M	74	/
Vision Nystromformer (VIN)(2022)	0.53M	65.06	/
SmoothNetV1 (2022)	3.41M	73.5	/

Table 4.3: Performance comparaison with first framework where both the encoder and classifier are trained.

4.3.1.5 Comments

The advantage of our approach lies in the fact that we achieve a reduced size of the CIFAR dataset. Despite training the encoder, our classifier is simultaneously trained to utilize the latent representation. This implies that during testing, a smaller dataset is required, resulting in a reduced size of the classifier. As a result, the inference time is significantly reduced. Essentially, our approach allows for the creation of efficient convolutional neural networks by leveraging a compact dataset. It is worth noting that the size of the CIFAR dataset, from the classifier's perspective, is reduced by a factor of 3,5. Here is two exemples:

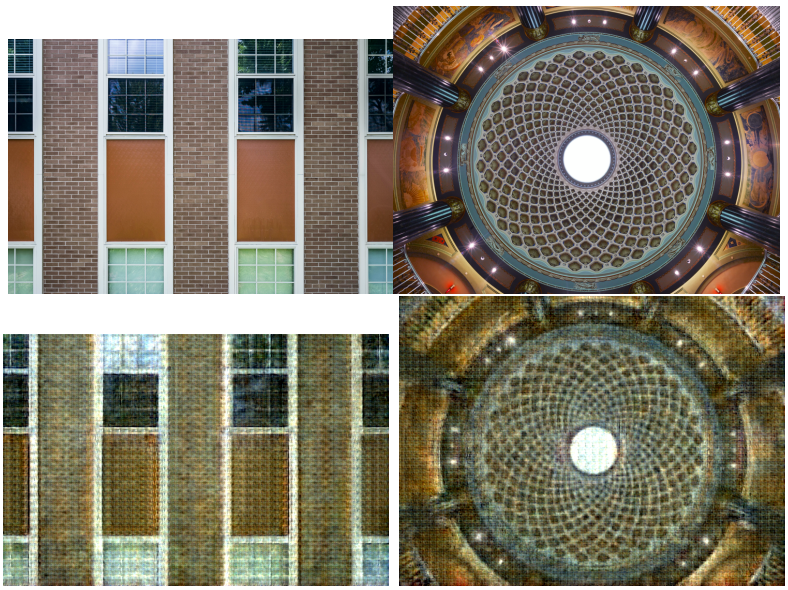

Figure 4.7: Upper row: Original images. lower row: Reconstructed with classification accuracy of 85.12%.

Simultaneously, it has come to our attention that the reconstruction loss has exhibited a noticeable surge, a phenomenon that can be deemed entirely expected given our intentional disregard for its significance within the loss function. Our primary objective, after all, revolved around exploring the outer boundaries of the latent representation, particularly in relation to the computer vision task at hand.

4.3.2 Framewrork 2: Classifier

4.3.2.1 Architecture

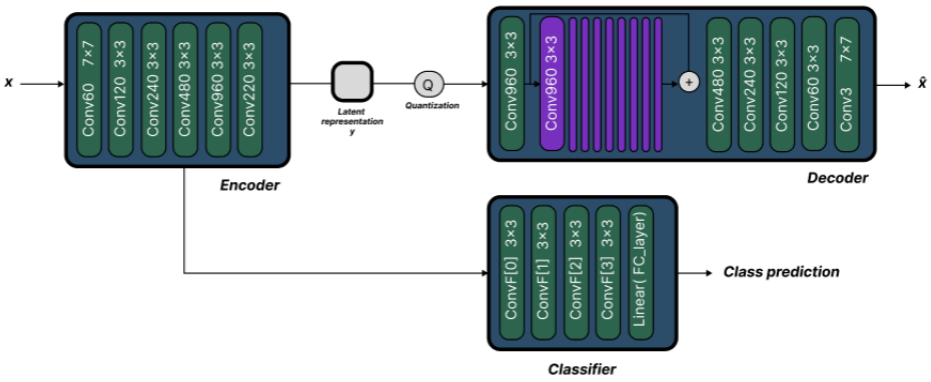

Figure 4.8: Protocol 2's architecture containing Encoder E on the left and Decoder/Classifier on the right taking the 4th layer of the encoder as input to the classifier.

To implement this framework in figure 4.8, we will utilize a pretrained encoder and decoder. The weights of these components will be frozen to prevent further training. Subsequently, we will extract the 4th layer from this architecture and employ it for classification purposes, leveraging our own classifier. Our classifier will be trained using its designated loss function. The rationale behind selecting the 4th layer lies in its ability to contain more informative features compared to the last layer, thereby promoting effective classification.

For the architecture of this model we have the same modules of the precedent protocol, the main difference is that the E encoder and D the decoder are already pretrained like in [23] and the only trainable part is the classifier. Where it architecture is:

- **Convolutional Block 1:** It takes an input with 220 channels and applies F[0] filters of size 2x2.
- **Batch Normalization layer:** It normalizes the output of the previous convolutional layer by subtracting the mean and dividing by the standard deviation computed over a batch of samples.

- Same layer are multiplied with different channels (as shown in the figure 4.8). Untill the FNN to classify 10 classes

4.3.2.2 Loss function

Since we are training the classifier only where a multi class classification loss is:

$$C(o,c) = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{4.5}$$

So the global loss for this model will be:

$$L = \lambda \mathbb{E}[-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})] \tag{4.6}$$

4.3.2.3 Training

The model has been trained for 30 epochs using stochastic gradient descent (SGD) and a fixed learning rate of 0.00045. The architectural details of the model are provided in the table below.

Operation	channels	activation
$conv3x3$	480	relu
$conv3x3$	960	relu
$conv2x2$	460	relu
$conv2x2$	300	relu
$conv2x2$	150	relu
$FClayer$	10	softmax

Table 4.4: Classifier of the second framework architecture.

The results are visualized in the figure below:

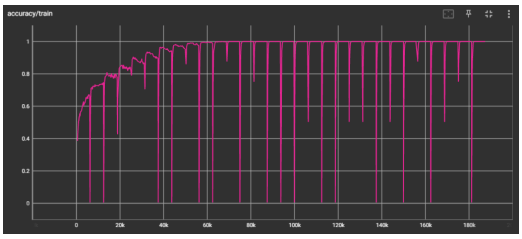

Figure 4.9: Training accuracy for the second framework for 30 epochs.

The validation set plots are depicted in the figure presented below:

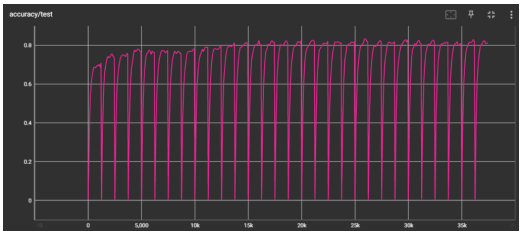

Figure 4.10: validation accuracy of the second framework.

The following table presents the results, comparing them with the previous models:

Classifier	Params	Test Accuracy (%)	Reconstruction loss (MSE)
Our 2^{nd} Classifier	10.0M	**83.0**	0.249
Hybrid (PiN) (2021)	0.99M	74	/
Vision Nystromformer (VIN)(2022)	0.53M	65.06	/
SmoothNetV1 (2022)	3.41M	73.5	/

Table 4.5: Performance comparaison with second framework where both the encoder and classifier are trained.

Comments

As anticipated, the accuracy has declined compared to the previous framework due to the encoder being fixed. This restriction prevents it from incorporating the essential information for classification into the latent representation. However, it is noteworthy that the reconstruction loss remains unchanged, similar to the findings in [23], which is equal to 0.24, it is important to note that the model is trained on MSSIM.

4.3.3 Comparison

In this section, our aim is to compare the latent representations obtained from two different encoders. Firstly, we have the latent representation from encoder trained solely for reconstruction, denoted as X. Secondly, we have the latent representation from encoder trained for both reconstruction and classification accuracy, denoted as X'. These two representations represent the extreme ends of our comparison. To measure the dissimilarity between them, we employ cosine similarity:

$$comparaison = \mathbb{E}\left[1 - \frac{\boldsymbol{X} \cdot \hat{\boldsymbol{X}}'}{\|\boldsymbol{X}'\|\|\hat{\boldsymbol{X}}\|}\right] \tag{4.7}$$

This metric allows us to assess the extent to which each component of the latent representation differs from other components.

Following the computation and subsequent averaging of the cosine differences within the LFW dataset and CIFAR Data set, we derived the following table:

Data set	Mean value of the Cosine loss	Standard deviation of the cosine loss
CIFAR-10	0.6135	0.011
LFW	0.5931	0.004

Table 4.6: Comparison table of the two latent representations derived from different encoders optimized for different objectives using cosine similarity.

comments

Initially, we observed that despite altering the dataset, the cosine loss exhibited minimal variations, indicating a direct correlation with the encoder weights. Moreover, we observed an increase in accuracy from 83% to 85% and a halving of the Mean Squared Error (MSE). The cosine similarity remained around 0.6, suggesting that even with a modified latent representation, the decoder successfully decoded the latent representation.

4.4 Conclusion

In this chapter, we not only achieved highly satisfactory classification results using compressed images, but we also examined and compared the impact of different representations on accuracy and restoration metrics. We conducted experiments using two distinct protocols: one involved freezing the encoder, while the other involved training it alongside the classifier. Through this analysis, we observed how variations in the representation directly influenced the differences in accuracy and restoration metrics.

In the next chapter we will see a new enhanced model that not only capable of computer vision task, but also image processing.

Chapter 5

Enhanced JPEG AI model with integrated face recognition and augmented resolution

5.1 Introduction

JPEG AI aims to develop a learning-based image coding standard that provides a compact compressed domain representation. It focuses on enhancing compression efficiency compared to commonly used image coding standards while maintaining equivalent subjective quality for human visualization. Additionally, it aims to deliver effective performance for image processing and computer vision tasks, in this chapter we will adress a new image compression that do the three main tasks firstly compression-reconstruction, augmenting the resolution as an image processing task, and face recognition as computer vision.

5.2 Motivation and objectives

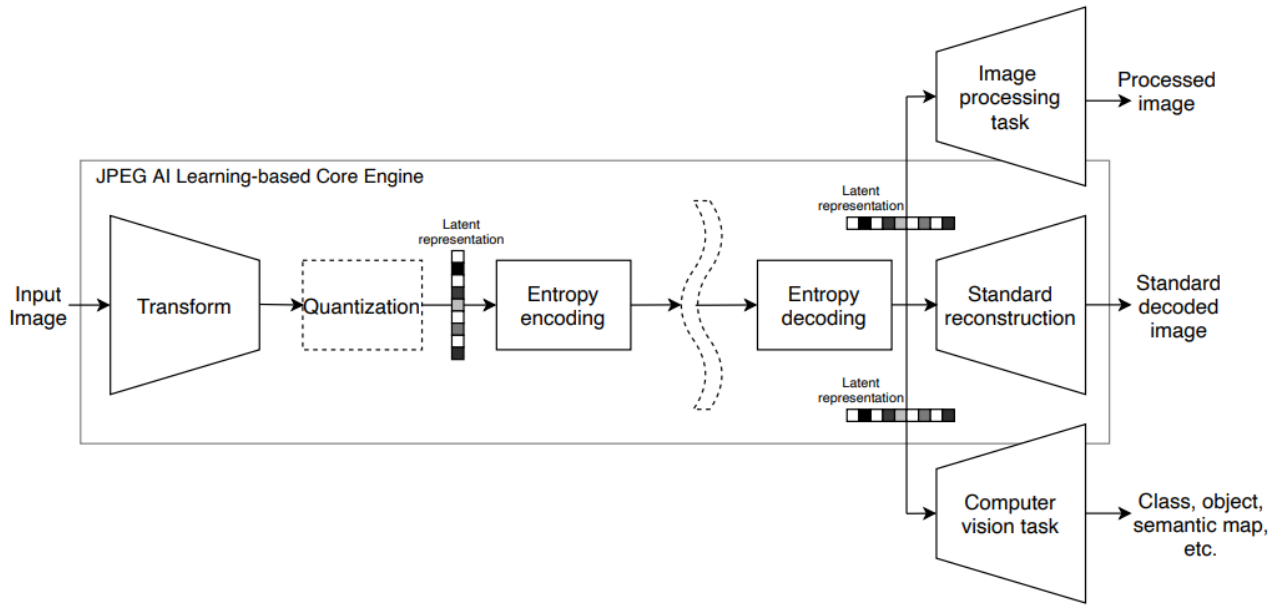

Figure 5.1: JPEG AI learning-based image coding framework.

The aim of introducing JPEG AI is to develop an image compressor capable of effectively performing image compression, image preprocessing, and computer vision tasks [32] presented in figure 5.1. Given the absence of existing models that cater to all three tasks simultaneously, our objective is to create a model that addresses each main task by focusing on a specific subtask within them. We chose for our model the following tasks:

Compression	Image Processing	Computer vision
Reconstruction	Augmenting image resolution	face recognition

Table 5.1: Our model tasks.

Its also important to note that we will not train the decoder for such a task. Since training all the CODEC for each task will be hard to implement on a large scale since their will be an "infinite" numbers of CODECs for each task. So we used the one in [23] which is optimized for image reconstruction and can go to lower BPPs. For this work optimizing the entropy to get lower BPP is not our main focus, its the information held in the latent representation and how to adapt to different objectives, that are the following:

- **Reconstruction:** The primary objective of our codec revolves around two main steps: firstly, to meticulously reconstruct the image, and secondly, to effectively utilize the latent representation for various additional purposes.

- **High resolution:** The reason behind selecting this particular task as our computer vision project stems from a logical progression that aligns with our future endeavors focused on optimizing the Bpp (bits per pixel) metric.

- **Face recognition:** Many modern devices and services use face recognition to personalize user experiences. For example, social media platforms can suggest tags for people in photos, and smart TVs can recommend content based on who is watching. Face recognition enables tailored experiences and enhances user convenience.

5.3 Preliminaries

Prior to delving into the intricacies of our model's architecture, we would like to provide an introduction to some foundational concepts, including an overview of the traditional methods employed in face recognition.

5.3.1 Face recongition

Face recognition can be divided into two main parts: feature extraction and classification. These two steps are integral to the overall process of recognizing and identifying faces. Where the feature extraction part focuses on extracting relevant and distinctive features from the input face image. The goal is to transform the raw input image into a compact representation (or embedding) that captures the essential characteristics of the face, making it easier for the classification part to differentiate between different individuals. Generally in the state of the art it a (512,1) vector.

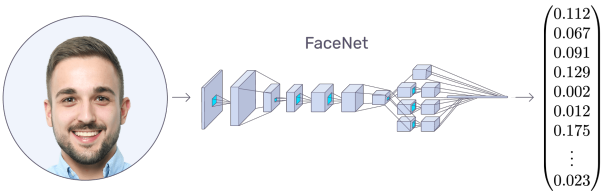

Figure 5.2: Facial embedding extraction using Facenet.

Once the feature extraction step is complete and the face image is transformed into a feature vector, the classification part comes into play. The goal of this step is to determine the identity of the face based on the extracted features. One commonly used technique for face recognition classification is the cosine similarity-based loss function, known as the Cosine Loss. The Cosine

Loss measures the similarity between two feature vectors by computing the cosine of the angle between them. It quantifies the degree of similarity, ranging from -1 (completely dissimilar) to 1 (identical).

5.3.1.1 MobileFacenet

MobileFacenet [33] is a facial feature extractor specifically designed for deployment on mobile devices such as smartphones and tablets. It is optimized to achieve accurate and efficient face recognition while maintaining a small model size and low computational requirements, making it suitable for real-time applications on mobile platforms. The development of MobileFacenet addresses the challenges posed by limited computational resources and power constraints on mobile devices. It aims to provide a compact yet powerful solution for facial recognition tasks without compromising performance. It has the following architecture:

Input	Operator	t	c	n	s
$112^2 \times 3$	conv3x3	-	64	1	2
$56^2 \times 64$	depthwise conv3x3	-	64	1	1
$56^2 \times 64$	bottleneck	2	64	5	2
$28^2 \times 64$	bottleneck	4	128	1	2
$14^2 \times 128$	bottleneck	2	128	6	1
$14^2 \times 128$	bottleneck	4	128	1	2
$7^2 \times 128$	bottleneck	2	128	2	1
$7^2 \times 128$	conv1x1	-	512	1	1
$7^2 \times 512$	linear GDConv7x7	-	512	1	1
$1^2 \times 512$	linear conv1x1	-	128	1	1

Table 5.2: MobileFaceNet architecture for feature embedding.

Each line describes a sequence of operators, repeated n times. All layers in the same sequence have the same number c of output channels. The first layer of each sequence has a stride s and all others use stride 1. All spatial convolutions in the bottlenecks use 3×3 kernels. The expansion factor t is always applied to the input size. GDConv7x7 denotes GDConv of 7×7 kernels.

In our research, we will employ the model as a feature extractor to promote the encoder's emphasis on facial features during the compression process of the original image.

Initially, we focused on comparing the features exclusively as an initial step. This approach is

taken because obtaining similar embeddings would likely yield similar classification outcomes. Exploring this connection further will be the focus of our future endeavors.

5.3.2 Augmented resolution

To accomplish the augmented resolution task portion of this enhanced model, we will employ a network model described in table 5.4. This model incorporates a double sampling layer that increases both the height and width of the 220 channels in the latent representation. Subsequently, this augmented representation is fed into the decoder, where it reconstructs the input by doubling its original dimensions through a multiplication by 2, followed by 16.

5.4 Global framework

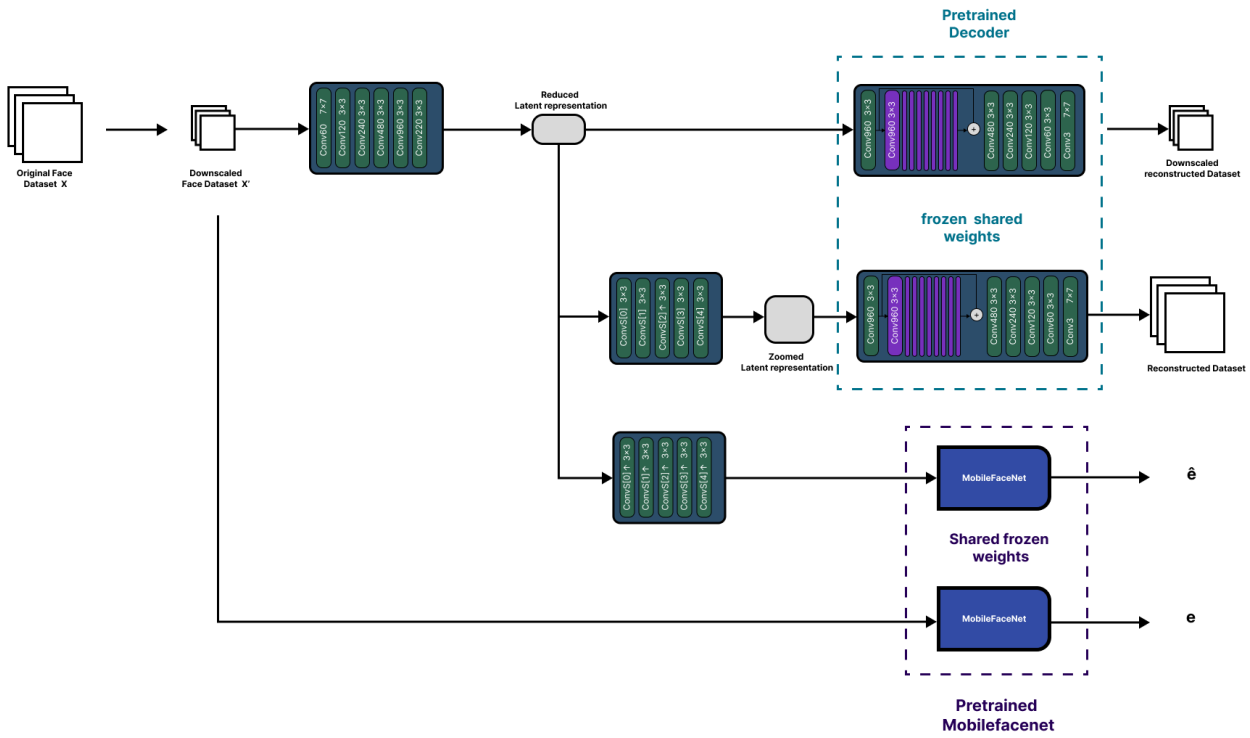

Figure 5.3: Our JPEG AI model's global architecturee, containing encoder to compress, Decoder to decompress, Facenet to adapt latent represenation as an input to mobilefacenet and supnet to make a bigger representation to be decoded by same decoder.

Initially, our approach shown in figure 5.3 involved reducing the dimensions of the LFW data set X to create a modified data set X', which would serve as a reference for the augmented resolution task. The reduction process entailed halving the height and width of X, resulting in a new data set X' with dimensions of (3x125x125) instead of (3x250x250). The primary objective of this reduction is to establish a ground truth for the subsequent steps.

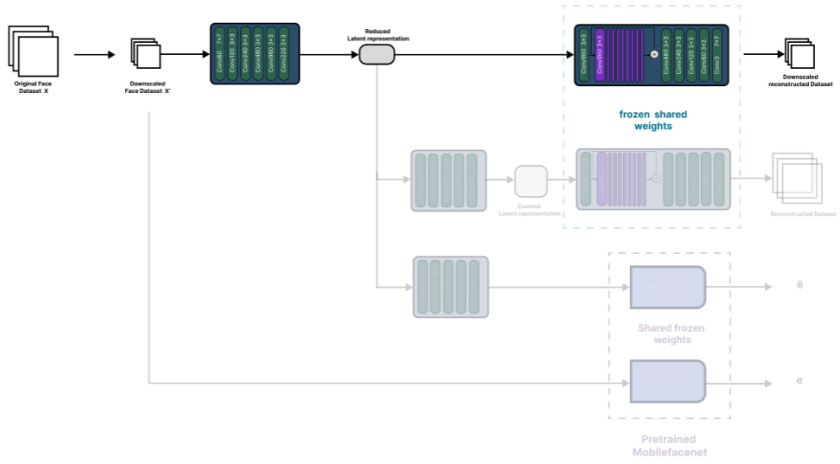

Figure 5.4: First path for reconstruction.

Subsequently, as seen in figure 5.4 we fed the reduced data set X' into the encoder E, which generated a compressed latent representation $E(X')$ with a size of 220x7x7. To evaluate the reconstruction quality, we employed the decoder D by inputting the reduced latent representation, resulting in the reconstructed reduced data set $D(E(X'))$, denoted as $\hat{X}'$. The distance between X' and $\hat{X}'$ is then compared using metrics such as Mean Squared Error (MSE) or multiscale structural similarity index (MSSIM/SSIM) [17].

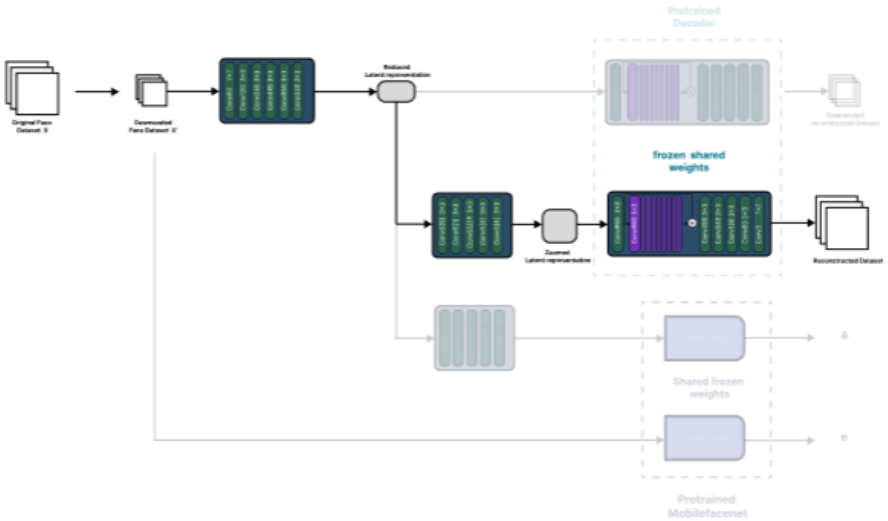

Figure 5.5: Model's second path for the augmented resolution.

Simultaneously, figure 5.6 shows we utilized the reduced latent representation $E(X')$ and passed it through the **supnet** S, a neural network specifically designed to handle augmented resolution tasks. The output of $S(E(X'))$ is a zoomed version of the latent representation, yielding dimensions of 220x14x14. This enlarged latent representation is subsequently fed into the same decoder whith the same frozen weights to obtain the augmented resolution output $D(S(E(X')))$, denoted as $\hat{X}$. During the training process, $\hat{X}$ is then compared with the original data set X using metrics such as Mean Squared Error (MSE) or multiscale structural similarity index

(MSSIM/SSIM) [17].

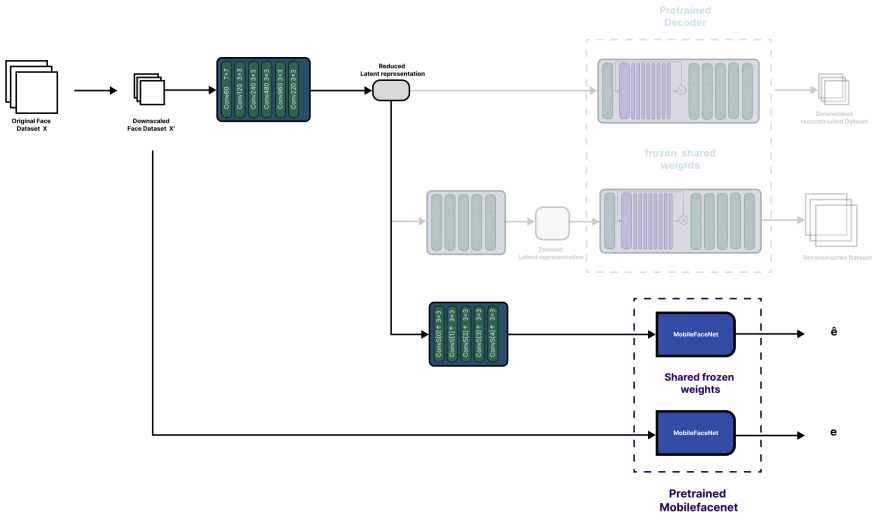

Figure 5.6: Model's third path for facial features.

On a parallel track, figure 5.6 explicit how we took the reduced latent representation $E(X')$ and utilized the **facenet** F to adapt it. This adaptation resulted in a decoded latent representation $F(E(X'))$, which then served as the input for the Mobilefacenet M [33]. The output of M, denoted as $\hat{e} = M(F(E(X')))$, represented an embedding produced by the modified latent representation, in other words it **facenet** work as a decoder just instead of extracting all the original information it decodes only the facial feature. This embedding $\hat{e}$ is then compared with the embedding e derived from the reduced data set X using the Mobilefacenet M, providing another means of evaluation using the the cosine similarity.

5.4.1 Modules architecture:

5.4.1.1 Encoder

We use the same architicture as the pervious experiences which will be the following:

Input	filters	channels
$H \times W \times 3$	conv 7x7	60
$H \times W \times 60$	conv3x3	120
$(H/2) \times (W/2) \times 120$	conv3x3	240
$(H/4) \times (W/4) \times 240$	conv3x3	480
$(H/8) \times (W/8) \times 480$	conv3x3	960
$(H/16) \times (W/16) \times 960$	conv3x3	220

Table 5.3: Encoder architecture.

Which will reduce the output by a factor of ≈ 3.5 since we down sampled 4 times, with 220 channels in the last layer.

5.4.1.2 Supnet

The architecture for our supnet can be described in table 5.4. It is of utmost importance to

Input	filters	channels
$H \times W \times 220$	conv 3x3	700
$H \times W \times 700$	conv3x3	500
$H \times W \times 500$	conv3x3	400
$H \times W \times 400$	upscale	/
$(H \times 2) \times (W \times 2) \times 400$	conv3x3	300
$(H \times 2) \times (W \times 2) \times 300$	conv3x3	220

Table 5.4: Supnet's architecture.

understand that following each convolutional layer, the inclusion of batch normalization plays a crucial role in normalizing the activations. Additionally, the purpose of incorporating an upscaling layer so in the decoding process we enhance the resolution by doubling it, resulting in a larger output size.

5.4.1.3 Facenet

Due to the fact that mobilefacenet is trained using actual RGB images rather than latent representations, it became necessary for us to develop facenet as some sort of a decoder architecture. This involved incorporating 3 channels at the end of facenet and increasing the height and width dimensions beyond or equal to 112x112, which is the required input dimension for facenet in order to have the correct embedding (512). Our approach, as outlined in table 5.5, effectively implemented these modifications. Additionally, it is crucial to note that when scaling up by multiplying the latent dimension by a power of 2, we encountered the need to perform a center crop just before injecting it into mobilefacenet.

Input	filters	channels	activation
$H \times W \times 220$	conv 3x3	400	relu
$(H \times 2) \times (W \times 2)400$	conv3x3	360	relu
$(H \times 4) \times (W \times 4)$ 360	conv3x3	240	relu
$(H \times 8) \times (W \times 8)$ 240	conv 3x3	120	relu
$(H \times 16) \times (W \times 16) \times 120$	60	60	relu
$(H \times 32) \times (W \times 32) \times 60$	conv3x3	3	/

Table 5.5: Facenet's architecture.

It is crucial to emphasize that we incorporated batch normalization after each layer throughout the facenet. This technique has been applied consistently to ensure proper normalization and stability at every stage of the facenet's operation.

5.4.1.4 Decoder

In order to construct the decoder, we opted to employ the exact same decoder as in a previous research study [23], leveraging its pretrained weights. The architectural intricacies of this decoder have already been extensively elucidated earlier in this thesis.

5.5 Loss function

Given that there are three tasks to be undertaken, it is evident that there will be three distinct losses that need to be optimized. The first loss, known as the distortion loss, involves comparing the downsampled dataset, denoted as X', with the downsampled reconstruction obtained through the process of encoding and decoding, represented as $D(E(X'))$. This comparison is carried out using the Mean Squared Error (MSE) metric to quantify the level of distortion:

$$\mathcal{L}_{\text{reco}} = \mathbb{E}[(X' - \hat{X}')^2] = \mathbb{E}[(X' - D(E(X')))^2] \tag{5.1}$$

The second loss, referred to as the augmented resolution loss, aims to measure the dissimilarity between the original dataset, denoted as X, and the reconstruction obtained after introducing the downsampled dataset to the encoder and subsequently passing it through the supernet $\hat{X} = D(S(E(X')))$. The purpose of this loss is to assess the effectiveness of the reconstruction in capturing the finer details of the original data. Where the loss is denoted:

$$\mathcal{L}_{\text{recoHR}} = \mathbb{E}[(X - \hat{X})^2] = \mathbb{E}[(X - D(S(E(X'))))^2] \tag{5.2}$$

The third loss is determined by calculating the cosine similarity shown in equation 5.3, which involves comparing the embedding resulting from the injection of the downsampled data set into MobileFaceNet with the embedding obtained by decoding the reduced latent representation using Facenet and subsequently injecting it into the same MobileFaceNet.

$$\mathcal{L}_{\text{face}} = \mathbb{E}\left[1 - \frac{\boldsymbol{e} \cdot \hat{\boldsymbol{e}}}{\|\boldsymbol{e}\|\|\hat{\boldsymbol{e}}\|}\right] \tag{5.3}$$

In summary, due to the presence of three distinct tasks, there are corresponding losses to optimize: the distortion loss, which evaluates the downsampled dataset against its reconstruction using MSE; and the augmented resolution loss, which measures the dissimilarity between the original dataset and the reconstruction after passing through the encoder and supernet, and the cosine loss. Resulting in the global loss being:

$$\mathcal{L} = \lambda_{\text{face}}\mathcal{L}_{\text{face}} + \lambda_{\text{recoHR}}\mathcal{L}_{\text{recoHR}} + \lambda_{\text{reco}}\mathcal{L}_{\text{reco}} \tag{5.4}$$

$$\boldsymbol{\mathcal{L}} = \boldsymbol{\lambda}_{\text{reco}}\mathbb{E}\left[(\boldsymbol{X'} - \boldsymbol{D}(\boldsymbol{E}(\boldsymbol{X'})))^2\right] + \boldsymbol{\lambda}_{\text{recoHR}}\mathbb{E}\left[(\boldsymbol{X} - \boldsymbol{D}(\boldsymbol{S}(\boldsymbol{E}(\boldsymbol{X'}))))^2\right] + \boldsymbol{\lambda}_{\text{face}}\mathbb{E}\left[1 - \frac{\boldsymbol{e} \cdot \hat{\boldsymbol{e}}}{\|\boldsymbol{e}\|\|\hat{\boldsymbol{e}}\|}\right] \tag{5.5}$$

The purpose of incorporating λ_i within the loss error function is to assign varying levels of

importance to different types of losses. However, for our specific task, where we will be engaging in a training phase followed by a fine-tuning process (which will be elaborated upon later), we have decided to set all the lambdas to be equal to 1/3, representing the average value of all three losses.

5.6 Training

In order to achieve optimal convergence, we implemented a two-part training approach. Initially, we focused on training each module individually while keeping the others frozen. This involved training the encoder to achieve a high-quality reconstruction. Subsequently, we proceeded to fine-tune all modules together to optimize the overall loss. Following the encoder training, we proceeded to train the supnet and facenet modules separately while keeping the previously trained encoder fixed. We trained all the modules using the LFW dataset.

5.6.1 Encoder's training:

During the training process, we focused on training the encoder module exclusively, while keeping all other modules frozen. This setup is maintained for a total of 30 epochs using the stochastic gradient descent optimizer with a consistent learning rate of 0.01. As a result, we were able to achieve a training loss, specifically Mean Squared Error (MSE), of **0.0049**. Where the MSE **(5.1)** can be visualised in the following figure:

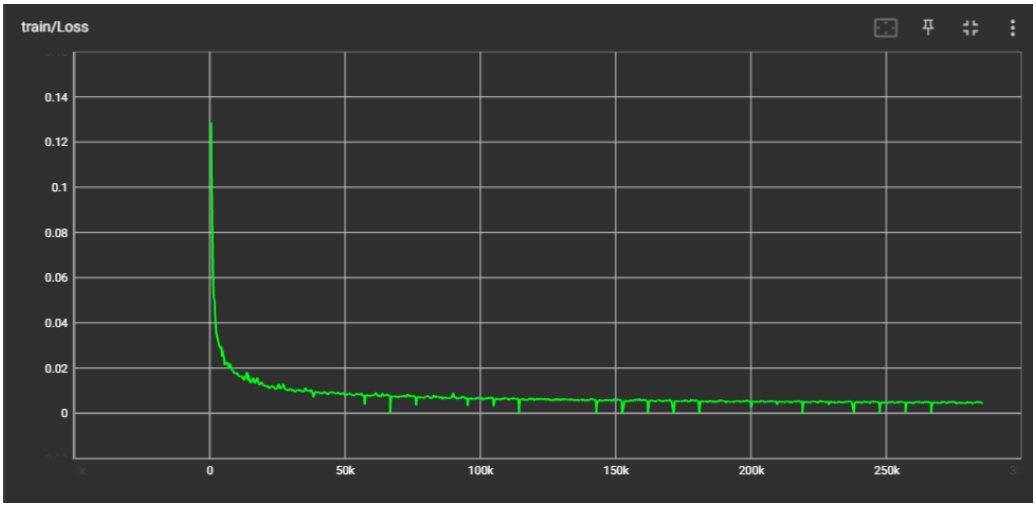

Figure 5.7: Training loss for the encoder.

In order to evaluate the proficiency of the trainer, we opted to utilize images derived from two distinct distributions: one drawn from the same distribution as the dataset being analyzed, and another obtained from a distribution that lies outside of it. This approach enables us

to comprehensively assess the trainer's performance across varying data sources and ascertain its adaptability to both familiar and unfamiliar distribution patterns. We can visualize some exemples of reconstruction:

Figure 5.8: Upper row: Original images. Lower row: Reconstructed images with the encoder trained with MSE

5.6.1.1 Results and performance evaluation

To assess the quality of our model's reconstructions, we utilized a set of nine images from the JPEG AI test set in [34], which is the same approach during the 85th JPEG Meeting held in San Jose, USA, in November 2019. These images, which are crucial for our evaluation, can be visualized in the accompanying figure:

Figure 5.9: JPEG AI 9 high resolution test images.

The table 5.6 presents the MSE loss values we obtained. (The images are numbered from up to down from right to left)

Image	Loss
TE 01	0.0181
TE 02	0.0197
TE 03	0.0246
TE 04	0.0180
TE 05	0.0114
TE 06	0.0182
TE 07	0.0333
TE 08	0.0042
TE 09	0.063
mean	0.023

Table 5.6: MSE loss of the nine JPEG AI test images.

Exemple

We can visualize TE08 :

Figure 5.10: Left: Original. Right: Reconstruction using encoder trained on MSE only.

5.6.1.2 Comments

We successfully reached an MSE of 0.023, which is considered good when considering the visual aspect. However, during our analysis, we observed some larger losses, such as TE 09 with an MSE of 0.06. These higher losses can be attributed to the image containing numerous intricate details that require more intricate coding.

5.6.2 Training Facenet

In the previous section, we employed the Pretrained encoder and kept its weights fixed. Our objective is to minimize the cosine loss, mentioned in equation 5.3. The purpose behind this is to minimize the angle, enabling us to extract facial features from the latent representation using the facenet. This approach allows for the direct utilization of latent representations in future endeavors related to face recognition. We conducted model training for 100 epochs, employing stochastic gradient descent with a fixed learning rate of 0.01. This approach enabled us to achieve an angle difference of less than 7 degrees or a cosine loss of 0.1. We can see the training loss plot in the following figure:

5.6.2.1 Results

Since its not the main objective we didn't extend the model to the classifier point, so we are currently unable to assess these outputs. Therefore, we rely solely on observing the outcomes, anticipating a subsequent decline in the accuracy of such classifiers. Additionally, we can

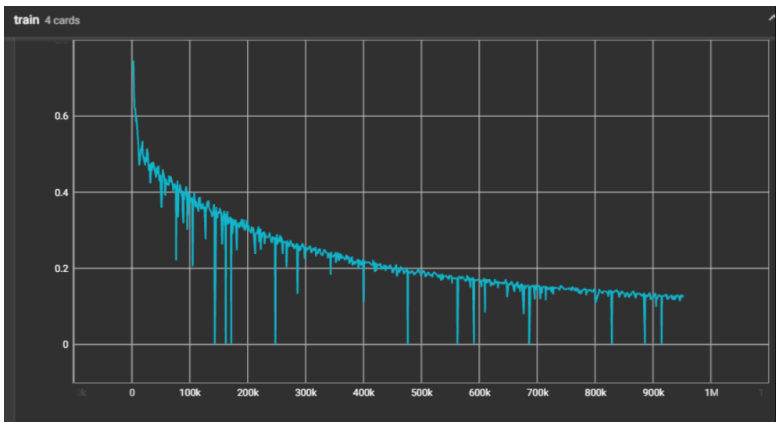

Figure 5.11: Training plot of facenet.

compare the generated output with those mentioned in [35]. The following figure shows the original input and the facial features in the output of facenet:

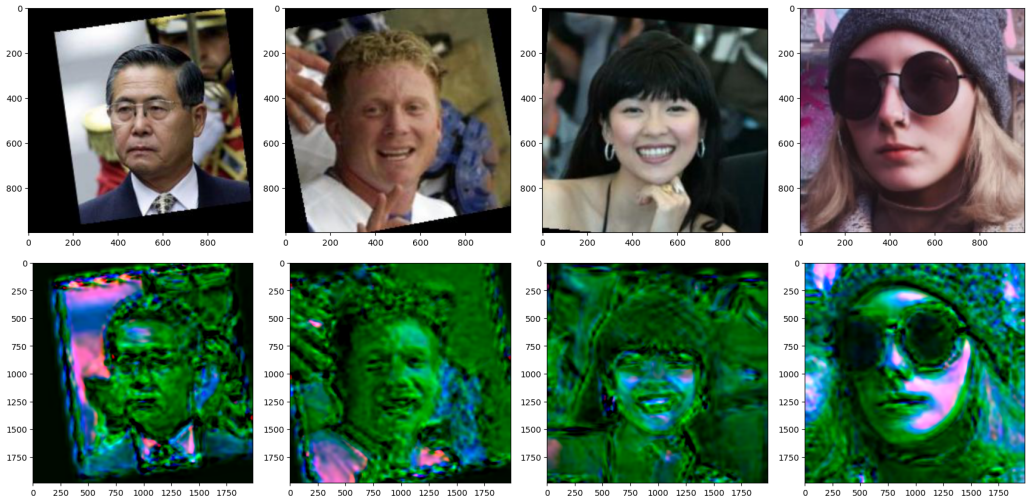

Figure 5.12: Upper row: Originals. Lower row: facenet output.

Upon observation, it becomes evident that the absence of RGB colors is quite prominent in the visual display. Instead, there is a predominant presence of the color green, which can be considered somewhat logical given that facial features do not inherently rely on color. It is indeed possible to perform face recognition solely based on gray scale information, without the need for color distinctions. Simultaneously, the model places significant emphasis on crucial aspects like facial noise and the eyes, deeming them noteworthy for recognition purposes. Conversely, it tends to overlook less significant elements such as the background and hair, which are deemed less pertinent in the context of facial recognition.

Furthermore, the provided figure presents a comparison between the model discussed in [35] where their CODEC is specifically designed and trained to enhance facial recognition accuracy. the model achieved an accuracy rate of 89.48%.

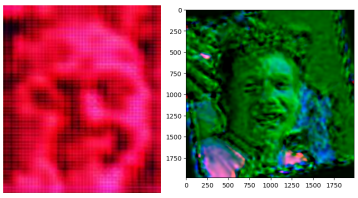

Figure 5.13: Left: output of the decoder in [33].

5.6.3 Training supnet

As previously stated, our approach involved downsampling the original dataset in order to obtain the ground truth. To achieve this, we maintained the encoder that had been pretrained in the aforementioned section. We then proceeded to optimize the loss, as described in equation 5.2. Our training process spanned 70 epochs, during which we utilized the SGD optimizer and employed a variable learning rate. The learning rate began at 0.1 and underwent a reduction of 95% with each epoch. This training regimen led us to attain a final loss value of **0.018**. The training loss can be visualized in the figure provided below:

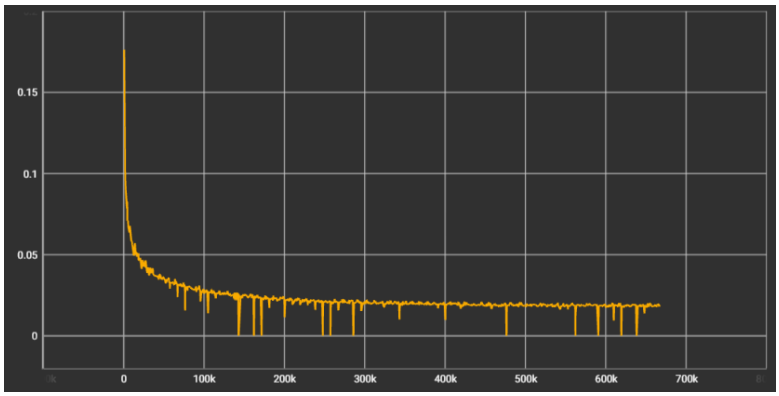

Figure 5.14: supnet training loss.

We are going to down sample the test images than compared the output of the decoded augmented resolution with the original images, and the results are presented in the table below:

Image	Loss
TE 01	0.0140
TE 02	0.0207
TE 03	0.0235
TE 04	0.0269
TE 05	0.0320
TE 06	0.0273
TE 07	0.0402
TE 08	0.0104
TE 09	0.0659
mean	0.023

Table 5.7: MSE_HR loss of the nine JPEG AI test images.

Comment

It is expected to have a somewhat lower resolution in the output during the initial training step, as we kept the encoder fixed. This approach enables us to augmented the resolution but results in a perceptibly lower quality output. Even tho it has been trained on a face data set.

We can provide an example to illustrate this concept visually:

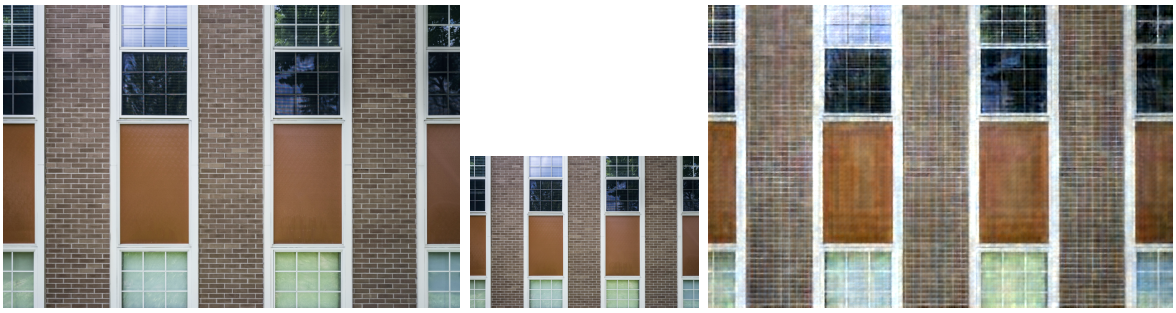

Figure 5.15: Left : Original TE07. Middle: Downscaled TE07. Right: Output of our model with augmented resolution.

5.6.4 Fine Tuning

After training each model individually, we proceeded to perform a fine-tuning process involving all three models. This fine-tuning process aimed to minimize the global loss mentioned in equation 5.5. We trained all the models for a total of 50 epochs, utilizing stochastic gradient descent with a learning rate of 0.002, which progressively decreased by a factor of 0.95 after each epoch. Additionally, it is worth noting that all the λ_i values were set to $\frac{1}{3}$ during this fine-tuning stage.

In the figure 5.16, we have the ability to visually perceive both the global loss plot as well as each individual loss value that comprises it.

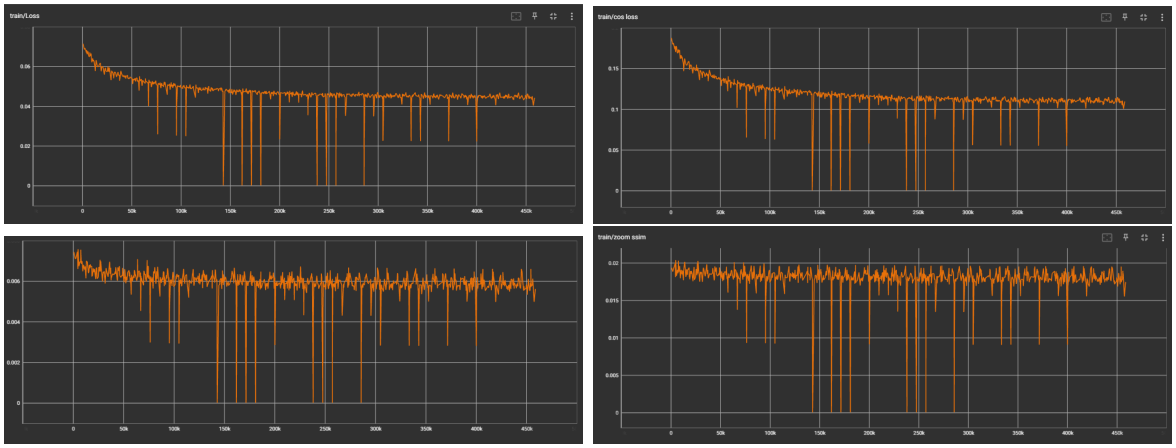

Figure 5.16: Losses plot: Up right: The global loss which is a combination of the three. Up left: cosine similarity loss. lower left : MSE loss for the reconstruction. lower right: MSE loss for the augmented resolution.

Within the depicted figure, positioned in the upper right section, we find the mean value derived from the combination of three distinct losses, all assigned equal weight (where lambdas are set to 1/3). Adjacent to this mean value is the representation of the cosine similarity loss. In the left bottom area of the figure, it represent the reconstruction Mean Squared Error (MSE) loss. Positioned in the bottom right portion of the figure, there is the doubled resolution loss MSE_HR.

Comments

Initially, we observe a significant decline in the overall global criteria function, a phenomenon predominantly influenced by the cosine similarity loss. This particular metric exhibits a remarkably substantial decrease compared to others. Additionally, we observe that despite employing an exceedingly minute learning rate, the MSE loss exhibits a sluggish decline. This slow descent suggests that we have reached a point of optimization. Similar behavior is observed with

the MSE_HR loss, which has remained almost constant since the inception of training and has shown minimal fluctuations, despite not freezing the encoder this time. This outcome is to be expected, considering that there are other crucial aspects to prioritize and emphasize.

In the meantime, it would be interesting to observe the results produced by faceNet shown in figure 5.17, considering the observed reduction in cosine similarity. This brings us closer to identifying the distinctive features that Mobile FaceNet considers crucial for face recognition.

Figure 5.17: Upper row: Originals. Middle row: facenet output before fine tuning. Lower row: facenet output after fine tuning.

Upon observation, we notice that the latest iteration of the fine-tuned Facenet model places a heightened emphasis on crucial facial attributes, namely the eyes, nose, and mouth...etc. In stark contrast to its previous version, the updated model assigns diminishing significance to surface-level elements such as the front of the head, cheeks, or even the presence of sunglasses, as illustrated in the right image. Moreover, it progressively converges towards a single dominant channel (green), suggesting a refined focus on capturing and representing the essential facial features.

5.7 Conclusion

In summary, the primary objective of this chapter is to address the specific requirements imposed by JPEG AI, necessitating the development of a codec capable of effectively compressing and decompressing photographic images. Furthermore, we extended its functionality to encompass additional image processing tasks, such as the application of Supnet for doubling the resolution. Additionally, we undertook a computer vision task focused on adapting the latent representation for facial recognition purposes, utilizing Facenet to extract the necessary facial features which is trained with mobile facenet. The training process for these modules involved a two-fold approach, initially training each model independently, followed by a comprehensive fine-tuning applied to all the models collectively.
To provide a more comprehensive summary of the final model's process, we can refer to the accompanying figure that encapsulates the essence of its operations:

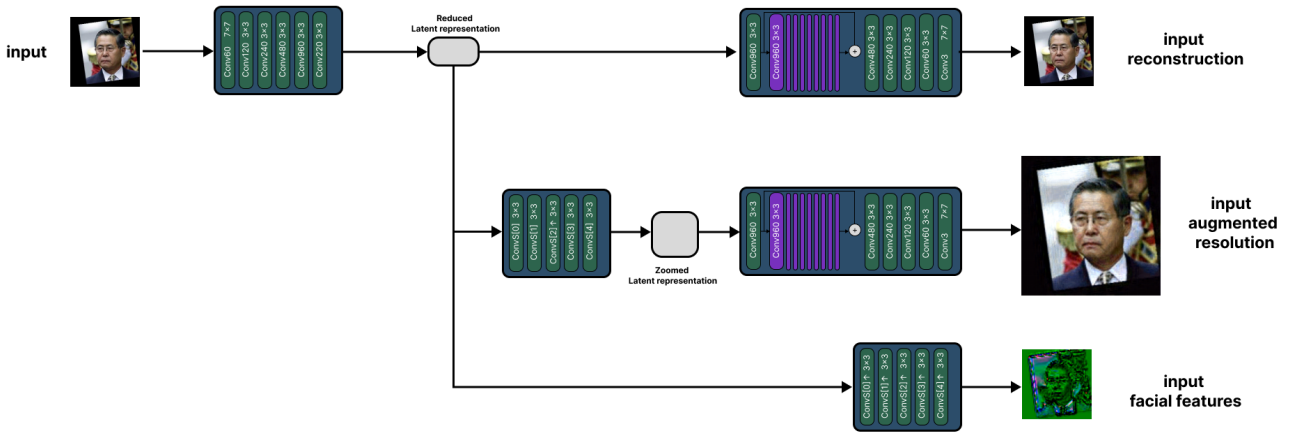

Figure 5.18: Our CODEC process, Image as input, and resulting in 3 outputs : input's reconstruction, input's augmented resolution, input's facial features.

Emphasizing the significance of maintaining a fixed decoder, as opposed to developing and training task-specific decoder, is crucial to promote widespread adoption of such a codec. This approach encourages a unified implementation of the codec, eliminating the need train both the encoder and decoder to a specific task resulting in a large amout of variants.

Chapter 6

Conclusion

6.1 Conclusion

In this research project, we initially employed a classification problem as a means to explore how different encoders, trained for various objectives, can alter the information contained within the latent representation. To assess these changes, we utilized the cosine similarity metric and observed that different performance levels were obtained in both classification and restoration tasks for a given cosine value. This finding suggests that cosine similarity could serve as a parameter for determining the preferred task when utilizing future encoders.

Furthermore, we developed an advanced AI codec specifically designed to address three key objectives: image compression, image processing, and image compression (repeated for emphasis). As an illustrative example, we focused on two specific tasks within the codec: face recognition and doubling the resolution of images. Notably, we obtained satisfactory results in terms of image reconstruction and face recognition. However, our investigation concluded at the stage of facial feature extraction, as the effectiveness of face classification heavily relies on the specific dataset used.

6.2 Future work

For future research, we propose considering the cosine similarity as a parameter when selecting the desired task. Given that the codec has demonstrated its ability to fulfill the three desired tasks of JPEG with a trade-off, future efforts can be directed towards optimizing it for various other tasks. These may include image restoration for image processing, segmentation for computer vision, and more. It is important to maintain a fixed decoder to enable seamless implementation on a large scale.

Bibliography

[1] G.K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992.

[2] A. Skodras, C. Christopoulos, and T. Ebrahimi. The jpeg 2000 still image compression standard. *IEEE Signal Processing Magazine*, 18(5):36–58, 2001.

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

[4] Information technology – digital compression and coding of continuous-tone still images – requirements and guidelines. https://www.w3.org/Graphics/JPEG/itu-t81.pdf.

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

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

[7] G. G. Langdon. An introduction to arithmetic coding. *IBM Journal of Research and Development*, 28(2):135–149, 1984.

[8] Warren S. McCulloch Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, December 1943.

[9] Gradient descent algorithm: A deep dive. *Towards Data Science*.

[10] Anh h. reynolds - convolutional neural networks. https://anhreynolds.com/blogs/cnn.html.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahra-

mani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[12] David Vint, Matthew Anderson, Yuhao Yang, Christos Ilioudis, Gaetano Di Caterina, and Carmine Clemente. Automatic target recognition for low resolution foliage penetrating sar images using cnns and gans. *Remote Sensing*, 13:596, 02 2021.

[13] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[14] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *CoRR*, abs/2003.05991, 2020.

[15] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *CoRR*, abs/1906.02691, 2019.

[16] A.C.; Sheikh H.R.; Simoncelli E.P. Wang, Zhou; Bovik. Image quality assessment: from error visibility to structural similarityimage quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing.*, 2004-04-01.

[17] Z. Wang; E.P. Simoncelli; A.C. Bovik. Multiscale structural similarity for image quality assessment. *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, 2004.

[18] Alain Horé and Djemel Ziou. Image quality metrics: Psnr vs. ssim. pages 2366–2369, 08 2010.

[19] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior, 2018.

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

[21] J. Rissanen and G. Langdon. Universal modeling and coding. *IEEE Transactions on Information Theory*, 27(1):12–23, 1981.

[22] Christopher M. Bishop. Latent variable models. In *Learning in Graphical Models*, pages 371–403. MIT Press, 1999.

[23] Fabian Mentzer, George Toderici, Michael Tschannen, and Eirikur Agustsson. High-fidelity generative image compression, 2020.

[24] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans, 2018.

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[26] Felipe Codevilla, Jean Gabriel Simard, Ross Goroshin, and Chris Pal. Learned image compression for machine perception, 2021.

[27] Yuge Huang, Yuhan Wang, Ying Tai, Xiaoming Liu, Pengcheng Shen, Shaoxin Li, Jilin Li, and Feiyue Huang. Curricularface: Adaptive curriculum learning loss for deep face recognition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5900–5909, 2020.

[28] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[29] Gary B. Huang, Marwan Mattar, Honglak Lee, and Erik Learned-Miller. Learning to align from scratch. In *NIPS*, 2012.

[30] Pranav Jeevan and Amit Sethi. Vision xformers: Efficient attention for image classification, 2021.

[31] Nicolas W. Remerscheid, Alexander Ziller, Daniel Rueckert, and Georgios Kaissis. Smoothnets: Optimizing cnn architecture design for differentially private deep learning, 2022.

[32] JPEG AI. https://jpeg.org/jpegai/index.html.

[33] Sheng Chen, Yang Liu, Xiang Gao, and Zhen Han. Mobilefacenets: Efficient cnns for accurate real-time face verification on mobile devices, 2018.

[34] JPEG AI. Jpeg ai test images. https://jpegai.github.io/test_images/.

[35] Nai Bian, Feng Liang, Haisheng Fu, and Bo Lei. A deep image compression framework for face recognition, 2019.