TÉCNICO LISBOA

Compressed Domain Face Recognition Assessment

Francisco Vieira Ramos Ferreira

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. João Miguel Duarte Ascenso

Prof. Catarina Isabel Carvalheiro Brites Ascenso

Examination Committee

Chairperson: Prof. José Eduardo Charters Ribeiro da Cunha Sanguino

Supervisor: Prof. João Miguel Duarte Ascenso

Member of the Committee: Prof. Luís Alberto da Silva Cruz

June 2023

Declaration

I declare that this document is an original work of my own authorship and that it fulfils all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

I would like to thank my family and friends for all the support, motivation and help throughout my journey in Instituto Superior Técnico. They have allowed me to overcome many obstacles along the years.

I want to thank my mother and father, Maria Isabel Ferreira and Manuel José Ferreira, for providing everything that I needed during this journey. They made sure that I could study and improve, both personally and professionally, throughout all these years. I specially want to thank my mom for all the emotional support and for all the rides and sacrifices. Thank you, mom!

Then, I would like to express my sincere appreciation and gratitude to Prof. João Ascenso and Prof. Catarina Brites. I am truly thankful for all your efforts, guidance, and patience throughout This MSc Thesis. Your support has been invaluable, and I am genuinely grateful for everything you have done for me. Likewise, I want to thank Instituto de Telecomunicações.

In fourth, I am grateful for my group of close friends, they have provided me many fun and exciting moments. They have been one pilar in my life that allowed me to overcome many difficulties, being capable of making me laugh when the time was needed.

I also would like to thank my brother, João Ferreira, and great friend Hugo Matos for all the advices and patience.

Finally, I want to thank Tuca and Bismarck for being both friends and inspirations in my life.

Without you all getting to this phase would have been almost impossible, I am eternally grateful!

Resumo

Nos últimos anos, emergiram mais dispositivos e plataformas que lidam com dados visuais, numa ampla gama de aplicações. Adicionalmente, os avanços na tecnologia levaram a que as imagens fossem representadas com mais dados visuais. Consequentemente, a codificação de imagens tornou-se uma etapa comum e necessária, permitindo superar algumas limitações de armazenamento e requisitos de atraso rigorosos.

Recentemente, aprendizagem automática tem-se tornado sucedida em várias aplicações, com aprendizagem profunda fornecendo soluções de última geração em tarefas relacionadas com imagem. Adicionalmente, surgiram soluções de codificação de imagem baseadas em aprendizagem com um desempenho competitivo face aos métodos convencionais de última geração.

Tradicionalmente, as tarefas de análise de imagem são realizadas sobre imagens reconstruídas, podendo levar a um desempenho mais baixo. No entanto, é possível explorar a representação compactada para executar essas tarefas. Os recursos extraídos durante a codificação da imagem original são usados em vez da imagem decodificada. Recentemente, soluções de processamento de domínio comprimido foram desenvolvidas com sucesso, mas existe a necessidade de investir em mais pesquisa nesta área. O objetivo desta Dissertação é fazer um contributo nesta área e visa a conceção, implementação e avaliação de uma nova solução de verificação facial no domínio comprimido.

Este relatório analisa alguns conceitos e ferramentas de aprendizagem profunda, bem como codecs de imagem convencionais e de aprendizagem produnda relevantes. Adicionalmente, algumas soluções relevantes de domínio comprimido são analisadas e avaliadas. Finalmente, o presente relatório descreve a conceção, a implementação e a avaliação do desempenho dos resultados da situação inicial e das soluções propostas.

Palavras-chave: codificação de imagem, eficiência de compressão, aprendizagem profunda, processamento em domínio comprimido, redes neuronais, verificação facial.

Abstract

In recent years, more devices and platforms that deal with visual data emerged, with a wide range of applications. Also, advances in technology led to images being represented with more visual data. Consequently, image coding became a common and necessary step, allowing to overcome some storage limitations and stringent delay requirements.

Recently, machine learning has become widely successful in several applications, with deep-learning providing state-of-art solutions in image related tasks. Moreover, learning-based image coding solutions have emerged with competitive performance against state-of-art conventional methods.

Traditionally, image processing and computer vision tasks are performed over lossy reconstructions of the image, which can lead to lower performance. However, it is possible to exploit the compressed representation to perform these task. The features extracted while coding the original image are then used instead of the lossy decoded image. Recently, some compressed domain processing solutions have been successfully developed, but there is still the need to invest in more research on this area. The objective of this MSc Thesis is to make a small contribution to fill this gap and targets the design, implementation and evaluation of a novel compression domain face verification solution.

This report reviews some concepts and tools of deep learning, as well as relevant conventional and learning-based image codecs. Also, some relevant compressed domain solutions are reviewed and assessed. Finally, this report describes the design, implementation and performance evaluation of the baseline results and proposed solutions.

Keywords: Image coding, compression efficiency, deep learning, compressed domain processing, neural networks, face verification.

Table of Contents

Declaration iii

Acknowledgments v

Resumo vii

Abstract viii

Table of Contents x

List of Figures xiii

List of Tables xvi

List of Acronyms xvii

1. Introduction 1

1.1. Context and Motivation 1

1.2. Objectives and Structure 2

2. Deep-learning Image Analysis: Foundations and Architectures 3

2.1. Neural Network Basics 3

2.2. Main Neural Network Types 6

2.2.1. Convolutional Neural Network 6

2.2.2. Autoencoder 8

2.2.3. Generative Adversarial Network 8

2.3. Reviewing Some Relevant Deep-learning Image Analysis Solutions 9

2.3.1. Image Classification 9

2.3.2. Object Detection 10

2.3.3. Semantic Segmentation 12

2.3.4. Face Recognition 13

3. Image Coding: A Brief Review 14

3.1. Conventional Coding 14

3.2. Learning-based Coding 17

3.3. Performance Evaluation of Learning-based Image Codecs 18

4. Relevant Compressed Domain Deep Learning Based Image Analysis Solutions 22

4.1. Towards Image Understanding From Deep Compression Without Decoding: Image Classification 23

4.1.1. Objective 23

4.1.2. Architecture 23
4.1.3. End-to-End Training 24
4.1.4. Performance Evaluation 25
4.2. Towards Image Understanding From Deep Compression Without Decoding: Semantic Segmentation 26
4.2.1. Objective 26
4.2.2. Architecture 26
4.2.3. End-to-end Training 27
4.2.4. Performance Evaluation 27
4.3. Scalable Facial Image Compression with Deep Feature Reconstruction 29
4.3.1. Objective 29
4.3.2. Architecture 30
4.3.3. Techniques 30
4.3.4. Performance Evaluation 31
5. Anchor Creation for Compressed Domain Face Recognition 33
5.1. Context and Objectives 33
5.2. Original Anchor for Face Verification 33
5.3. Decoded Anchor for Face Verification 34
5.4. Face Detection and Image Processing 36
5.5. Face Verification 39
5.5.1. ResNet50 Architecture 39
5.5.2. ResNet50 Training 40
5.6. Image Codecs 43
5.7. Similarity Score Computation and Decision Making 44
5.8. Performance Metric Computation 45
5.9. Performance Evaluation 47
5.9.1. Datasets 47
5.9.2. Face Verification Protocol 47
5.9.3. Experimental Results 48
6. Compressed Domain Face Verification 52
6.1. Context and Objectives 52
6.2. Compressed Domain Face Verification Architecture 52

6.3. Compressed Domain cResNet39 54
6.3.1. Architecture 55
6.3.2. Model Training 56
6.4. Performance Evaluation 57
6.4.1. Dataset and Face Verification Protocol 58
6.4.2. Experimental Results 60
7. Final Remarks 66
7.1. Summary and Conclusions 66
7.2. Future Work 68
References 70
Appendix A 80
A. Performance Evaluation of Learning-based Image Codecs 80

List of Figures

Figure 1 – Examples of machine learning real-world applications [2] (left); example of deep-learning image applications [3] (right). 2

Figure 2 – Biological neuron (left) and artificial neuron (right) [7]. 3

Figure 3 – Graphical representation of the Logistic/Sigmoid (left), Hyperbolic Tangent (middle) and the Rectified Linear Unit (right) functions, with unidimensional input. 4

Figure 4 – Representation of an example of a Fully Connected Neural Network (FCNN) with one hidden layer [9]. 4

Figure 5 – Simplified Gradient Descent algorithm flowchart for the training of a neural network. 5

Figure 6 – CNN architecture example with an image input, two convolutional layers, two pooling layers and one fully connected layer [17]. In this case, after every convolution a non-linear activation is used. 6

Figure 7 – Example of the resulting feature/activation map by applying one kernel/filter over an input (lef) [18], representation of convolution operation in the first two windows with stride (right) [19]. 7

Figure 8 – Max pooling and average pooling examples, with filter of size 2x2 and stride 2 [20]. 8

Figure 9 – Autoencoder architecture example with fully connected layers (left)) [23] and example of a general representation of an autoencoder (right) [21]. 8

Figure 10 – General architecture of a GAN. 9

Figure 11 – Building block example of a Residual Network (left) [33], ResNet-18 architecture (right) [37]. 10

Figure 12 – Overview of the object detection system of the R-CNN [40]. 11

Figure 13 – Faster R-CNN example architecture [43]. 12

Figure 14 – Example of semantic segmentation performed by a Mask R-CNN model (left) [48], Mask R-CNN framework for segmentation (right) [49]. 12

Figure 15 – Logic of face recognition systems that use NNs (left) [53], example of a 68 face landmark estimations on a real image (middle) [54], example of the affine transformation applied to a detected face in an image by using the blue points as reference (right) [53]. 13

Figure 16 – JPEG encoder (left) and decoder (right) processing steps [59]. 14

Figure 17 – DCT's basic functions (left) [60], example of DCT coefficients (middle) [59], example of quantized DCT coefficients (right) [59]. 15

Figure 18 – JPEG2000 codec architecture [64]. 15

Figure 19 – DWT transformation on a single tile (left) [67], dyadic decomposition on the all set of tiles (middle) [63], example of the dyadic decomposition on the entire image (right) [67]. 16

Figure 20 – Architecture of HEVC Intra coding (encoder), based in [75]. 16

Figure 21 – Example of CTU partitioning (left) [76] and corresponding quadtree (middle) [76]; Modes and orientations for intra-picture prediction (right) [73]. 17

Figure 22 – MOS results [80] for test images TE00, TE03, TE04, TE08, TE16, TE20, TE25, TE39 (Appendix A), for performance evaluation of the FRICwNN, FactMSE, FactMS-SSIM, HyperMSE and HyperMS-SSIM solutions. 20

Figure 23 – Testing results on the Kodak dataset of the: ArHPMSE (as “Our Method” and “Our Method (opt. for MSE)”), using PSNR (left), and ArHPMS-SSIM (as “Our Method (opt. for MS-SSIM)”), using MS-SSIM (right) [83]. 21

Figure 24 - Testing results on the Kodak dataset of the: ContAdMSE (as “Proposed (optimized for MSE)”) and ContAdMS-SSIM (as “Proposed (optimized for MS-SSIM)”) solutions, using PSNR (left) and MS-SSIM (right) metrics [84]. 21

Figure 25 – JPEG-AI learning-based image coding framework [1]. 22

Figure 26 – Image classification results of finetuning the compression (coding) network and of finetuning both coding and image classification networks [90]. 26

Figure 27 – Semantic segmentation performed by ResNet-50-segm and cResNet-51-segm at each operating point [90] for some image (left); Semantic segmentation results of finetuning the compression (coding) network and of finetuning both coding and sematic segmentation networks [90] (right). 28

Figure 28 – Architecture of the scalable facial image compression [102] solution. 30

Figure 29 – Rate-accuracy face verification performance obtained for the proposed solutions and for the CTA JPEG and JPEG2000, over the LFW dataset (left) [102]; rate-distortion performance and reconstructed images obtained for the proposed solutions and for the CTA JPEG and JPEG2000 (right) [102]. 31

Figure 30 – Original anchors architecture for the compressed domain face verification model. 34

Figure 31 – Decoded anchor architecture for the compressed domain face verification model. 35

Figure 32 – Face Detection Pipeline. 36

Figure 33 – Examples of the processing performed when multiple faces are detected for the initial threshold 0.985. In the top, the center of the image is contained in multiple facial areas. In the bottom, the center of the image is only inside one facial area. Images belong to the LFW dataset. 38

Figure 34 – Image processing pipeline with an output image with dimension 224x224x3. Example image retrieved from the LFW dataset. 38

Figure 35 – Framework illustration of the ResNet50 final architecture for face verification. Dashed lines represent downsampling paths of the residual blocks input which is added to the residual block’s first residual unit output. 40

Figure 36 – Illustration of the pre-processing for training the ResNet50. Face image obtained from the VGGFace2 dataset [119]. 41

Figure 37 – Framework illustration of the ResNet50 final architecture with the extra FC layer for fine-tuning the 2048-d FC layer with the ArcFace Loss. (Green) ResNet50 FC layer to be updated. (Red) Extra FC layer for fine-tune with ArcFace Loss. Dashed lines represent downsampling paths of the residual blocks input which is added to the residual block’s first residual unit output. 42

Figure 38 – Illustration of the decision making process given a threshold of 0.4550 and two pairs of faces. One pair has the two images with the same identity (left) and the other pair has two images with different identities (right). The pairs were retrieved from the LFW dataset testing pairs and use the original images from the LFW dataset. 45

Figure 39 – Illustration of the ROC curve, the AUC and the ERR metrics [127]. In face verification, True Positive Rate (TPR) is the same as True Acceptance Rate (TAR) and False Positive Rate (FPR) is the same as False Acceptance Rate (FAR). 46

Figure 40 – Face verification results for all anchors (ResNet50) for the different mean rate for the test images of the LFW dataset with the LFW2 protocol (6k pairs). (a) TAR@FAR=$10-3$; (b) TAR@FAR=$10-2$; (c) ERR; (d) AUC. 49

Figure 41 – Similarity Scores distribution for the LFW dataset (6k pairs). Red: Negative labels (False); Green: Positive labels (True). (left) Decoded anchor with Balle_2018 codec, MSE loss and quality 1; (middle) Decoded anchor with Balle_2018 codec, MSE loss and quality 8; (right) original anchor. 49

Figure 42 – Compressed domain face verification solution. 53

Figure 43 - Illustration of the latent representation pre-processing for training the ResNet39. Face image obtained from the VGGFace2 dataset [119]. 54

Figure 44 - Framework illustration of the cResNet39 final architecture for face verification. N corresponds to the number of channels of the input latent representation, this value differs between the codec models. Dashed line represents the downsampling path of the residual block input which is added to the residual block's first residual unit output. 56

Figure 45 – Face verification results for all anchors (ResNet50) and novel compressed domain solutions (cResNet39) for the different mean rate for the test images of the LFW dataset with the LFW2 protocol (6k pairs). (a) TAR@FAR=$10-3$; (b) TAR@FAR=$10-2$; (c) ERR; (d) AUC. 61

Figure 46 – Image reconstruction testing results on the LFW dataset, using the images included in the LFW2 protocol probe images, for the codecs JPEG (Green), JPEG2000 with MSE optimization (Red), JPEG2000 with visual optimization (Purple), HEVC-Intra (Grey), Balle_2018 with MSE optimization (Orange), Balle_2018 with MSE optimization with target qualities 1, 3, 6 and 8 with face crop, align and resize after the decoder, as in Section 5.3 (Blue): (left) MS-SSIM; (right) PSNR. All solutions perform face crop, align and resize before the encoder, as in Section 6.4, with the exception of the solution with the blue line. 62

Figure 47 – Selected images from JPEG AI database: TE00 (left), TE03 (middle) and TE04 (right) [80]. 80

Figure 48 - Selected images from JPEG AI database: TE08 (left) and TE16 (right) [80]. 80

Figure 49 - Selected images from JPEG AI database: TE20 (left), TE25 (middle) and TE39 (right) [80]. 80

List of Tables

Table 1 – Description of the ResNet and cResNet-k architectures [90]. 23

Table 2 – Image classification accuracies for the ResNet-50-original (original image), ResNet networks (reconstructed domain representations) and cResNet networks (compressed representation) over the target bitrates, created from the data available in [90]. 25

Table 3 – mIoU performance for the ResNet-50-segm-original (original image), ResNet-segm networks (reconstructed domain representations) and cResNet-segm networks (compressed representation) over the target bitrates, created from the data available in [90]. 28

Table 4 – Structure of the ResNet50 architecture. Each residual block is composed of residual units that contain 3 convolutional layers each, e.g., residual block 1 has 3 residual units. 39

Table 5 – LFW dataset information. 47

Table 6 – Structure of the ResNet50 and cResNet39 architectures. Each residual block is composed of residual units that contain 3 convolutional layers each, e.g., residual block 1 has 3 residual units. N is the number of channels of the latent representation, which varies between codec solutions. 55

Table 7 – Computational complexity and face verification performance results on the LFW dataset (LFW2 protocol) of the ResNet50_Original_Anchor, ResNet50_Balle2018_mse_Qn and cResNet39_Balle2018_mse_Qn solutions, for n in (1, 3, 6, 8). Computational complexity is measured with number of operations in GMAC and in kMAC/px (per pixel) and with computational time in s/batch and in s/pair. Each batch contains a total of 64 pairs which corresponds to 128 images. The face verification performance is measured with the TAR@FAR=$10-3$. (*) the solutions' computational times are mean values obtained by repeating the test process 50x with the NVIDIA RTX 3080 GPU [133]. 61

List of Acronyms

Adam	Adaptive Moment Estimation
AI	Artificial Intelligence
ASPP	Astrous Spatial Pyramid Pooling
ATC	Analyse-then-Compress
AUC	Area Under the Curve
Balle_2018	Ballé 2018 with hyperprior
BN	Batch Normalization
bpp	Bit Per Pixel
CABAC	Context-based Adaptive Binary Arithmetic Coding
CEL	Cross-Entropy Loss
Cheng_2020	Cheng 2020 with attention
CNN	Convolutional Neural Network
ContAd	Context-Adaptive entropy model for end-to-end optimized image compression
COS	Cosine Similarity Score
cResNet39_8631-d	ResNet39 model with a FC layer with 8631-d output
CTA	Compress-then-Analyse
CTU	Coding Tree Unit
CU	Coding Unit
DCT	Discrete Cosine Transform
DL	Deep Learning
DSIS	Double Stimulus Impairment Scale
DWT	Discrete Wavelet Transform
EBCOT	Embedded Block Coding with Optimized Truncation
ERR	Equal Error Rate
FAR	False Acceptance Rate
FCNN	Fully Connected Neural Network
FFNN	Feed-Forward Neural Network
FLOP	Floating Point Operation
FNR	False Negative Rate
FPR	False Positive Rate
FRICwRNN	Full Resolution Image Compression with Recurrent Neural Networks
FRR	False Recognition Rate
GAN	Generative Adversarial Network
GD	Gradient Descent
HEVC	High Efficiency Video Coding
HOG	Histogram of Oriented Gradients
ILSVRC	Large Scale Visual Recognition Challenge
IoU	Intersection Over Union

JAHP	Joint Autoregressive and Hierarchical Priors model
JPEG	Joint Photographic Experts Group
LFW	Labelled Faces in the Wild
MAC	Multiply-Accumulate
mIoU	Mean Intersection Over Union
ML	Machine Learning
MOS	Mean Opinion Scores
MSE	Mean-Squared Error
MSE	Mean Square Error
MS-SSIM	Multi-Scale Structural Similarity Index Measure
NN	Neural Network
PSNR	Peak-Signal-to-Noise Ratio
PU	Prediction Unit
R-CNN	Region Based Convolutional Neural Networks
RCT	Reversible Component Transform
ReLU	Rectified Linear Unit
ResNet	Residual Network
ResNet50_8631-d	ResNet50 model with a FC layer with 8631-d output
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristics
RoI	Region of Interest
RoIAlign	Region of Interest Alignment
RPN	Region Proposal Network
RQT	Residual Quadtree
SGD	Stochastic Gradient Descent
SSL	Self-Supervised Learning
SVM	Support-Vector Machine
TAR	True Acceptance Rate
TPR	True Positive Rate
TU	Transform Unit
URQ	Uniform Reconstruction Quantization
VVC	Versatile Video Coding

1. Introduction

This chapter introduces the context and motivation for this MSc Thesis entitled “Compressed Domain Visual Analysis”. This chapter also defines the objectives and structure of this report.

1.1. Context and Motivation

Due to recent advances in technology image applications are used by a very large percentage of the world’s population [1]. Several popular applications rely on image transmission and storage such as in entertainment applications, mainly in social media, and in professional contexts, such as visual monitoring surveillance systems. Since image resolution and quality are increasing, images are being represented with larger amounts of data, which consequently increases the storage cost and transmission delay. Also, most image related applications require image transmission between devices, often in bandwidth limited connections, leading to the need to efficiently compress this information.

Image coding solutions focus on representing the image as much compact as possible by exploiting spatial and statistical redundancies. These solutions can be lossless (decoded information is mathematically the same as the original) or lossy (some information is lost in the coding process). However, lossy compression is more popular especially since many applications do not require perfect mathematical reconstruction. Usually, lossless coding is used in applications where the loss of information can have a negative impact, e.g., medical image analysis. Lossy coding tries to achieve a good trade-off between compression rate and perceptual quality, often considering the human visual system.

There have been some advances throughout the last thirty years in lossy conventional image coding solutions, such as JPEG, JPEG2000, HEVC Intra and VCC Intra. These standards employ lossless techniques to exploit the spatial and statistical redundancies and lossy techniques that exploit the visual relevance (often with quantization) of the image. The spatial redundancy is usually exploited by the transformation of the original image or residue information (obtained by intra-picture prediction) into frequency domain coefficients. The coefficients can then be used to reduce the visual redundancies by discarding information that is represented in less sensitive frequencies of the human visual system.

In recent years, machine learning has gained a lot of popularity in the computer science community, providing efficient solutions for a wide set of applications, like those shown in Figure 1 (left). It has already been shown in several works that machine learning tools, especially deep-learning solutions, have high efficiency in the representation of visual information. This is one of the reasons for the success of deep-learning supervised techniques in image analysis tasks, as shown in Figure 1 (right). Deep learning has also been successful for many image processing tasks based on unsupervised learning, including super-resolution, denoising, low-light enhancement, colour correction, exposure compensation and inpainting. Thus, learning-based image coding solutions are now being study by several standardization groups, namely by the JPEG AI project [1] in the JPEG standardization group.

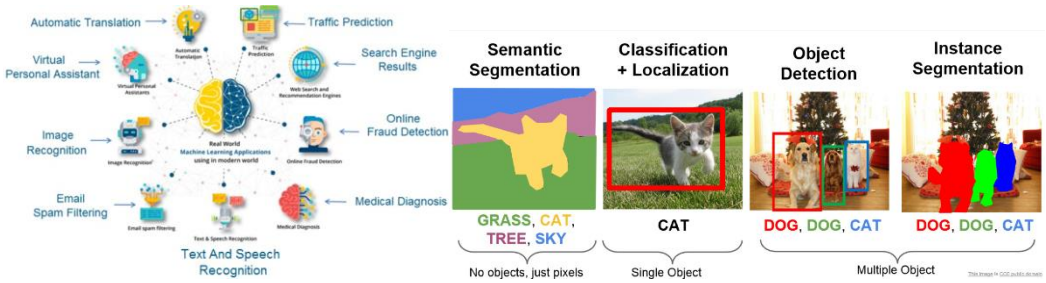

Figure 1 – *Examples of machine learning real-world applications [2] (left); example of deep-learning image applications [3] (right).*

Traditionally, image processing and computer vision tasks are performed over lossy decoded images since often it is not possible to obtain the original image. This can lead to lower performance, since usually a lossy and thus with artifacts representation of the image is used.

However, these tasks can also be performed over the compressed representation. In this case, the usual decoding process is not performed entirely and an intermediate representation (usually obtained after entropy decoding and referred as latent representation) is exploited for both image processing and computer vision tasks, as well as for decoding [1]. This means that features that were extracted while coding the original image are used instead of the lossy decoded image. Moreover, since these solutions are applied on the latent representation, they are capable of performing these tasks with lower complexity and storage cost. In conclusion, this new way of performing learning-based image processing and/or computer vision in compressed domain can have a significant impact in performance in a wide range of applications, like in cloud storage, surveillance systems, autonomous vehicles, image collection storage and management, live monitoring of visual information, media distribution, etc. [1].

1.2. Objectives and Structure

One of the objectives of this report is to review some of the available literature in image coding using conventional and deep learning principles as well as image analysis in the compressed domain. The main goal in this MSc Thesis is to design, implement and assess a novel compression domain face verification solution, ideally with increase in performance when comparing to the available literature. To achieve the defined goals, this report will be structured as followed:

- **Chapter 2 –** review of neural networks basic concepts and architectures, and relevant deep-learning image analysis solutions.
- **Chapter 3 –** review and evaluation of conventional and state-of-art deep-learning image coding solutions.
- **Chapter 4 –** review and evaluation of relevant compression domain deep-learning image analysis solutions in image classification, semantic segmentation and face recognition.
- **Chapter 5 –** description of the design, implementation and evaluation of the selected anchors solutions for face verification, to be used as baseline results for the novel compressed domain face verification model.
- **Chapter 6 –** description of the design, implementation and assessment of the proposed compressed domain face verification model.
- **Chapter 7 –** report summary and presentation of conclusions of this MSc Thesis and future work suggestions.

2. Deep-learning Image Analysis: Foundations and Architectures

This chapter reviews the most relevant Deep-Learning (DL) image analysis solutions. Moreover, the main concepts and architectures in DL technology for the analysis of images are presented.

2.1. Neural Network Basics

Artificial Intelligence (AI) is any method that allows computers to mimic human behaviour [4]. Machine Learning (ML) is a subset of AI, which can be defined as the computer ability to learn without explicitly being programmed [4]. ML has different areas, such as deep-learning, one of the most popular. DL algorithms are based on artificial neural networks (NN) with the purpose of solving complex machine learning problems in an efficient way. DL is widely used in image processing and computer vision applications since it allows to extract important features and perform classification on multiple classes over relevant input data.

Basic concepts

Any neural network consists of artificial neurons or nodes that are inspired by the way neurons work on the human brain [5], as can be shown in Figure 2. The nodes transmit information between them, usually in a certain flow, just like in the biological brain where information is transmitted between neurons through synapses [6].

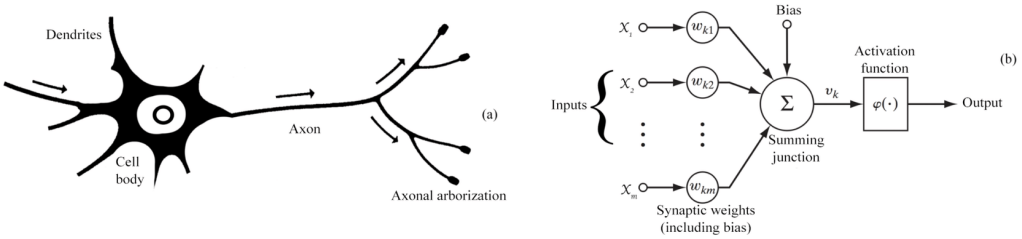

Figure 2 – *Biological neuron (left) and artificial neuron (right) [7].*

The computational structure of an artificial neuron is rather simple, each neuron receives a set of inputs ($\boldsymbol{x_i}$) which are linearly multiplied by their corresponding weights ($\boldsymbol{w_{ki}}$). These weights represent the impact that each input will have on the output of the neuron and will be adjusted (learned) to obtain the best fit for the desired task. The neuron can represent any linear combination of the inputs and thus, the weighted inputs are all added together with a bias ($\boldsymbol{b_k}$). The output of the summing joint ($\boldsymbol{v_k}$) will be the input of the activation function ($\boldsymbol{\varphi_k}$) that will compute the output of the neuron ($\boldsymbol{y_k}$). The mathematical representation of an artificial neuron can then be represented by its output ($\boldsymbol{y_k}$) as defined in (1).

$$y_k = \varphi_k\left(b_k + \sum_i w_i\, x_i\right) \qquad (1)$$

The activation function φ_k is typically non-linear, being the neuron's component that allows it to have a non-linear response to the input, thus allowing the neuron to represent more complex functions. There is a large range of activation functions, being the most used the following:

- **Rectified Linear Unit** (ReLU) – output corresponds to the maximum between 0 and the input.
- **Logistic/Sigmoid Function** – output between 0 and 1, according to the function shown in Figure 3.

- **Hyperbolic Tangent Function** (tanh) – output between -1 and 1, according to the function shown in Figure 3.
- **Softmax Function** – activation function used in the output layer of neural network to obtain probability distribution over a discrete variable with n possible values and thus, often used for multi-class classification problems. The output is contained between 0 and 1.

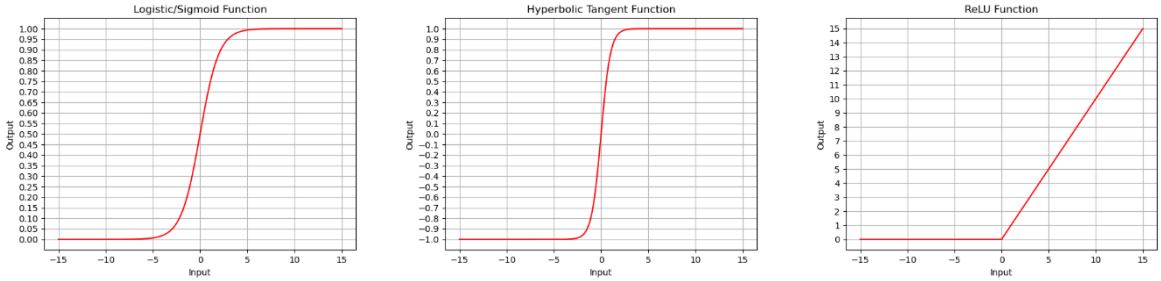

Figure 3 – *Graphical representation of the Logistic/Sigmoid (left), Hyperbolic Tangent (middle) and the Rectified Linear Unit (right) functions, with unidimensional input.*

Neural network architecture

A NN is simply a combination of artificial neurons all connected in a graph like structure, that is usually acyclic. In the highest structural level, the NN is composed by layers, which are blocks of neurons, usually not connected with each other. Typically, a layer is uniform, i.e., the neurons have the same characteristics, for example all of them use the same activation function [8]. The depth of a NN corresponds to the number of layers. A fully connected layer is created when all the neurons of one layer are connected to all the neurons of the next layer, like in Figure 4.

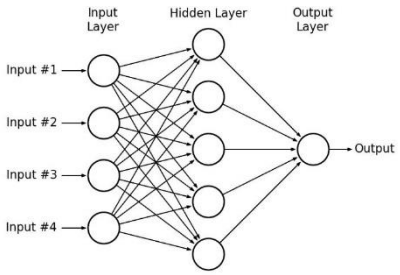

Figure 4 – *Representation of an example of a Fully Connected Neural Network (FCNN) with one hidden layer [9].*

The layers in a NN are divided in three types. The input layer, which receives the input data, pre-processed or not, followed by the hidden layers which are located between the input and output layers, as shown in Figure 4. The number of hidden layers may be zero to as many as needed to represent the input data in an efficient way. The output layer is the last layer of the NN, it receives the processed data from the input layer or from the last hidden layer and generates the result in the expected form, according to the problem being solved, e.g., in a linear regression problem the output would be the parameters of the line: bias and slope. When the data flow is oriented from the input layer to the output layer without any cycles, an acyclic graph is formed and thus, the NN is denominated as Feed-Forward Neural Network (FFNN). Otherwise, the NN is a Recurrent Neural Network (RNN) and cycles can occur. RNNs are distinguished by their memory capability as they retrieve prior information to influence the current input and thus, the output of each neuron [10].

The concatenation of multiple layers defines the architecture of a neural network. There are many possible architectures that differ from the number of layers (depth of the NN), the number of connections between layers, the number of neurons (size of the NN), etc. The oldest and most straightforward neural network is a non-recursive fully connected neural network (FCNN) which only contains fully connected layers, which means that each neuron in a layer will be connected to all neurons on the previous layers, as shown in Figure 4. Nowadays, with the evolution of neural networks, FCNNs are one of the simplest types of NN, being mostly used as a component in more complex architectures.

Model Training

Usually, for the training (i.e. to learn a suitable NN model) a large dataset is used, which is typically is divided into training set, validation set and testing set. There are different learning methods that can be applied, supervised methods, unsupervised methods and self-supervised methods. Supervised methods use labelled examples where the ground truth is known, with the purpose of finding the best relationship between the input and output (label) data. Unsupervised methods use unlabelled examples, it aims to "discovers hidden patterns or data groupings without the need for human intervention" [11]. Self-supervised learning (SSL) methods also do not require humans to perform data labelling, often exploiting the data structure. The main idea of SSL is to predict hidden parts or properties of the input from the observable parts [12].

Before training, a set of hyperparameters are defined which will influence the learning process and the overall performance of the neural network model. Hyperparameters are variables that define both the network structure, e.g., number of hidden units, and how the network is trained, e.g., learning rate [13]. The training of a neural network is based on iterative methods to find the model's weights that best fit the problem, that is represented by the data set. The most used methods are the Gradient Descent (GD) method and others derived by gradient descent, which can be summed by the flowchart in Figure 5.

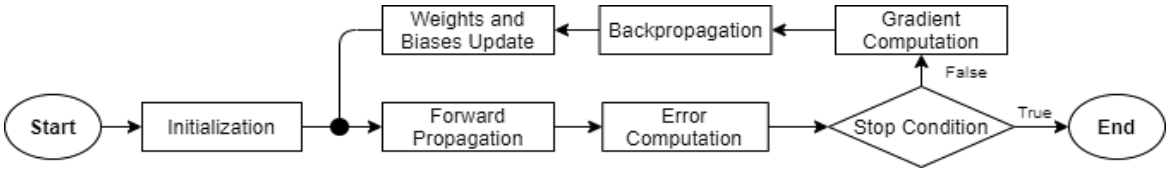

Figure 5 – *Simplified Gradient Descent algorithm flowchart for the training of a neural network.*

In the start of training, the weights and biases (parameters) are random initialized or already defined from another training session (pre-trained models). The input data is then propagated throughout the network and all the processing leads to an output. The error between the output and the ground truth is calculated by a loss or cost function. The selected loss function depends on the problem being addressed. The gradient of the loss function in respect to the model parameters ($\frac{\partial L}{\partial w_{ki}}$ and $\frac{\partial L}{\partial b_{ki}}$) is computed, which defines the direction and rate (amount) that the model parameters must change to minimize the loss function. Backpropagation is then applied to propagate the error backwards throughout the network. By computing an error in each neuron it is possible to adjust the parameters of the entire model [14]. For the gradient descent, the weights (w_{ki}^{j+1}) are updated using the weights (w_{ki}^{j})

and the gradient in respect to the weights ($\frac{\partial L}{\partial w_{ki}^{j}}$) of the previous iteration; a learning rate parameter (η) is also used to control the deviation caused by the gradient, as defined in (2). The choice of the value of η must be done carefully. By increasing the value of η the algorithm may converge faster to the solution but at the risk of overshooting the minimum value of loss function. By decreasing the value of η, the algorithm takes more time to reach the minimum (long training sessions) [15]. Some solutions use variational learning rates to better adapt to the needs at different stages of the training, e.g., higher learning rates are used in the beginning of training followed by a decay process as the algorithm comes closer to the desired minimum.

$$w_{ki}^{j+1} = w_{ki}^{j} - \eta \frac{\partial L}{\partial w_{ki}^{j}} \qquad (2)$$

2.2. Main Neural Network Types

There is a large amount of neural network types, each with different characteristics, processing steps, and applications. Therefore, considering the objective of this MSc Thesis, a small set of neural network types was selected and are described in this Section.

2.2.1. Convolutional Neural Network

Convolutional neural networks (CNN) have shown excellent performance in several visual analysis tasks [16]. A CNN applies filters to an image or a set of images to capture spatial and/or temporal dependencies [17]. The first layers have the role of extracting low-level features from the data, like colours and edges, but as it advances throughout the layers, medium or high-level features are extracted, which is rather suitable for many computer vision and image processing problems, such as image classification, recolouring, etc. The CNN is composed by four different types of layers: input, convolutional, pooling and fully connected, as shown in Figure 6. Classification is performed on the fully connected layer in which the softmax activation (calculates a probability for every possible class) is applied. The number of convolutional and pooling layers and their parameterization (and thus the CNN architecture) varies with the problem being addressed.

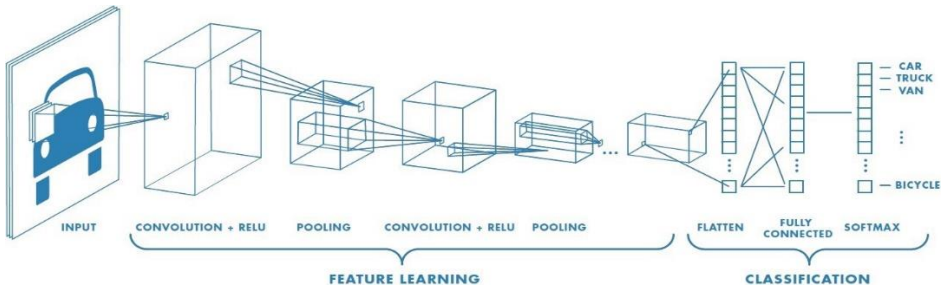

Figure 6 – *CNN architecture example with an image input, two convolutional layers, two pooling layers and one fully connected layer [17]. In this case, after every convolution a non-linear activation is used.*

Convolutional Layer

The convolutional layers are the main layers of the CNN, being responsible for extracting features from the input image. Convolutional layers perform convolution of the input data with filters or kernels. This operation corresponds to the dot product between the filter kernel and a window of the input (receptive field). The process is then repeated to different windows (spatial locations) and the

combination of the filtered outputs corresponds to the feature or activation map. This process allows to detect different features according to the kernel being used. During training, the kernel's weights are updated through backpropagation and gradient descent mechanisms.

When processing images, the input is a 3D array representing the contribution of three colour components for each pixel, typically RGB. A 2D feature map will result from applying one kernel, with the same depth of the input, to the image, since each value is a linear combination of the filtered outputs for all depth levels within a receptive field. This means that an image with dimensions WxLxD, where W and L are the spatial dimensions and D is the depth, requires kernels with dimension SxSxD, e.g., for an 32x32x3 RGB image the kernel 5x5x3 (of size S) can be applied, as shown in Figure 7 (left).

The size of the kernel is smaller than the size of the input, usually with dimension of 3x3 or 5x5. After filtering some location in the input, it moves on to the next position according to the predefined *stride*, which corresponds to the number of positions that the filter skips before the next placement of the window. The increase of the stride leads to a decrease in the size of the feature map. When the filter passes the limits of the input matrix it is necessary to perform *padding*, i.e., expanding the dimensions by setting to zero the elements that fell outside the border [16]. In Figure 7 (right) an example of a convolution operation with a kernel of 3x3 can be observed, with stride of 2, over a 6x6 input array for the first two outputs.

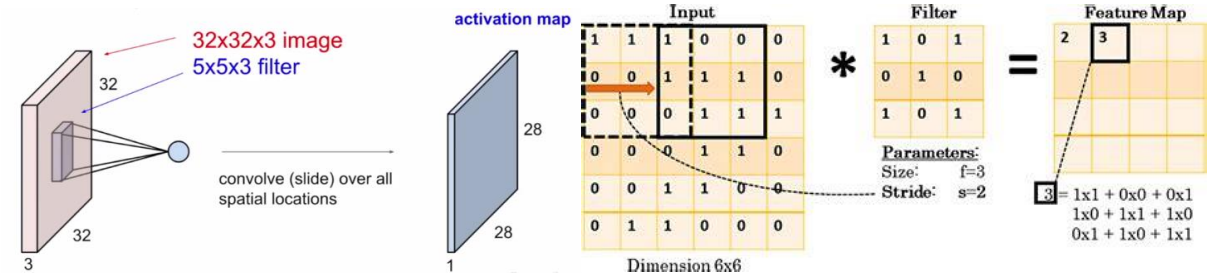

Figure 7 – *Example of the resulting feature/activation map by applying one kernel/filter over an input (lef) [18], representation of convolution operation in the first two windows with stride (right) [19].*

A convolutional layer is composed by a set of learnable kernels (filters) in parallel that generates a feature map for each one, producing an output with depth equal to the number of kernels used. Typically, after convolution the feature map is processed by a ReLU unit, introducing non-linearity to the model.

Pooling Layer

The pooling layers are the down sampling components of the CNNs and are typically inserted after the convolutional layers. The dimensionality reduction of the feature map (input) is achieved by using filters in a similar way as in convolutional layers. The difference is that these filters do not have any weights, instead they have an aggregation function associated. There are different types of pooling methods according to the aggregation function used, i.e., the operation executed over the receptive field when the filter moves slides over the input. The most common types are the average pooling, the average value within the receptive field is calculated and sent to the output, and the max pooling, the maximum value in the receptive field is used as output instead. Both methods are illustrated in the example shown in Figure 8.

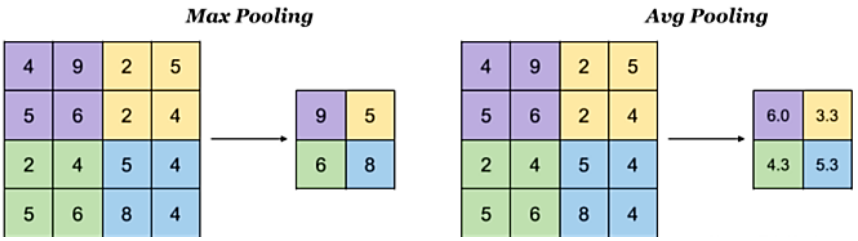

Figure 8 – *Max pooling and average pooling examples, with filter of size 2x2 and stride 2 [20].*

2.2.2. Autoencoder

Autoencoders are feed-forward neural networks that learn to represent data in a compact way. Thus, autoencoders can reduce the dimensionality of the data efficiently, by learning how to ignore noise and irrelevant information [21]. These NNs are used in image compression, image denoising, image colorization and image segmentation.

Autoencoders have three main components: the encoder, the decoder and the bottleneck layer which separates the encoder from the decoder. The bottleneck layer forces the encoder to lower the dimension of the input and outputs the latent representation. The encoder learns to exploit the dependencies in the input data to produce more compact representations. The decoder learns to transform the latent into an output that resembles as much as possible the input data [22]. The decoder and encoder layers can either be fully connected or convolutional layers. An architecture example with fully connected layers is shown in Figure 9 (left) as well as a general representation of the structure of an autoencoder (right).

This type of NN uses unsupervised (or self-supervised) learning methods, since a reconstruction loss (difference between input and output) is used to drive the computation of the weights during training. Therefore, the loss function has a significant influence in the performance of the autoencoders and usually defines the distortion (or quality) between the original and output data for the case of images.

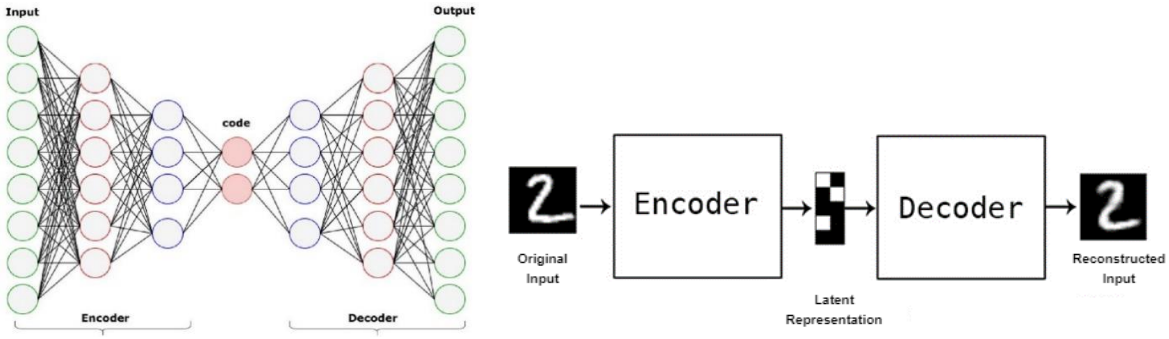

Figure 9 – *Autoencoder architecture example with fully connected layers (left)) [23] and example of a general representation of an autoencoder (right) [21].*

2.2.3. Generative Adversarial Network

Generative models create new data samples that resemble the data used for training. They learn the underlying pattern of the input to generate similar content [24]. Generative Adversarial Network (GAN) is a generative model that uses two NNs (generator and discriminator) that compete against each other. An illustration of the structure of a GAN is shown in Figure 10. The generator learns to create new data (fake data) to deceive the discriminator, which learns to detect the fake data from the training data (ground truth). GANs are an exciting innovation that have shown good results, as stated by Ian

Goodfellow, creator of the GAN architecture, in his paper: "…we believe that these samples are at least competitive with the better generative models in the literature and highlight the potential of the adversarial framework." [25].

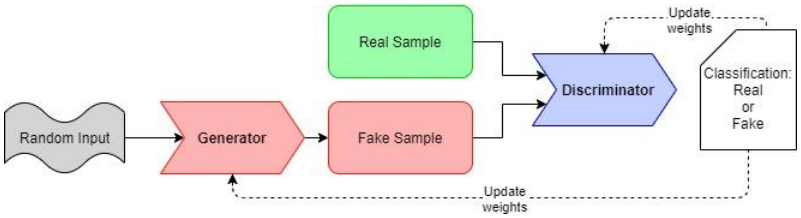

Figure 10 – *General architecture of a GAN.*

The discriminator is a classifier that indicates if the samples computed by the generator are fake or real. While training, the discriminator classifies the samples and an adversarial loss value is obtained. The weights of both generator and discriminator are then updated through backpropagation by using this adversarial loss. This means that a discriminator model is found that correctly classifies real data and fake data, while a generator model is found to generate realistic fake data. Moreover, the generator learns to create plausible (yet fake) data that resembles as closely as possible to real data and thus can fool the discriminator.

The random input shown in Figure 10 in its simplest form is just noise, usually with a normal distribution; in practice, this distribution does not have a strong impact in the results [26]. When training, the generator generates fake samples to the discriminator which classifies them, enabling to calculate the adversarial loss value. Training a GAN is an alternating process, since only one component can be trained at once. Both NNs start with random weights and usually the training of the discriminator is the first to start. The main idea is to train one component for a set of data samples and then train the other for a set of samples of the same size. This process is repeated to fully train the GAN. Initially, the generator fails in fooling the discriminator with the fake data. But as the training goes by, the generator starts producing more realistic data, until reaching the point where the discriminator can no longer differentiate real from fake [27]. Consequently, the discriminator's accuracy becomes worse and worse, until it reaches the point of 50% probability of getting it right. This imposes a problem, since the discriminator feedback becomes less meaningful, the generator will be training with bad feedback, which will worsen its quality [28].

2.3. Reviewing Some Relevant Deep-learning Image Analysis Solutions

Nowadays, there are several image analysis problems that exploit deep learning foundations techniques to achieve very high levels of performance. In this Section some image analysis problems are described along with some relevant state-of-art solutions.

2.3.1. Image Classification

As defined in [29], image classification is the process of identifying the category to which an image belongs. In the past, several solutions have addressed this problem, such as convolutional neural networks which have been rather successful [30].

Different architectures have been designed, implemented and evaluated through the years, mostly between 2012 and 2016 due to the ImageNet's Large Scale Visual Recognition Challenge (ILSVRC) competition. LeNet-5 [31] was the first neural network (in 1998) that exploited CNN

architectures, rather successfully for handwritten digit recognition. AlexNet [32] came after, solving some issues of the LeNet-5 model, introducing the ReLU activation function in the convolutional layers and exploiting max polling instead of average poling. AlexNet has won the ILSVRC'2012 challenge and marks the beginning of a period of high interest in deep learning applications that still persists today. Later, in 2015, it was proposed the Residual Network (ResNet) [33] by Microsoft represents a significant advance in CNN's architectures. In this Section, only the ResNet model is reviewed since it represents a successful state-of-the-art method in image classification.

ResNet architectures are mostly composed of convolutional layers with 3x3 filters (3x3 conv), with the exception of a global average pooling layer and a fully-connected layer with softmax at the end of the network. The 3x3 conv layers are designed such that the computational complexity between layers is fixed, hence the number of filters is doubled whenever the output of the layer has half size of the input. This occurs due to downsampling performed by 3x3 conv layers with stride 2.

In previous deep neural networks it was noticed that lower accuracy was obtained when the number of layers started to increase, leading to models with a higher training error and thus with no performance gains. This was happening due to a phenomenon referred as vanishing gradients [34], [35].

ResNet mitigates this problem by using shortcuts or skip connections in the network. These skip connections perform identity mapping, and their outputs are added to the outputs of the stacked layers [37], like shown in Figure 11 (left). The stacked layers, inside the shortcut area, learn a residual mapping F(x) (equal to H(x)-x) instead of directly learning the desired underlying signal H(x) (equal to F(x)+x), which is usually more difficult to model for deep neural networks. In [33] it was shown that is easier to optimize the residual mapping than to optimize the mapping without the skip connection.

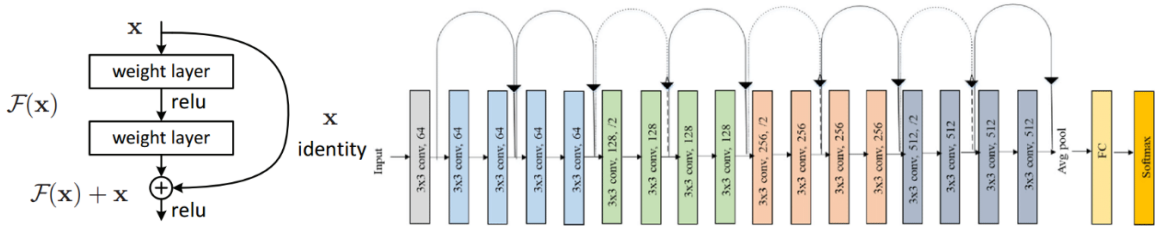

Figure 11 – *Building block example of a Residual Network (left) [33], ResNet-18 architecture (right) [37].*

In Figure 11 (right), it is presented an example architecture of a ResNet (ResNet-18). In [38] it is proposed a Residual Attention Network, a more complex model, that uses stacked attention modules to compute attention-aware features. These networks provide some improvements in performance by exploiting different levels of attention [38].

2.3.2. Object Detection

Object detection [39] is the process that aims to detect and classify instances of objects from a large range of known classes. CNNs have been used to perform such task, due to their capability in extracting features from an image. Region Based Convolutional Neural Networks (R-CNNs) are a popular model for object detection. R-CNN architectures are mainly divided in three phases:

1. Attempts to find image regions (region proposals) with potential objects.

2. From these regions, features are extracted by using CNNs.

3. The extracted features are used to classify the detected objects.

R-CNN

The R-CNN system [40] is composed of three parts, like previously described and as shown in Figure 12. Initially, it produces region proposals through a selective search Edge Boxes [41] inspired algorithm; these boxes are the candidate detections set. These regions are cropped out from the original image, resized and sent to a large CNN which aims to extract fixed-length feature vectors for each region [40]. Category-specific linear Support Vector Machines (SVMs) [42] are then trained from the extracted features to classify each region and to refine the region proposal bounding boxes [43].

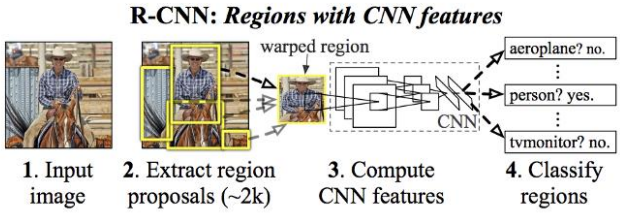

Figure 12 – *Overview of the object detection system of the R-CNN [40].*

Fast R-CNN

Fast R-CNN [44] seeks to improve the R-CNN model accuracy and lowers the complexity of accurately locate the objects. Like in R-CNN, this model uses a selective search inspired algorithm to create region proposals. However, Fast R-CNN does not crop these regions before feeding them to the CNN, instead the whole image is processed by the CNN, generating a convolutional feature map. Afterwards, for each region proposal, a fixed-size feature vector is retrieved by a Region of Interest (RoI) pooling layer [44].

These vectors then go through classification and a regression refinement bounding box process, performed by classification and regression layers respectively, like shown in Figure 13. Besides improving accuracy, Fast R-CNNs achieve a better efficiency since the convolutional feature map is computed over the entire input image, instead of independently compute a feature map for each region proposal [43].

Faster R-CNN

Faster R-CNN [45] introduces a deep fully convolutional network to generate region proposals, denominated as Region Proposal Network (RPN), instead of the Edge Boxes algorithm [43] of the previous Sections. The RPNs are designed to be flexible in the prediction of object boundaries, by using anchor boxes [46] at multiple scales and aspect ratios [45]. Faster R-CNN is manly composed by two modules, the RPN and a Fast R-CNN detector that analyses the proposed regions, like shown in Figure 13. The CNN of Figure 13 extracts features from the image that are used for generating regions of interest (RoI) by the RPN and for feature computation. The RoI pooling layer works as in the Fast R-CNN, but for the Faster R-CNN the RoIs are generated by the RPN. This new method of generating region proposals is more efficient and allows for better adjustment of the regions to the input data [43]. The classification and the regression refinement bounding box are then executed as in the Fast R-CNN.

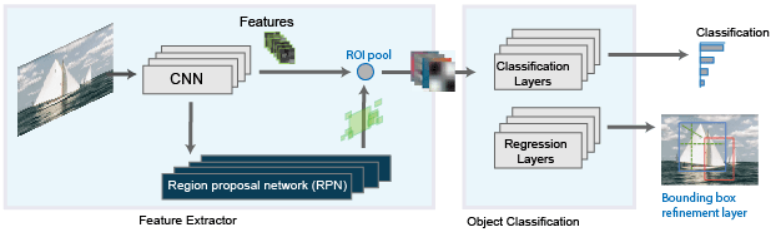

Figure 13 – *Faster R-CNN example architecture [43].*

2.3.3. Semantic Segmentation

Image segmentation is the partition of an image into non-overlapping regions with a semantic meaning considering a specific application. Image segmentation produces a label map, where each pixel of the image is classified into a set of fixed categories, like those shown in Figure 14 (left). Many methods were developed to achieve this goal, and Mask R-CNN is one of the most popular and successful methods using deep-learning principles.

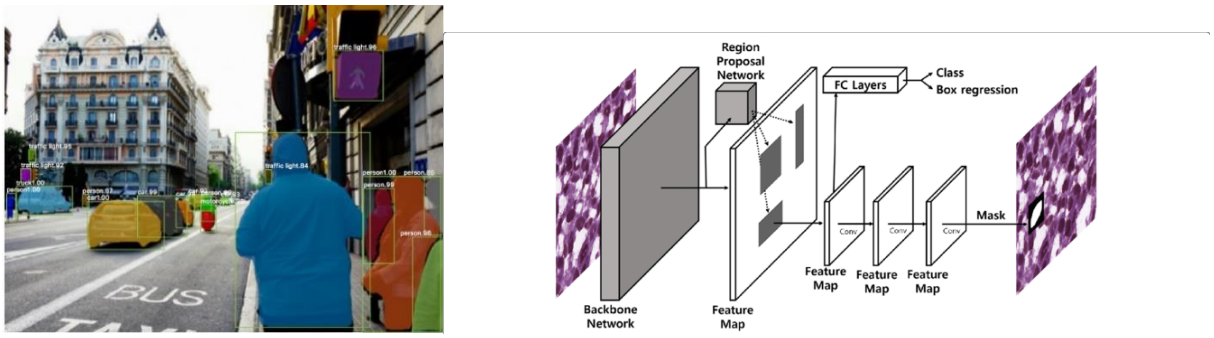

Figure 14 – *Example of semantic segmentation performed by a Mask R-CNN model (left) [48], Mask R-CNN framework for segmentation (right) [49].*

Mask R-CNN [48] is an improvement of the Faster R-CNN such that it can detect the exact boundaries of an object (instead of a bounding box). It achieves this goal by adding a branch to the Faster R-CNN for binary segmentation mask prediction for each region of interest generated by the Faster R-CNN's RPN, classifying if a given pixel belongs to the object or not. It takes as input a CNN feature map and produces a matrix with the value 1 in the pixel locations which belong to the object and 0 in the remaining locations [50]. This branch is a set of convolutional layers applied to each region of interest and it operates in parallel with the classification and bounding box regression component, like shown in Figure 14 (right).

The Mask R-CNN implements another relevant change to the Faster R-CNN for segmentation tasks, the RoIAlign (region of interest alignment). In the Mask R-CNN, the Faster R-CNN's RoI pooling layer is replaced by a quantization-free layer, RoIAlign, that preserves spatial location [48]. This way, the Mask R-CNN mitigates any misalignment between the regions of the feature map chosen by the RoI pooling layer and the original image regions [50].

The Mask R-CNN architecture allows to generate a mask for each class without competition. This means that once the regions of interest are already classified it is sufficient to predict a mask without concern on classification, making the model easier to train [48].

2.3.4. Face Recognition

Face recognition is an automatic method of identifying or verifying the identity of a person using their face. Nowadays, CNNs are often used by face recognition deep learning solutions, e.g., DeepFace by Facebook [51] and FaceNet by Google [52] which achieve a very high level of accuracy. In this Section a system called OpenFace [53] proposed in 2016 is described, which achieves high accuracy with low training and prediction times.

The idea behind face recognition techniques that use deep neural networks can be summarized in Figure 15 (left), being mainly composed by four stages: face detection, pre-processing, feature extraction and classification. The first process to detect a face, since it is necessary to identify the location of the faces in the image in order to process each face individually. There are several available methods that can perform this task, like the CNN for face detection [55] that operates at multiple resolutions, discarding the background information of the image in low resolution and carefully analysing the potential face regions in higher resolutions. The extracted faces can be oriented in different poses which may create a problem for the recognition. To solve it, a pre-processing machine learning algorithm is applied to identify 68 specific points in the faces, denominated as face landmark estimation [53], [56]. An example of the identification of the 68 face landmarks in an image can be seen in Figure 15 (middle). A 2D affine transformation is then applied by using as reference three defined landmarks so that the eyes and nose of all faces are in similar locations, like shown in Figure 15 (right). When performing the transformation, the image is resized and cropped so that the inputs in the NN have a dimension of 96x96 pixels [53].

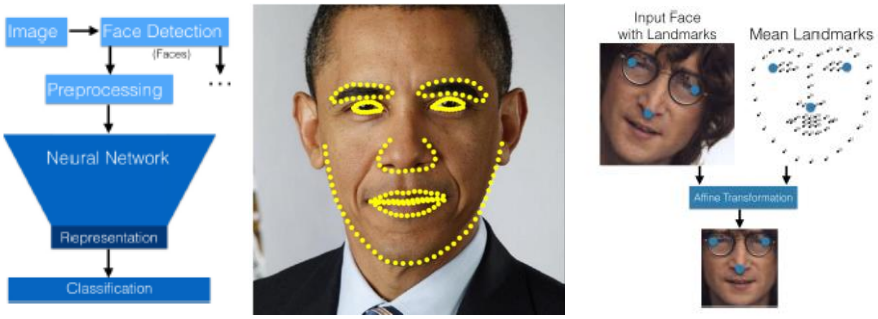

Figure 15 – *Logic of face recognition systems that use NNs (left) [53], example of a 68 face landmark estimations on a real image (middle) [54], example of the affine transformation applied to a detected face in an image by using the blue points as reference (right) [53].*

Now there is the need to extract features from each face to compare them with features extracted from already labelled faces. To do this, OpenFace creators have published several trained networks that can generate these features for any face [56]. These networks are designed based on the deep neural networks used in DeepFace solution [51], composed by a CNN trained for face representation, and by using the FaceNet's triple loss function [52]. After having the features for each face, the classification process is performed by a linear SVM. Important to note that only people that are already in the database can be identified.

3. Image Coding: A Brief Review

Nowadays, there is an increased number of digital images that are shared between applications and users. Moreover, the requirements to represent an image are now more demanding, users are asking for more spatial resolution as well as bit-depth. However, there are consequences in the transmission and storage costs which are now much higher [57]. Therefore, there is the need to obtain even higher compression ratios and thus achieve compressed representations that require less amount of data for some target quality, this means using some lossy compression methods.

In this chapter, a brief review of the most popular conventional methods in image coding and a selected set of deep-learning based image coding solutions is presented. The selection was made to represent the state-of-art in image coding. A short evaluation of several image coding solutions is also presented, especially focused on their rate-distortion performance.

3.1. Conventional Coding

Conventional coding methods are mainly composed by the modules of transformation, quantization, entropy coding and optionally spatial prediction. They try to reduce spatial redundancy and statistical redundancy with hand-crafted methods to efficiently compress the image [58].

JPEG

The JPEG codec is a successful image compression system [58] created by the Joint Photographic Experts Group (JPEG), first developed in 1992. It is still the most widely used lossy image coding solution. Figure 16 shows the processing steps of the JPEG coding architecture, of both encoder (left) and decoder (right) components.

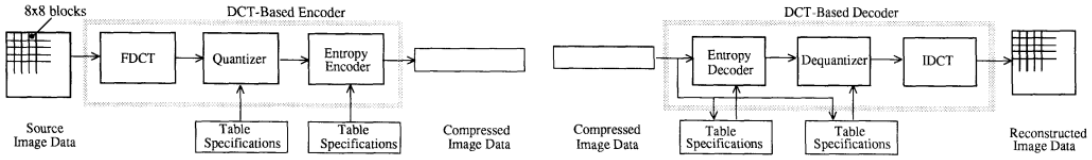

Figure 16 – *JPEG encoder (left) and decoder (right) processing steps [59].*

The JPEG coding architecture is mainly divided in the following sequential processing steps:

- **Block splitting –** Initially, each component of the image (RGB or YCrCb) is divided into 8x8 non-overlapped sample blocks of pixels [59].
- **Discrete Cosine Transform (DCT) –** DCT transform (Figure 17, left) is applied to each block to transform the visual data from spatial to frequency domain. This method exploits the spatial redundancy to compactly represent the block pixels [59], [60], [61]. An example of DCT coefficients is shown in Figure 17 (middle).
- **Quantization –** process used to achieve higher compression, being the step that allows the control of the quality and rate of the coding process. Its main goal is to discard data that is not visually significant [59]. Specific quantization tables are used to eliminate visual data, this means some frequencies will be coarsely represented compared to others. It is more common to use tables that preserve low-frequency coefficients and discard the more high-frequency coefficients; these tables were inspired by the human visual system, which is less sensitive to information loss in those high

frequencies [58]. This step is a lossy procedure since it is a many-to-one mapping, being the principal source of information loss in the encoder [59]. An example of quantized DCT coefficients is shown in Figure 17 (right).

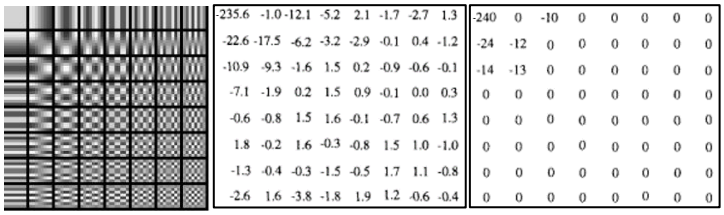

Figure 17 – *DCT's basic functions (left) [60], example of DCT coefficients (middle) [59], example of quantized DCT coefficients (right) [59].*

- **Entropy coding –** lossless coding process which exploits the statistical redundancy. Starts with zig-zag serialization of the quantized DCT coefficients into symbols [59], followed by Huffman coding [62].

JPEG2000

The JPEG2000 image coding solution improves the previously developed standards by providing higher rate-distortion performance and new functionalities [63], [65]. It allows to extract from the bitstream different spatial resolutions, provides random spatial access, lossless compression and the use of ROI coding [63], [65]. It also allows different types of scalabilities and robustness to errors [63], [65], [65]. Figure 18 shows the architecture of the JPEG2000 codec.

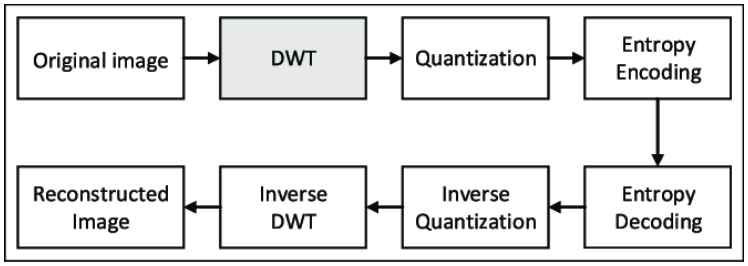

Figure 18 – *JPEG2000 codec architecture [64].*

The processing steps of the JPEG2000 codec can be described in the following sequence:

1. **Tiling –** division of the image into non-overlapping tiles that can be of arbitrary size, usually rectangular [65]. This process allows to reduce memory requirements of the codec. Moreover, since all operations are performed independently to each tile, the decoding process can be done exclusively to some desired parts of the image [67].
2. **Color component conversion –** point-wise decorrelating transformation over the color components [65]. Can either allow lossy reconstruction (e.g. YCrCb transformation) or lossless reconstruction (Reversible Component Transform, RCT) [65]. Usually applied over images with multiple components (e.g., RGB) [65].
3. **Discrete Wavelet Transform (DWT) –** decomposes each tile, or the entire image, into a set of sub-bands organized in different levels [63], [67], [68]. These sub-bands are a set of coefficients that represent the spatial frequency characteristics of the tile, in both vertical and horizontal dimensions [63]. The transform is unidimensional, applied in both dimensions and done independently for low frequencies (low-pass samples) and high frequencies (high-pass samples) [67]. The transform

generates four partitions for each tile, low resolution (LL), high resolution (HH), high vertical resolution and low horizontal resolution (HL) and low vertical resolution and high horizontal resolution (LH) [67]. This process is called dyadic decomposition and is repeated a predefined number of times over the low-resolution block of each tile, like shown in Figure 19 (left and middle) [67]. An example of the dyadic decomposition on a complete image is shown in Figure 19 (right).

Figure 19 – *DWT transformation on a single tile (left) [67], dyadic decomposition on the all set of tiles (middle) [63], example of the dyadic decomposition on the entire image (right) [67].*

4. **Quantization and Partition –** similar to what is done in the JPEG codec, all coefficients are quantized, i.e., reduced in precision [67]. Each sub-band is then partitioned, creating packets that contain continuously improved resolution or quality levels, for each tile [67]. Code blocks are then formed by dividing each packet partition location into regular non-overlapping rectangles [63].
5. **Entropy coding –** performed over each code block independently [63]. A context dependent binary arithmetic decoding algorithm is applied to perform this step [63], denominated by Embedded Block Coding with Optimized Truncation (EBCOT) [69], that reduces spatial redundancy.

HEVC

The High Efficiency Video Coding (HEVC) standard is a result of decades of research in digital video coding [70]. Essentially, HEVC targets to double compression capability while maintaining the same quality [70], [71] and to achieve compression of video content with higher resolutions than the H.264/AVC codec [72], [73]. Although it was developed for video coding, it can also be employed in image coding. In this case, HEVC only uses the Intra mode for prediction, mainly to exploit spatial dependencies within the image [74]. The general processing steps of the HEVC codec for image coding (HEVC Intra) can be seen in Figure 20.

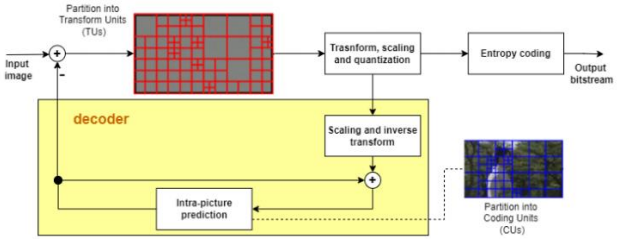

Figure 20 – Architecture of HEVC Intra coding (encoder), based in [75].

In video codecs, the commonly used color representation is the YCbCr colour space in 4:2:0 sampling [73], although others are supported. This ratio indicates that the luma component (Y) has four times more samples than each chroma component (Cb and Cr) [73]. The HEVC Intra codec introduces a flexible and efficient block partitioning structure [76], in which four different block types can be identified:

- **Coding Tree Units (CTUs) –** basic processing units obtained by dividing the image [73], [75].

- **Coding Units (CUs) –** regions with same prediction mode [76], in image coding that can either be intra or none. Results from the partition of the CTUs, by means of a quadtree (coding tree) [73], [76] are shown in Figure 21 (left and middle). Each CU is a leaf node of the coding tree.

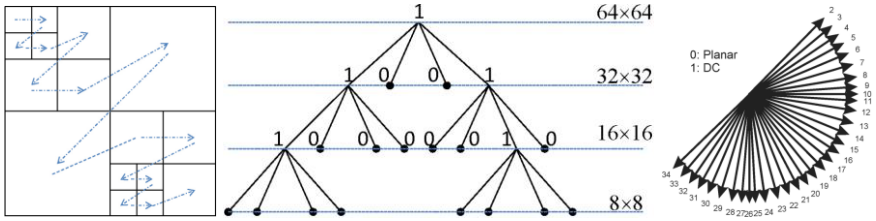

Figure 21 – *Example of CTU partitioning (left) [76] and corresponding quadtree (middle) [76]; Modes and orientations for intra-picture prediction (right) [73].*

- **Prediction Units (PUs) –** regions that share the same prediction information [76]. For each CU, one or more PUs can be specified [76]. For each PU, only essential information obtained from the prediction process goes to the decoder [76].
- **Transform Units (TUs) –** regions that share the same transformation [76]. A CU can be related to one or more TUs [76]. This partition follows a quadtree methodology, transform tree or residual quadtree (RQT), over the residual blocks obtained by after the Intra prediction process [76]. The transformation is then applied to each leaf node of the RQT (TUs) [76].

The HEVC's main coding process can then be described as follow:

- **Transform, scaling and quantization –** the transformation applied over the TUs is done by scaled integer basis functions of a DCT transform [73]. Like in H.264/AVC, uniform reconstruction quantization (URQ) is applied over the DCT coefficients [73].
- **Intra-picture prediction –** prediction is made by using previously decoded samples from neighbouring blocks [73]. It can be a directional/angular prediction process with 33 orientations as shown in Figure 21 (right). In this prediction mode the TBs are predicted by spatial neighbouring reconstructed TBs in the specified orientation. The HEVC-Intra also has a DC intra prediction mode, where an average value of reference samples is determined for prediction, and a Planar intra prediction mode, in which the prediction is computed by the average value of two linear prediction using four reference samples.
- **Entropy coding –** only one method is specified by the HEVC standard, the Context-based Adaptive Binary Arithmetic Coding (CABAC) [77] to generate the coded bitstream [73].

Although, the HEVC standard represents an improvement relative to previous solutions, the state-of-art in conventional methods for image coding is nowadays the Versatile Video Coding (VVC) standard [78]. A VVC compliant codec is able to achieve half bit rate reductions, while maintaining the same video quality, compared to an HEVC codec [78].

3.2. Learning-based Coding

In the last years, learning-based image coding solutions have been developed to compete and even outperform existing conventional methods [79], in terms of rate-distortion performance. DL image coding uses an end-to-end approach, typically with an auto-encoder architecture [80]. The coding procedure is highlighted by the encoder-decoder architecture based on neural networks, where non-linear transforms are trained with a large amount of training data and some selected loss function [80].

These methods generate different compression artifacts in the decoded image, when comparing with the blocking and blurring degradation of conventional methods [79], [80]. The following learning-based image coding solutions are briefly described:

- **Full Resolution Image Compression with Recurrent Neural Network (FRICwRNN)** [81] **–** performs lossy image coding based on RNNs [80]. The recurrent characteristic of the model allows the propagation of information between different iterations of both the encoder and decoder process, aiming to achieve in a flexible way, several rate-distortion trade-offs [80].
- **Variational Image Compression with a Scale Hyperprior** [82] **–** end-to-end optimized image compression, by exploiting the concept of variational autoencoders [80], [82]. This solution uses a hyperprior (understood as side information), to exploit the spatial dependencies of the latent representation during the entropy coding process [82].
- **Joint Autoregressive and Hierarchical Priors for Learned Image Compression** [83] **–** performs end-to-end optimized image coding, by using hierarchical entropy models and generative models with an extra autoregressive component. It combines the autoregressive model with a hyper-network to improve context-based predictions.
- **Context-Adaptive Entropy Model for End-to-end Optimized Image Compression** [84] **–** proposes a context-adaptive entropy coding model that includes two types of contexts, bit-consuming context and a bit-free context. The used context is determined considering extra bit allocation that may be required. This leads to a more generalized entropy model, which allows to model the spatial dependencies of the latent representation more efficiently.

3.3. Performance Evaluation of Learning-based Image Codecs

For better analysis of the previously described image coding methods, their performance is evaluated with real images. None of the models and methods of these solutions were implemented, instead the results obtained in [80], [83] and [84] will be used to perform this analysis, in which different learning-based image solutions are studied in different testing environments.

Image Coding Solutions

The selected learning-based solutions are compared also to conventional methods, such as, JPEG, JPEG2000, HEVC Intra and WebP [85]. From [80], [83] and [84], a total of nine learning-based solutions, were defined, which are listed below:

- **FRICwRNN** [81][90] **–** FRICwRNN using a pre-trained Residual Gated Recurrent Unit (GRU) model [80].
- **FactMSE** [82][90] **–** Variational image compression with factorized entropy model obtained using mean squared error (MSE) as the loss function.
- **FactMS-SSIM** [82][90] **–** Same as previous codec but with multi-scale structural similarity index measure (MS-SSIM) as the loss function.
- **HyperMSE** [82][90] **–** Identical to FactMSE but with a hyperprior model.
- **HyperMS-SSIM** [82][90] **–** Identical to FactMS-SSIM but with a hyperprior model.
- **ArHPMSE** [83] **–** Joint autoregressive and hierarchical priors for learning image compression with MSE as the loss function.

- **ArHPMS-SSIM** [83] **–** Same as previous codec but with MS-SSIM as the loss function.
- **ContAdMSE** [84] **–** Context-adaptive entropy model for end-to-end optimized image compression with MSE as loss function.
- **ContAdMS-SSIM** [84] **–** Same as previous codec but with MS-SSIM as loss function.

The solutions FRICwRNN, FactMSE, FactMS-SSIM, HyperMSE and HyperMS-SSIM are the pre-trained solutions made available by the authors of the corresponding paper. For the training of the ContAdMSE and ContAdMS-SSIM solutions it was used 256x256 patches randomly extracted from the YFCC100m image set [87]. There is no information about the training process of ArHPMSE and ArHPMS-SSIM.

Test Conditions

In each of the selected codecs ([80], [83], [84]), different datasets are used to evaluate the learning-based image coding solutions. In [80] it was selected eight images from the JPEG AI dataset (Appendix A) and for [83] and [84] it was used the Kodak dataset [86].

In [80] it is applied a subjective test methodology, the Double Stimulus Impairment Scale (DSIS) Variant I [80], based in human judgement, to obtain the Mean Opinion Scores (MOS) [80]. While in [83] and [84], the performance evaluation is done by the means of two objective quality metrics, the Peak-Signal-to-Noise Ratio (PSNR) and the Multi-Scale Structural Similarity Index Measure (MS-SSIM) [88], [89]. To obtain several rate-quality points, the images are encoded for a set of rates (each corresponding to a different NN model), which are measured by bit per pixel (bpp), covering different compression rates for each solution.

Experimental Results and Analysis

The rate-MOS plots obtained from the selected papers for the evaluation of the FRICwRNN, FactMSE, FactMS-SSIM, HyperMSE and HyperMS-SSIM are presented in Figure 22. By analysing these results, some conclusions can be made:

- **JPEG and FRICwRNN –** As expected, the JPEG and FRICwRNN solutions revealed to have a similar behaviour, usually with inferior performance than other tested codecs in the observed bitrates [80]. In several images the FRICwRNN outperforms the JPEG solution.
- **JPEG2000 –** The JPEG2000 has high performance, especially for high bitrates, being among the best solutions for several images.
- **MSE vs MS-SSIM –** The FactMSE and HyperMSE solutions usually have worse performance than other codecs, being outperformed by their MS-SSIM optimized solutions, FactMS-SSIM and HyperMS-SSIM, which are among the best solutions in several images. This can be explained by the fact that the MS-SSIM metric better represents the human assessment of image quality.
- **HEVC Intra –** Achieves a similar performance compared to the best learning-based codecs, representing the best conventional coding solutions in many images.

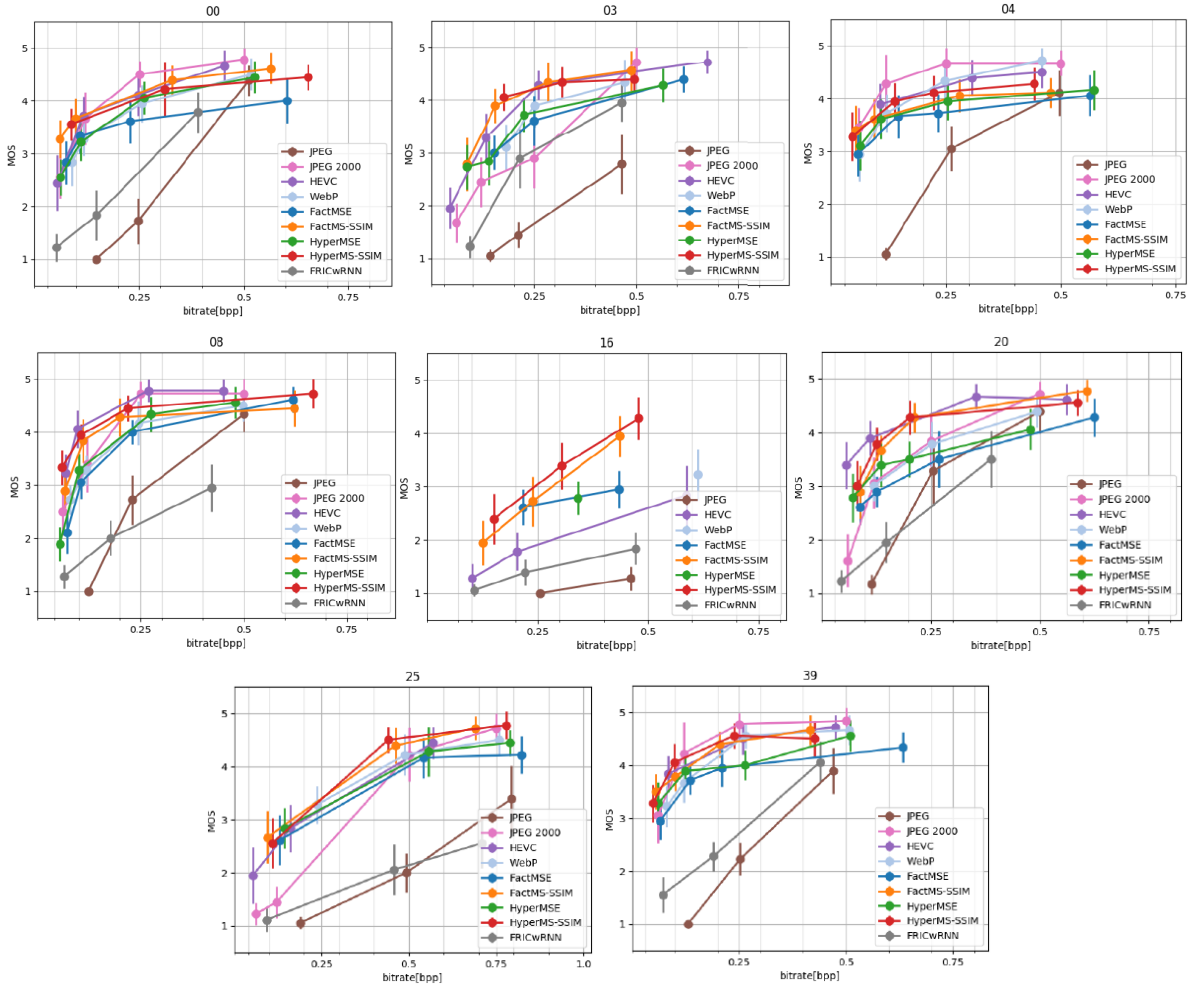

Figure 22 – *MOS results [80] for test images TE00, TE03, TE04, TE08, TE16, TE20, TE25, TE39 (Appendix A), for performance evaluation of the FRICwNN, FactMSE, FactMS-SSIM, HyperMSE and HyperMS-SSIM solutions.*

The objective quality evaluation of the ArHPMSE (as "Our Method" and "Our Method (opt. for MSE)") and ArHPMS-SSIM (as "Our Method (opt. for MS-SSIM)") solutions, given as bitrate-PSNR and bitrate-MS-SSIM plots, are presented in Figure 23. Important to notice that the codecs identified in Figure 23 as *Ballé* solutions optimized for MSE and MS-SSIM correspond to the HyperMSE and HyperMS-SSIM solutions respectively. The following conclusions can be drawn:

- **PSNR Evaluation –** ArHPMSE achieves a better performance than all codecs, outperforming the HyperMSE solution. However, both MS-SSIM solutions were not evaluated for this metric.
- **MS-SSIM Evaluation –** The ArHPMSE is still placed among the best solutions but does not outperform the HyperMS-SSIM. The ArHPMS-SSIM has superior performance than all other codecs, outperforming HyperMS-SSIM.
- The proposed solutions ArHPMSE and ArHPMS-SSIM have higher coding efficiency compared to the HyperMSE and HyperMS-SSIM codecs, especially because of the joint autoregressive and hierarchical prior models.

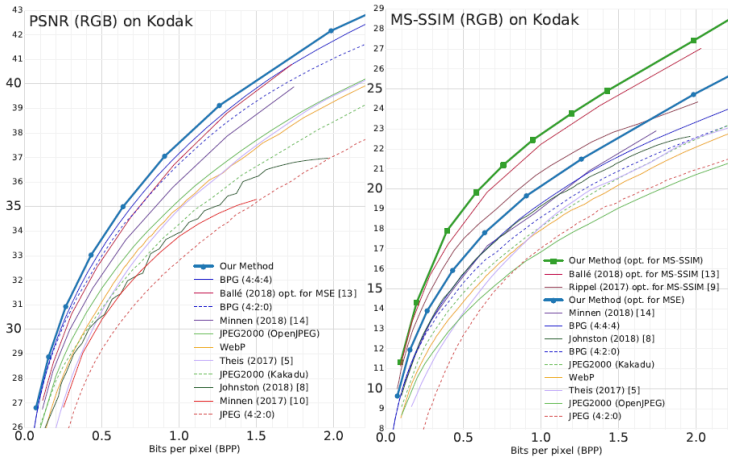

Figure 23 – *Testing results on the Kodak dataset of the: ArHPMSE (as "Our Method" and "Our Method (opt. for MSE)"), using PSNR (left), and ArHPMS-SSIM (as "Our Method (opt. for MS-SSIM)"), using MS-SSIM (right) [83].*

In Figure 24 is shown the bitrate-PSNR and bitrate-MS-SSIM plots from [84] that represent the objective quality evaluation of the ContAdMSE (as "Proposed (optimized for MSE)") and ContAdMS-SSIM (as "Proposed (optimized for MS-SSIM)") solutions, from which some conclusions can be drawn:

- **PSNR Evaluation -** ContAdMSE outperforms all other codecs, including the HyperMSE, while the ContAdMS-SSIM competes with the HyperMS-SSIM for the lowest performance. This was expected since PSNR evaluation favors codecs optimized with the MSE metric.
- **MS-SSIM Evaluation -** ContAdMS-SSIM has a performance superior to all other codecs. The ContAdMSE still outperforms the HyperMSE and the conventional methods but fails to outperform the HyperMS-SSIM.
- The proposed solutions ContAdMSE and ContAdMS-SSIM have an higher performance compared to the HyperMSE and HyperMS-SSIM, especially because of the context-adaptive entropy coding model.

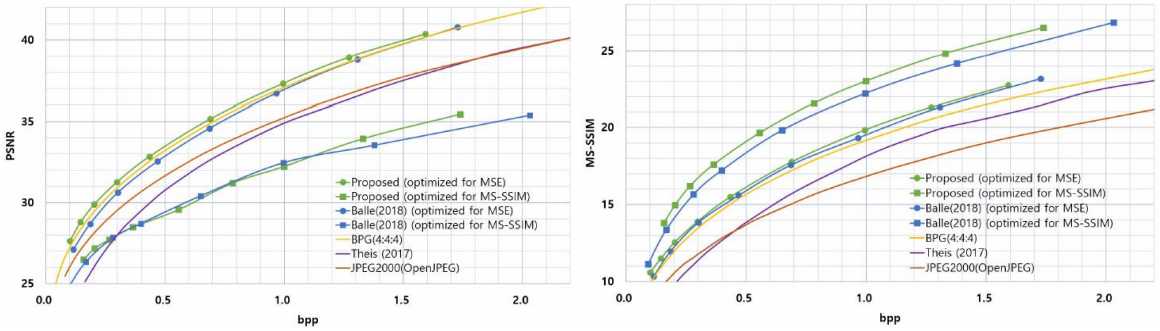

Figure 24 - *Testing results on the Kodak dataset of the: ContAdMSE (as "Proposed (optimized for MSE)") and ContAdMS-SSIM (as "Proposed (optimized for MS-SSIM)") solutions, using PSNR (left) and MS-SSIM (right) metrics [84].*

These results allow to conclude, that state-of-art learning-based image coding solutions are able to compete in compression efficiency against conventional image coding codecs. Despite the advances made learning-based image coding solutions are still research today to further increase the coding efficiency and add new requirements such as progressive decoding.

4. Relevant Compressed Domain Deep Learning Based Image Analysis Solutions

The growth in machine vision related applications, and the massive increase of imaging data led to the need for systems to perform efficient analysis and processing in the compressed domain [1]. These technologies may take advantage of the learning-based image coding solutions capability to extract relevant features suitable not only for human consumption but also for machine consumption, i.e. for automated image analysis and processing tasks. Often, it is possible to assign a triple-purpose (decoding, image processing and computer vision) for the latent representation, as shown in Figure 25. Computer vision focus on extracting relevant information from the input image to gain high-level understanding of the input, usually with the purpose of automating visual analysis tasks, e.g., semantic segmentation, face recognition, image classification, etc. Image processing applies transformations to the input image resulting in an output image with some enhanced characteristics. Some image processing tasks examples are super-resolution, denoising, color correction, etc. Moreover, these computer vision and image processing tasks can be performed over the latent representation obtained by applying entropy decoding on the coded bitstream.

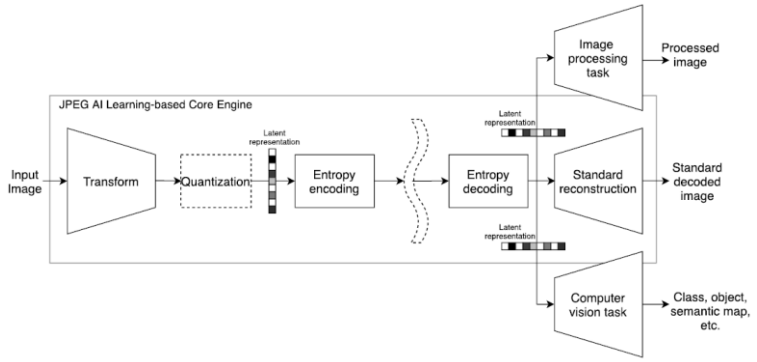

Figure 25 – *JPEG-AI learning-based image coding framework [1].*

The JPEG-AI learning-based image coding framework present multiple advantages for image processing and computer vision tasks, namely:

- Allows the same compressed bitstream to be used for decoding, image processing and computer vision tasks.
- Reduces the resources needed to perform these tasks, since it is not necessary to fully reconstruct the image at the decoder side.
- Allows to perform these tasks by using features extracted from the original image instead of the lossy decoded images, which may have artifacts that impair the performance. When reconstructing the image in a learning-based codec, the decoder is usually trained to perform the reconstruction in a manner that the reconstructed image can be used to perform specific tasks. This process changes the information contained in the image, which can significantly impact the performance of other computer vision and image processing applications.

In this chapter it will be reviewed three selected compressed domain deep learning-based image analysis solutions for image classification, semantic segmentation and face recognition.

4.1. Towards Image Understanding From Deep Compression Without Decoding: Image Classification

In [90], two methods are proposed to perform image classification (Section 2.3.1) and semantic segmentation (Section 2.3.3) on the latent representation obtained by a representative learning-based image coding solution. The models are obtained by the design of new decoders and training procedures that perform these computer vision tasks in the compressed domain [90]. However, these models were not implemented, it is only used the results obtained in the corresponding paper for better analysis of the described solutions. In this Section, compressed domain image classification is presented while in the next Section is presented the compressed domain semantic segmentation.

4.1.1. Objective

The goal of [90] is to design, implement and evaluate a well-designed compressed domain image classification network that uses latent representations as input, using 1.5x to 2x fewer operations [90]. The compressed domain solution has competitive accuracy with solutions that require the full decoding process to be performed (reconstructed domain solution), while lowering computational complexity.

4.1.2. Architecture

The learning-based image coding solution proposed in [93], trained by a variant of the training procedure presented in [94] is used in this work. The latent representation of the encoder has a dimension of (w/8)x(h/8)xC, where w and h the image spatial dimensions and C the number of channels [90]. This coding solution was trained in [90] for three different compression rates (0.0983bpp, 0.330bpp and 0.635bpp), obtaining three different coding models, each operating at a different rate.

A variant of the ResNet-50 (V1) [92], the cResNet-k (with k as the number of convolutional layers and c standing for compressed), is used to perform image classification in the compressed domain [90]. The cResNet-k networks' implementation process is accomplished by removing the first convolutional layers, namely the first layer and the residual blocks with spatial dimension larger than 28x28 (conv3_x) [90]. For the models to have the corresponding k convolutional layers, 14x14 residual blocks (conv4_x) are inserted based on some architectural recipe [92]. Three different architectures are then obtained:

- **cResNet-39** [90] **–** obtained by removing ResNet-50's first 11 layers as previously described resulting in a total of 39 layers, to represent the solution with lowest complexity.
- **cResNet-51** [90] **–** obtained by adding 14x14 residual blocks to cResNet-39 resulting in 51 layers.
- **cResNet-72** [90] **–** obtained by adding 14x14 residual blocks to cResNet-51, to a total of 72 layers.

A more complete description of the obtained architectures is presented in Table 1.

Table 1 – *Description of the ResNet and cResNet-k architectures [90].*

Network	root	conv2_x	conv3_x	conv4_x	conv5_x	FLOPs
		56×56	28×28	14×14	7×7	$[\times 10^9]$
ResNet-50	yes	3	4	*6*	3	3.86
ResNet-71	yes	3	4	*13*	3	5.38
cResNet-39	no	none	4	*6*	3	2.95
cResNet-51	no	none	4	*10*	3	3.83
cResNet-72	no	none	4	*17*	3	5.36

For the reconstructed domain solutions, the ResNet-50 and ResNet-71 were selected since variants of these architectures are used for the compressed domain image classification network, naturally using the fully decoded RGB images. The ResNet-50 is a convolutional network with residual units that has 50 layers and thus has more layers compared to cResNet-31 [90], [92] working with input data with higher resolution. The ResNet-71 is constructed by adding seven 14x14 residual blocks (conv4_x) to the ResNet-50, based in [92]. A more detailed description of these networks is presented in Table 1, that also shows the complexity differences with the compressed domain solutions. A ResNet-50 solution (ResNet-50-original) is also used to perform image classification over the original image.

4.1.3. End-to-End Training

Independent Training

The training in [90] was performed with the ImageNet dataset from the ILSVRC2012 [95], containing 1.28M training samples and 50k validation samples distributed across 1000 classes. In this case, the pre-trained encoder and decoder networks do not change during the training of both compressed domain solutions and reconstructed domain solutions.

The output of the encoder (latent representation) is used as the input of the cResNet solutions (compressed domain) and the output of the standard decoder is the input of the ResNets (reconstructed domain). The hyperparameters and a similar pre-processing procedure of [92] were used, with a faster decay of the learning-rate through the epochs to speed up training [90]. A Stochastic Gradient Descent (SGD) [101] optimizer is employed with momentum 0.9 [90], batch size 64 and initial learning-rate 0.025 for 28 epochs. Before processed by the image classification network, the input is cropped, being performed a 224x224 random crop on the reconstructed images and a 28x28 random crop on the coded representations [90]. This procedure is then performed for each operating rate resulting in three different models for each cResNet and ResNet architectures. For a more detailed performance analysis, a ResNet-50 (ResNet-50-original) network was trained in [90], using the same pre-processing and learning-rate but over uncompressed images (original).

Joint Training

In [90] it is also presented a joint training procedure that trains the encoder, standard decoder and image compressed domain classification networks jointly. A new architecture was implemented to evaluate the proposed method, that is constructed by combining the coding network and the cResNet-51 [90], in the same way as Figure 25. A new loss function is then obtained by the combination of the networks' loss functions [90].

The entire coding network and the cResNet-51 are initialized by individually training them, as previously described. From this point, the networks are jointly finetuned, where the learning-rate is set to be low enough, so that weight updates suffer a small variation [90]. This process will train the compression network to assign greater importance to features that are more relevant for the image classification task. To better analyse the impact of joint training, the cResNet-51 was trained from scratch by using the coded representation of the newly trained coding solution for 9 epochs [90].

4.1.4. Performance Evaluation

The top-1 classification accuracy and top-5 classification accuracy are used to measure the performance of the trained models. The top-1 score corresponds to the probability of the top label, i.e., label with highest confidence, being the true label. While the top-5 score corresponds to the probability of one of the 5 highest probability labels, i.e., the five labels with highest confidence, being the true label.

The validation set from the ILSVRC2012's ImageNet dataset [95], 50k samples, is used for evaluation. The results of the models obtained from [90] are shown in Table 2.

As expected, the top-5 accuracy scores are significantly better than the top-1 scores, since it is a less demanding accuracy metric. From Table 2 is also possible to conclude:

- **Best accuracy –** the best observed accuracy for both top-1 and top-5 scores is the ResNet-50-original applied over the original image (just the 1st line of Table 2). This is an expected behaviour since there is loss of information in the coding process.
- **Compressed vs Reconstructed –** the performance of both cResNet-51 and cResNet-39 are similar to the results obtained for the ResNet-50 (reconstructed domain) in both accuracies and for all rates. For the cResNet-51 and cResNet-39 the highest observed difference in accuracy to the ResNet-50 is 0.55% and 1.11% respectively.
- **Compressed vs Original –** for 0.635bpp the cResNet-39 and cResNet-51 models have a competitive performance against the Resnet-50-original applied over the original image, with a difference of 2.77% and 2.34% respectively. This gap in performance increases with the decrease of the target rate, reaching 13.68% and 13.07% for 0.0983bpp (lowest rate). This behaviour is expected since more information is lost when the target rate is decreased.

Table 2 – *Image classification accuracies for the ResNet-50-original (original image), ResNet networks (reconstructed domain representations) and cResNet networks (compressed representation) over the target bitrates, created from the data available in [90].*

bpp	Network architecture	Top 5 acc. [%]	Top 1 acc. [%]
	ResNet-50-original	89.96	71.06
0.635	ResNet-50	88.34	68.26
	cResNet-51	87.85	67.68
	cResNet-39	87.47	67.17
0.330	ResNet-50	86.25	65.18
	cResNet-51	85.87	64.78
	cResNet-39	85.46	64.14
0.0983	ResNet-50	78.52	55.30
	cResNet-51	78.20	55.18
	cResNet-39	77.65	54.31
	ResNet-71	79.28	56.23
	cResNet-72	79.02	55.82

The joint training shows no significant impact in the performance of the coding solution [90], measured by MS-SSIM and PSNR quality metrics. As presented in Figure 26, Top-5 accuracy results from [90] are shown for networks where only finetuning of the coding network was allowed and by finetuning both coding and image classification networks. For 0.635bpp it is not noticeable a great improvement in performance for the finetuning methods. While for lower resolution (0.0983bpp) the joint finetuning of the networks showed some improvement with respect to the cResNet-51 baseline.

The compressed domain solution present in this Section represents an exciting approach for image classification. It can achieve similar (and in some cases even better) accuracy performance as

applying these tasks in the reconstructed domain (i.e., for decoded images and even for original images), while having significantly lower complexity.

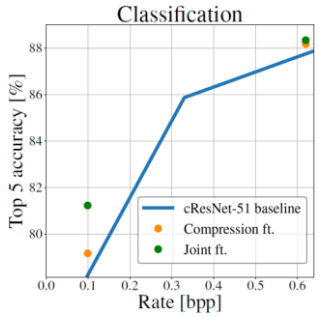

Figure 26 *– Image classification results of finetuning the compression (coding) network and of finetuning both coding and image classification networks [90].*

4.2. Towards Image Understanding From Deep Compression Without Decoding: Semantic Segmentation

In this Section the compressed domain deep learning-based semantic segmentation solution proposed in [90] is reviewed.

4.2.1. Objective

The main objective is to design, implement and evaluate a compressed domain semantic segmentation network that can have similar or better performance compared to reconstructed domain networks, while reducing complexity. The same image coding solution from Section 4.1.2 was used to obtain the latent representation used by the compressed domain segmentation solution. However, the models from [90] are not implemented, therefore the results obtained in the corresponding paper are used to analyse the solution.

4.2.2. Architecture

The cResNet architecture for semantic segmentation is an adaptation of the ResNet-based Deeplab architecture [96], [97]. To obtain the proposed models, the cResNet architecture in Section 4.1.2 is re-purposed to perform semantic segmentation. This is achieved by performing dilated/atrous convolutions, where the filters upsample instead of downsample [90], [96] the feature maps, obtaining the cResNet-segm variant. This way there is an increase in the receptive field and prevent against aggressive subsampling [90]. To compute the final pixel-wise classification, the last classification layer of the cResNet architecture is replaced by an atrous spatial pyramid pooling (ASPP) [90]. In this variant the output feature map has the same spatial dimension as the input latent representation (8x smaller than the original image). Therefore, by applying the above described process it is obtained in [90] the cResNet-39-segm (39 layers), cResNet-51-segm (51 layers) and cResNet-72-segm (72 layers) solutions [91], [92].

The ResNet-segm architecture used for semantic segmentation over the reconstructed image is obtained in a similar way as cResNet-segm [90], [96], [90], namely the ResNet is restructured with dilated convolutions. The output feature map of this architecture has spatial dimension 8x smaller than the original image. The feature maps computed by the compressed domain and reconstructed domain models will have comparable spatial resolution [90]. Like in the cResNet-segm, the ResNet-segm has

an ASPP for the classification of each pixel. By employing the above described procedure it is obtained the following solutions: ResNet-50-segm with 50 layers and ResNet-71-segm with 71 layers [90], [90]. It is also implemented in [90] a ResNet-50-segm solution (ResNet-50-segm-original) for semantic segmentation over the original image.

4.2.3. End-to-end Training

Independent Training

In this case, the learning-based image coding solution is maintained fixed. The training procedure applied in [90] is then divided into two phases:

- **Pre-training –** the cResNet and ResNet solutions are pre-trained for image classification tasks, following the processing steps described in Section 4.1.3. The ILSVRC2012's ImageNet dataset [95], containing 1.28M training samples, is used.
- **Finetuning –** the semantic segmentation cResNet-segm and ResNet-segm solutions are obtained by adapting the trained image classification cResNet and ResNet models of the previous step with dilated convolutions [90]. To perform finetuning, the PASCAL VOC-2012 dataset for semantic segmentation [99] is used, with 20 object classes and 1 background class. The dataset has 1464 training samples being added extra samples according to [100], resulting in a total of 10582 samples. Then, finetuning for semantic segmentation is applied following the procedure described in [90]. A SGD optimizer is used for training with momentum 0.9, batch size 10 and initial learning-rate 0.001, 0.01 in the final classification layer, for a total of 20k iterations. The learning-rate is updated at each iteration following the rules presented in [90]. Before performing semantic segmentation, random cropping is applied on the input images with a spatial resolution of 320x320 for the reconstructed domain networks and of 40x40 for the compressed domain networks [90].

The ResNet-50-segm-original is trained in [90] in a similar way as the ResNet-segm solutions but over the original input images.

Joint Training

A joint training procedure is presented in [90] to achieve better performance in semantic segmentation tasks. This process is accomplished by performing the joint training process described in Section 4.1.3, obtaining the cResNet-51 joint trained network. After network adaptation and finetuning, cResNet-51-segm is obtained, which in [90] is trained in a similar way as described above.

4.2.4. Performance Evaluation

To evaluate the proposed compressed domain solutions, it is employed a performance metric that is normally used in semantic segmentation: mean intersection over union (mIoU). The IoU measure, intersection over union, determines the similarity between the predicted region and the ground-truth region of an object (class) by calculating the area of the intersection divided by the area of the union of these regions [98]. This way, the IoU is able to take into consideration the class imbalance issue encountered in most images [98] where background pixels are often much more. In [90], this process is done for each of the considered classes of objects, obtaining the mIoU score by computing the mean IoU from all classes. The PASCAL VOC-2012 dataset for semantic segmentation [99], containing 1449

validation samples, is used to evaluate the compressed and reconstructed domain solutions performance. The results of [90] are shown in Table 3, where some conclusions can be drawn:

- **Compressed vs Reconstructed –** the compressed domain solutions (cResNet-51-segm and cResNet-39-segm) have similar performance to the reconstructed domain solution (ResNet-50-segm) across the defined rates (0.635bpp, 0.330bpp and 0.0983bpp). For the lowest target rate (0.0983bpp), cResNet-51-segm and cResNet-39-segm perform even better than ResNet-50-segm. This was expected since compressed domain networks (cResNet-51-segm and cResNet-39-segm) receive as input the latent representation and not a decoded image (as in ResNet-50-segm) for which coding artifacts may have a significant impact on the segmentation.
- **Compressed vs Original –** the ResNet-50-segm-original, applied over the original image, achieves the best performance between all solutions, as expected since no information of the original image is lost. The compressed domain solutions cResNet-51-segm and cResNet-39-segm have competitive performance in the highest target rate (0.635bpp) with the original image solution, with a difference of 4.40% and 5.93% respectively.

Table 3 – *mIoU performance for the ResNet-50-segm-original (original image), ResNet-segm networks (reconstructed domain representations) and cResNet-segm networks (compressed representation) over the target bitrates, created from the data available in [90].*

bpp	Network architecture	mIoU [%]
	ResNet-50-segm-original	65.75
0.635	ResNet-50-segm	62.97
	cResNet-51-segm	62.86
	cResNet-39-segm	61.85
0.330	ResNet-50-segm	60.75
	cResNet-51-segm	61.12
	cResNet-39-segm	60.78
0.0983	ResNet-50-segm	52.97
	cResNet-51-segm	54.62
	cResNet-39-segm	53.51
	ResNet-71-segm	54.55
	cResNet-72-segm	55.78

The results in Table 3 and Figure 27 (left), obtained from [90], show the effect of decreasing the target rate, which leads to a worse performance. This occurs because at these rates more image details are lost [90]. Figure 27 (left), also highlights the challenging of the semantic segmentation task, being visible some distortion on the detected objects [90].

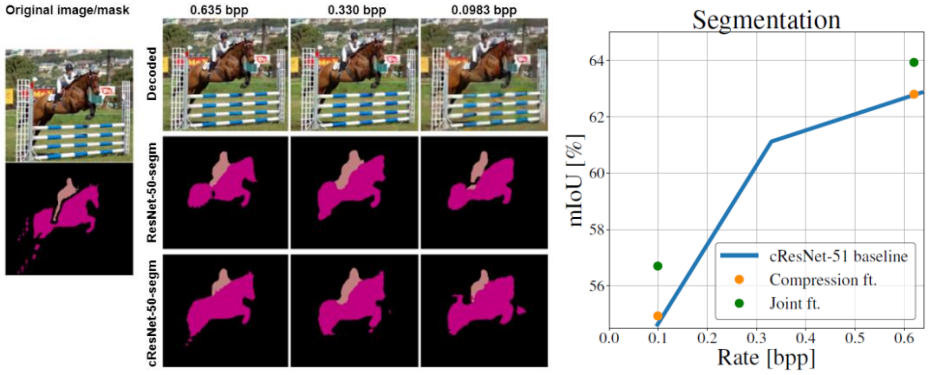

Figure 27 – *Semantic segmentation performed by ResNet-50-segm and cResNet-51-segm at each operating point [90] for some image (left); Semantic segmentation results of finetuning the compression (coding) network and of finetuning both coding and sematic segmentation networks [90] (right).*

As also shown in Section 4.1.4, the joint training did not have any noticeable impact on the compression performance of the image coding solution. However, by jointly training both coding and image classification networks in [90], as presented in Figure 27 (right), better performance is achieved for the semantic segmentation task, leading to a significant performance improvement in all rates.

The proposed compressed domain solutions present to be an exciting technology for semantic segmentation. They can perform as accurately as reconstructed domain solution while achieving significantly lower complexity.

4.3. Scalable Facial Image Compression with Deep Feature Reconstruction

A scalable image compression solution for surveillance systems is proposed in [102], which includes two layers, the base layer for image analysis and the enhancement layer for conventional pixel-wise reconstruction. This information is used at the decoder side for face recognition and image decoding. The results obtained in [102] are used to make the performance evaluation of this solution, which means that these models were not implemented.

4.3.1. Objective

Traditional surveillance systems are usually based on compress-then-analyse (CTA) schemes, in which the retrieved data from security cameras is compressed and then transmitted for image analysis. High compression rates are often employed due to the bandwidth limitation of these systems, which leads to degradation of the image analysis and distortions in decoded textures [102]. Consequently, the low bit-rate compression has a significant impact in the image analysis performance.

To overcome this problem, some solutions adopt the analyse-then-compress (ATC) paradigm. These systems extract features from the original image, which are then compressed and transmitted for analysis. Since features are represented with less information, it is possible to achieve promising analysis performance at high compression rates [102]. The problem is that the extracted features are not capable of accurately reconstructing the texture information of the image, i.e. image decoding can only be achieved for low perceived qualities.

The objective of [102] is to propose a scalable framework that can deal with these issues by inheriting the advantages of both CTA and ATC schemes. To implement these systems, two main blocks were defined in [102], the base layer, responsible for the representation of facial features essential for recognition, and the enhancement layer, responsible for texture feature representation. Some of the advantages of this method can be listed as below:

- **Image analysis reliability –** since feature extraction for analysis is performed over the original image as in the ATC paradigm, a more reliable solution can be obtained.
- **Independent image analysis –** only the base layer needs to be transmitted for image analysis to be performed; the texture information may not be transmitted, saving bitrate.
- **Improvement in texture compression performance –** obtained by exploiting the redundancy between features for analysis and the features for texture representation.

4.3.2. Architecture

The architecture proposed in [102] is illustrated in Figure 28, being represented the encoder and decoder components and their main operations.

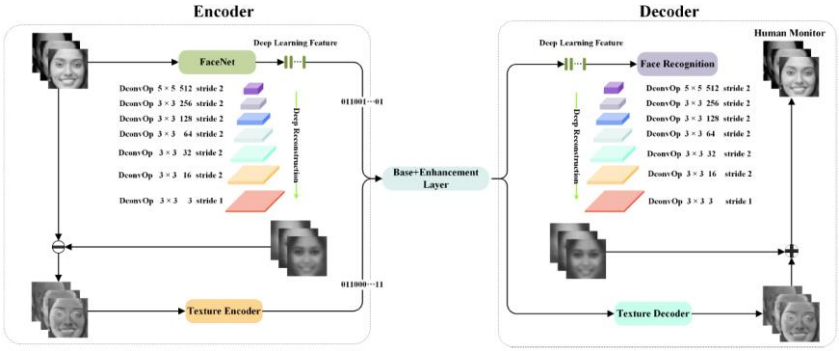

Figure 28 – *Architecture of the scalable facial image compression [102] solution.*

In the encoder side, three main operations are defined:

- **Base layer compression –** task performed by the FaceNet [103] system, in which the facial images are used as input for the deep neural network [102]. A vectorized representation of the facial features with 128 dimensions is generated [102]. They are then compressed with quantization and entropy coding, to be transmitted for future analysis.
- **Deep face reconstruction –** the extracted features of the previous step are used to reconstruct the texture of the facial image; the de-convolutional neural network described in [104] is used (DconvOp). The DconvOp is composed of a series of components, including de-convolution layer, batch normalization and ReLU activation, with exception of the last layer in which a Tanh activation is employed [102].
- **Enhancement layer compression –** the enhancement layer, represents the residual information between the input image and the reconstructed face obtained in the previous step. To compress this data, different codecs can be used, being selected in [102] three solutions: JPEG, JPEG2000 and the learning-based model of [105]. But before coding, a min-max normalization is performed on the residual information, thus being necessary to encode and transmit the normalization parameters to accurately reconstruct the texture.

The features of the base layer after compression are used for face recognition at the decoder side but they are also used at both encoder and decoder sides to obtain a coarse estimation of the facial image [102]. The enhancement layer is composed by the information needed to enhance the quality of this estimation of the facial image. This means that at the decoder side, effective face recognition and image reconstruction with high perceptual quality can be obtained.

4.3.3. Techniques

The deep feature reconstruction network is trained in [102] with the unconstrained large scale face dataset VGG-Face2 [106]. This training set is composed of more than 3.3M face images distributed over 9131 subjects. The deep face reconstruction network is first trained with this dataset and then a residual image dataset is obtained for training the end-to-end enhancement layer compression network.

The deep face reconstruction model implementation in [102] is done with TensorFlow, in which the initialization of the network is based in [107]. The training in [102] uses the adaptive moment

estimation (Adam) [108] optimizer with a defined batch size of 64 and a learning-rate that exponentially decays from 0.01 to 0.0001 for 50 epochs. For the training of the deep face reconstruction model, it is used in [102] a linear combination of the MAE (Mean Absolute Error) and a perceptual loss that is computed by using the VGG-19 neural network. This is mainly to ensure that the structure information is correctly reconstructed, instead of texture information (which is not available in the features computed by the FaceNet model).

In the learning-based enhancement layer coding solution, the Adam optimizer is also used for training in [102], with a batch size of 16 and a learning-rate of 0.0002 for a total of 10 epochs.

4.3.4. Performance Evaluation

The evaluation of the proposed solution is divided into two different performance analysis, the face verification and the standard reconstruction. In the paper [102], different learning-based coding models are trained to operate at different rates and thus several coding rates are employed over the base and enhancement layer. The standard reconstruction performance in [102] is measured by an objective quality metric, the PSNR, while the face verification performance is evaluated by measuring the accuracy in identifying if two faces correspond to the same person or not. Both analyses are performed by comparing the proposed solutions with CTA scheme models implemented with JPEG and JPEG2000 to compress the image [102].

The proposed solution is evaluated with the face verification dataset, Label Faces in the Wild (LFW) [109], that contains 13233 images of 5749 individuals in variable image conditions. The obtained results for rate-accuracy face verification performance and rate-distortion performance can be observed in Figure 29, in which the bit-rate values (bpp) correspond to the sum of the coding bits of the base layer and the enhancement layer.

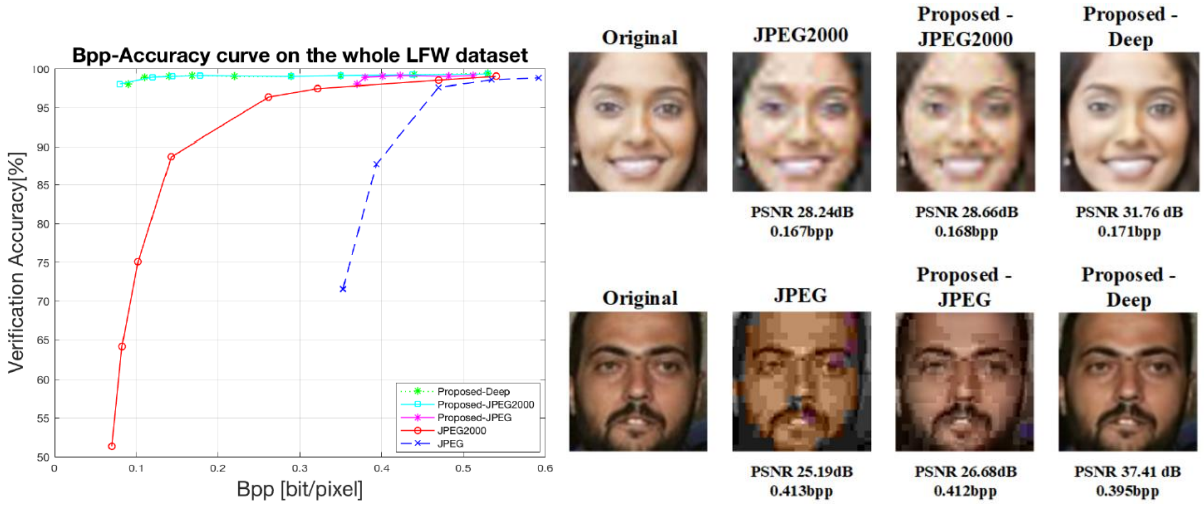

Figure 29 – *Rate-accuracy face verification performance obtained for the proposed solutions and for the CTA JPEG and JPEG2000, over the LFW dataset (left) [102]; rate-distortion performance and reconstructed images obtained for the proposed solutions and for the CTA JPEG and JPEG2000 (right) [102].*

From the obtained rate-accuracy curve shown in Figure 29 (left) the following conclusions can be drawn:

- **Overall performance –** the proposed solutions achieved the best performance, having similar accuracy between each other. The gap in performance from the CTA models becomes higher when

the bitrate becomes lower. This was expected since the proposed models extract the features from the original image, while in the CTA solutions, the features are extracted from the lossy reconstructed images.

- **Accuracy vs bitrate –** for low bitrates there is a slightly lower accuracy of the proposed solution compared to higher bitrates. This occurs due to loss of information due to the quantization of the face features.

The following conclusion can be drawn from Figure 29 (right) when observing the rate-distortion performance and by performing a subjective analysis on the reconstructed images:

- **Overall performance –** the proposed solutions presented the best performance regarding image quality, expressed by the PSNR values (bitrate is similar). It is also possible to conclude that the reconstructed images have better perceptual quality. The best results are achieved when an enhancement layer learning-based image compression is used, compared to JPEG or JPEG2000.

The proposed scalable facial image compression solution is rather promising when facial features and texture information needs to be transmitted. They can inherit the ATC advantages, achieving high accuracy in face recognition, and significantly improve the compression performance, when comparing to conventional coding methods.

5. Anchor Creation for Compressed Domain Face Recognition

This chapter introduces the design, implementation and assessment of the anchors and the different models and methods that are used to obtain the baseline results for the novel compressed domain face recognition model to be presented in the next Chapter.

5.1. Context and Objectives

The main objective of this MSc Thesis is the design, implementation and assessment of a novel compressed domain face recognition model using some performance metrics. This performance evaluation must be done in comparison with relevant anchor solutions based in state-of-art models. The anchors choice has a strong impact in the conclusions that can be taken, since they define the baseline to which the proposed solution will be compared.

Face recognition corresponds to identification or verification of a person's identity through his/her face. There are two types of face recognition tasks: face verification and face identification. Face identification is a multi-classification problem, in which the person's face is compared to a database of labelled faces to obtain the person identity or the "Unknown User" label. Alternatively, face verification is the binary classification problem (True/False, Yes/No, 0/1, etc.) of verifying if the person's identity is correct by using facial traits, i.e., if the person really is who he/she claims to be. The user's face is compared to previously collected reference faces of the alleged identity to make the classification decision. The focus of this MSc Thesis will be on face verification, hence the models and methods are implemented to assess this task. The selected performance metrics are crucial for a solid and complete analysis and thus two different anchors are defined:

- **Original anchors**: these anchors represent the ATC scheme (Analyze-then-Compress), in which the face recognition task is applied over the original image.
- **Decoded anchors**: these anchors represent the CTA scheme (Compress-then-Analyze), where the image is initially coded (usually to be transmitted/stored), decoded and then face recognition is performed over the lossy reconstruction of the image.

The anchors are mainly composed by a face detection model and a face recognition model. For the decoded anchor, several codecs are possible which leads to different baseline results for each anchor type.

5.2. Original Anchor for Face Verification

The original anchors represent the ATC scheme for face verification tasks, in which facial processing is performed over the original image. The extracted information and image can then be compressed, usually for storage and/or transmission. Since face verification is applied over original images, the input does not present artifacts due to compression, hence it is expected that the original anchors achieve higher performance than the other models.

The original anchors are mainly composed by a face detection and processing model, a face recognition model and a similarity score and decision-making model, as shown in Figure 30.

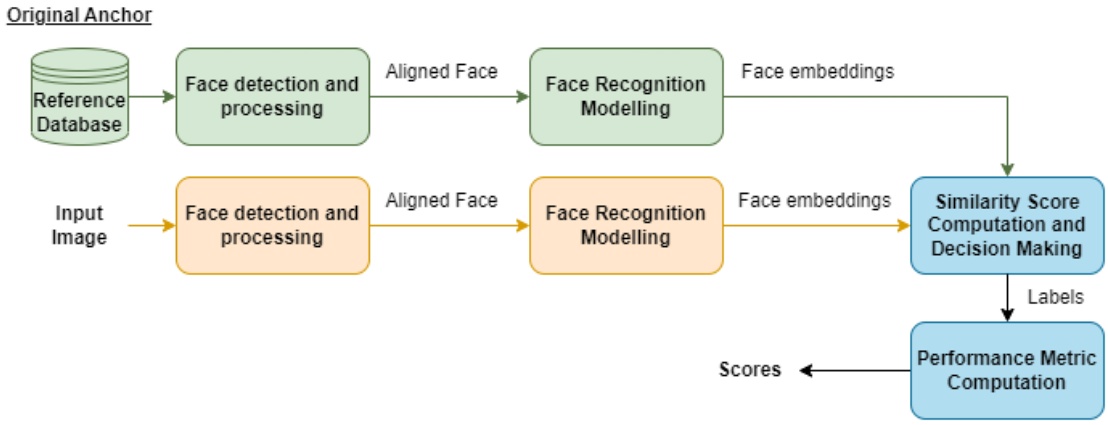

Figure 30 – *Original anchor architecture for face verification.*

The original anchor architecture can be described in the following steps:

- **Reference Database –** database that contains all the original images of the faces used as references.
- **Face Detection and Processing –** the facial region that is going to be verified needs to be identified from the entire image. The image goes through a face detection model that returns the coordinates and landmarks of the detected faces in the image, according to a predefined confidence threshold. Since more than one face can be detected in an image, other methods need to be applied to retrieve the labelled face, which is further described in Section 5.4. The selected face then needs to be pre-processed, to obtain a resized and aligned face, before used by the next module.
- **Face Recognition Modelling –** deep neural network model which takes an image of a face and generates a face embedding that can be used in different face recognition tasks.
- **Similarity Score Computation and Decision Making –** the obtained face embeddings represent information about different features of the input face, however, to perform face verification a similarity score is computed. The numerical similarity between two face embeddings allows to perform the comparison between two faces. In face verification, these scores are used to make a binary decision (True/False), determining if the input face corresponds to the specified identity.
- **Performance Metric Computation –** the predicted labels (True/False) obtained in the decision-making module are used to make a performance assessment using the ground truth labels and the selected performance metrics.

5.3. Decoded Anchor for Face Verification

The decoded anchors represent the CTA scheme, where the original images are initially encoded, obtaining the bitstream after entropy coding; this compressed representation is usually used for storage and/or transmission. At the decoder, entropy decoding is performed (using as input the bitstream) and the latent representation is obtained; naturally, from this latent representation is possible to obtain a lossy reconstruction of the image (synthesis transform). The decoded image is then used to perform face verification with original images as reference. Since there is loss of information in the encoding process it is expected an impact on the performance depending on the severity of the compression. The decoded anchors are mainly composed by a learning-based image codec, face

detection model, face processing, face recognition model and a similarity score and decision making process, as represented in the pipeline of Figure 31.

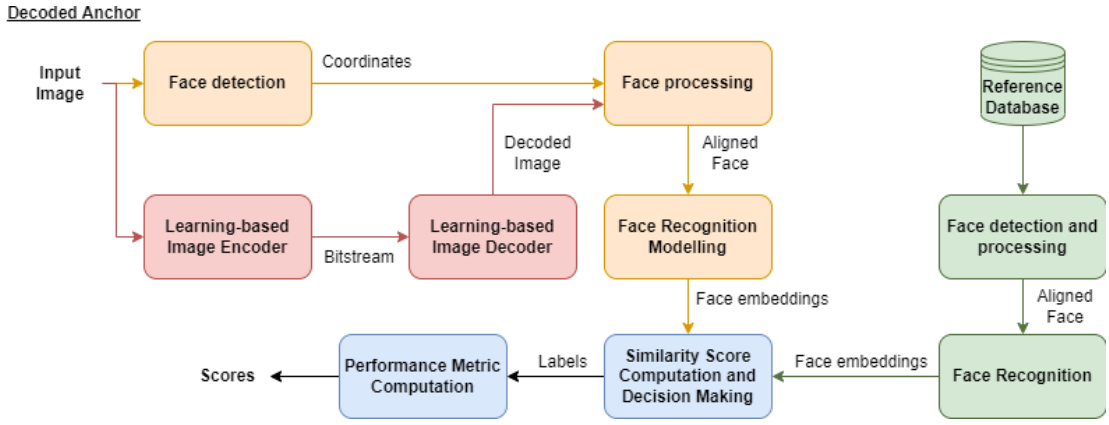

Figure 31 – *Decoded anchor for face verification.*

The decoded anchors architecture does not differ much from the pipeline of the original anchors, and can be divided into two parts:

- **Reference Images Pipeline –** the reference database images represent the original images that are associated with certain identities and are used as reference in face verification tasks. This reference images follow the same pipeline of the original anchors (Section 5.2), in which the images go through the face detection and processing module and a face recognition model to generate the face embeddings.
- **Decoded Images Pipeline –** the original images, which are used to verify the person's face accordingly to the specified identity. The original image is used to obtain the decoded image and for face detection over the decoded image. Afterwards, the same face recognition model of the original anchors is applied to obtain the face embeddings.

The decoded images pipeline can be divided into the following components:

- **Learning-based Image Encoder –** encoder based in neural networks to generate the bitstream for a desired reconstruction quality.
- **Learning-based Image Decoded –** decoder based in neural networks to perform a lossy reconstruction of the image by using the obtained bitstream, generating a decoded image.
- **Face Detection –** same face detection model used in the original anchors to obtain the face coordinates and landmarks of the identified person in the image. In this case, original images before encoding were used to perform face detection and this information should be transmitted to the decoder side to obtain the desired resized and aligned faces. The main reason is that this MSc Thesis studies the impact of the different methods of performing face verification (original, decoded and compressed domain) and the impact of the face detection model is left for future study.
- **Face Processing –** same face pre-processing model of the original anchors, explained in Section 5.2. In this case, the original images are used to obtain the faces coordinates and landmarks to perform pre-processing over the decoded images.
- **Face Recognition Modelling –** same face recognition model of the original anchors (Section 5.2) to generate the face embeddings.

- **Similarity Score Computation and Decision Making –** same module used in the original anchors (described in Section 5.2) to compute a binary label (True/False) for each face verification sample. In the decoded anchors it is used the face embeddings obtained from the original reference images and the decoded images.
- **Performance Metric Computation –** same module used in the Original Anchors (Section 5.2) to obtain the quantitative performance results of the model.

5.4. Face Detection and Image Processing

The input image may contain more than one face and/or other information (e.g., background) that is not related to the face being verified, and thus, can negatively impact the face verification performance; this occurs in the test dataset (LFW) used. Hence, there is the need to only extract the labelled face to obtain better results in the face verification tasks. To accomplish this goal, a face detection model is used to locate all the faces and corresponding landmarks in the image given a specified confidence threshold. From all detected faces, only one corresponds to the labelled face, so other methods, which take into consideration the nature of the dataset (LFW), are implemented to select the desired face. After detecting the labelled face, the image needs to be pre-processed (e.g., crop the face from the image) before being processed by the Face Recognition model.

Face Detection

To detect all faces in the image it was chosen the RetinaFace model [111], which is a deep-learning based face detection model that performs pixel-wise localisation of faces in multiple scales. In this case, it is used a Python implementation of the RetinaFace [112] with pre-trained weights retrieved from [113], which were obtained by training the neural network with the training set of the WIDER FACE dataset [114], [115].

The RetinaFace is a deep learning based cutting-edge facial detector [112], which detects the landmarks and the coordinates of each face in the image.This model will detect candidate faces and assign them a computed confidence score, which determines the likelihood of the candidates being actual faces. If the obtained score is higher than the defined confidence threshold, then the candidate face is considered to be a correctly detected face. The output of the face detection model for each detected face in the image is: the coordinates of the facial area, which corresponds to the top-left point and bottom-right point of a rectangle with the face inside its boundaries; the face landmarks related to the eyes, nose and mouth. Some heuristics are applied to the RetinaFace output and thus to be able to detect the labelled face in the image, as shown in the pipeline in Figure 32.

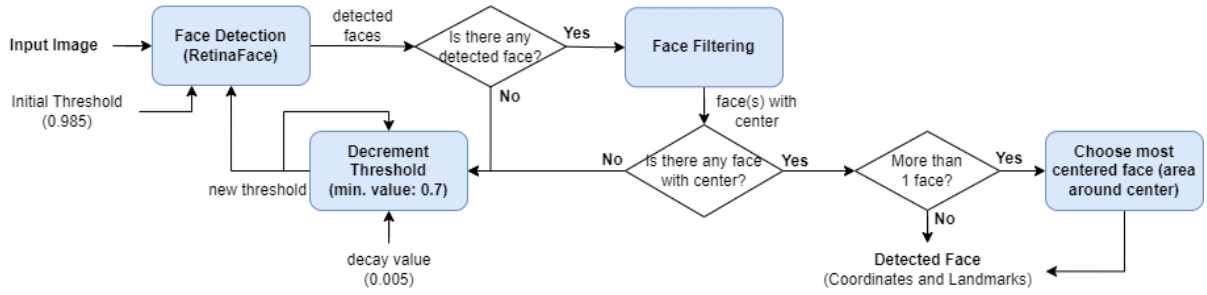

Figure 32 – *Face Detection Pipeline.*

Initially, the confidence threshold is set to be 0.985, which in most cases is enough to detect the labelled face. However, it was noticed that in some images the initial threshold leads to multiple detected faces or zero detected faces. Each problem is solved with the following heuristics:

- **No Faces Detected –** In this case, none of the candidate faces has a confidence score higher than the given threshold. In the used test dataset (LFW), the labelled face is on the center of the image contained in its bounding box, therefore the detected faces are only valid candidates if the center of the image is inside the facial area. Hence, it is considered that no faces are detected if all the candidate faces with confidence score higher than the given threshold do not have the image center in their facial area. To solve this issue, it was chosen to use a constant decay of the threshold, in which the threshold value is decremented by 0.005. This process is repeated until finding at least one face in the image or reaching a minimum defined threshold value of 0.7, from which it is assumed that there is no face present in the image.
- **Multiple Detected Faces –** Images may contain more than one face, and in some cases, it can lead to multiple faces being detected by the face detection model, as shown in Figure 33. This is a problem since in the assessment of the anchors solutions each test image (LFW) is labelled with only one identity. From an analysis of the selected dataset (presented in Section 5.9.1) to evaluate the model, it was observed that the labelled face is positioned mainly in the center of the image when comparing with the other faces. Hence, it was chosen to select the most centered face, by first excluding the faces that do not contain the center of the image inside the facial area as in Figure 33. If only one face is left after exclusion, then the remaining face is selected as the labelled face, as in Figure 33 (bottom). Otherwise, if multiple faces include the center of the image inside their facial area, then it is necessary to perform extra computation to select which face is most centered in the image. For this criterion, the area of the maximum squares around the center of the image and between the facial boundaries of each face were used, as shown in Figure 33 (top). Each square represents the maximum area of each face around the center of the image when it is made an uniform expansion from the center to the face boundaries. This means that the face with the square with highest area is the most centered face in the image, i.e., it is the face which corresponds to the image identity label in the test dataset (LFW). The minimum distance from the center to the facial boundaries is computed, which corresponds to half-length of the side of the square. It is used the minimum distance to obtain the maximum square around the center, because it is the maximum expansion of the square from the image's center such that the square is still inside the facial boundaries. After obtaining the minimum distance (***d***), the area of the square around the center (***A***) for each face is given by $\boldsymbol{A} = (\boldsymbol{d} * 2)^2$. The selected face will correspond to the face which contains the largest described area around the center, as shown in Figure 33 (top). From the area equation, it can be observed that it would be enough to use the computed minimum distance to select the most centered face. Although, it was decided to use the area as the decision criterion, since it better illustrates, visually, the most centered face situation.

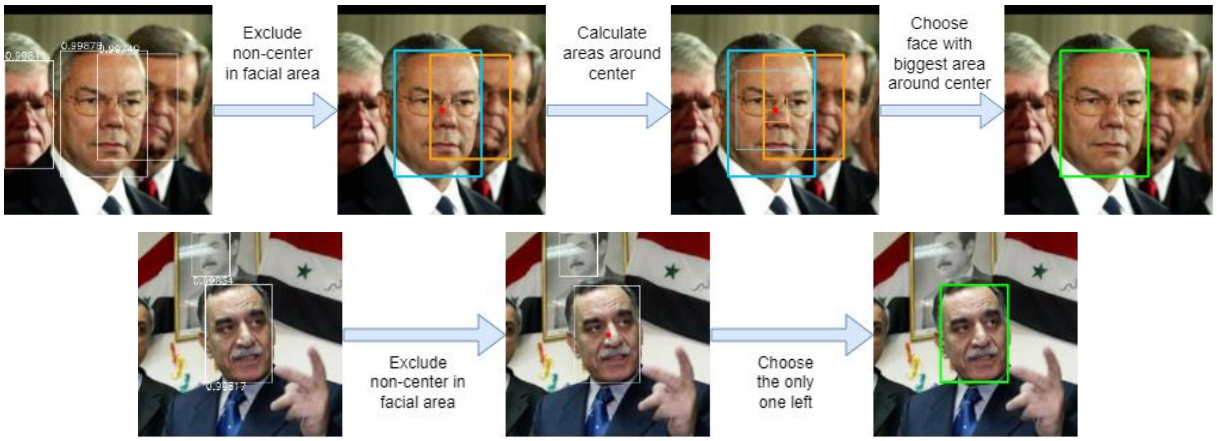

Figure 33 – *Examples of the processing performed when multiple faces are detected for the initial threshold 0.985. In the top, the center of the image is contained in multiple facial areas. In the bottom, the center of the image is only inside one facial area. Images belong to the LFW dataset.*

Image Processing

The face detection procedure allows to locate the labelled face in the image, returning as output the coordinates of the facial area and the face landmarks. However, the detection model only allows to have information of the position of the face, there is still the need to separate the face from the rest of the image. This task is performed by cropping the image with the facial area coordinates previously obtained, as shown in the first step of the pipeline of Figure 34.

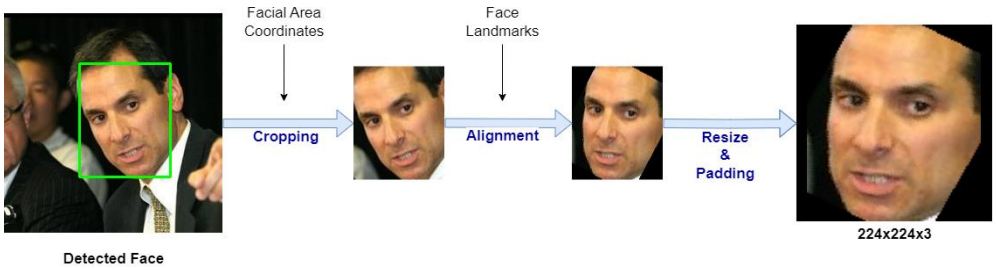

Figure 34 – *Image processing pipeline with an output image with dimension 224x224x3. Example image retrieved from the LFW dataset.*

The usage of face alignment before the face recognition model is applied has shown to improve performance in face recognition tasks [112]. Hence, after cropping, an alignment procedure implemented in [112] is applied, which will use the eyes and nose landmarks to perform a 2D affine transformation of the face, as shown in Figure 34. In this way, the eyes and noses of all input faces are in similar locations.

Moreover, the face recognition model has a fixed input dimension of (3,224,224), and thus some adaptation is needed since the faces detected can have an arbitrary size. The adaptation process is composed of two parts: resize and padding. The resize consists in increasing the input image spatial dimensions, first, it is selected the largest dimension (width or height) of the face region to determine how many pixels (margin) are needed to obtain 224, e.g., if the face's spatial dimension is (180,120), then the margin is 224-180=44. Then, resize is performed with the target resolution of the original image plus this margin in both height and width, this will lead to an image with spatial dimensions equal to (224, 164). Since the image has not the desired spatial dimensions (224,224), padding (new areas are black) is applied over the lowest dimension to obtain the final dimension of (224,224,3), as shown in

Figure 34. This procedure allows to minimize the amount of padded area while keeping the deformation to a minimum.

Before the face recognition model, it is subtracted a pre-defined mean RGB vector with the values [131.0912, 103.8827, 91.4953], according to [118]. Each value of the vector will be uniformly subtracted to all values of one channel of the RGB image (Red: 131.0912, Green: 103.8827, Blue: 91.4953).

5.5. Face Verification

Deep-learning face verification is based on face feature extraction, usually mapping the face to an appropriate lower-dimensional space, to solve the binary classification problem. The feature extraction output is denominated as a face descriptor, whose size and shape may vary between face verification models. The face descriptor could be used directly to perform comparisons with other faces descriptors, but currently, many models have an extra fully-connected layer as the output layer to perform dimensionality reduction of the face descriptors, obtaining non-linear face embeddings which are used to compare identities. In this Thesis, it was selected a model based on VGGFace2 [116], ArcFace [117] and the ResNet50 to compute the face embeddings.

5.5.1. ResNet50 Architecture

Residual Neural Networks (RNN or ResNet) have shown state-of-art performance in some image feature extraction tasks, as mentioned in Sections 2.3, 4.1 and 4.2. Hence, it was selected the ResNet50 architecture, like in the VGGFace2 and ArcFace models. A PyTorch (Python) implementation of the VGGFace2 model [118] based in [116] was used. The ResNet50 network takes as input RGB images with spatial dimension of 224x224 and is mainly composed by four types of residual blocks.. Residual blocks are composed of residual units with three convolutional layers, each layer followed by 2D Batch Normalization (BN) that is only applied for training. The number of residual units varies between residual blocks, residual blocks 1 and 4 have 3 residual units, residual block 2 has 4 residual units and residual block 3 has 6 residual units. Downsampling in the residual blocks is performed by convolutional layers but only to the residual block input, which is added to the residual block's first residual unit output, as shown in Figure 35 in which the downsampling path is represented by a dashed line. However, downsampling of the residual block input only occurs in residual blocks 2, 3 and 4, since in residual block 1 only the number of channels of the input is adjusted, keeping the spatial size. Average pooling is applied after the last residual block of the ResNet50, the output is then flattened and fed to a FC layer with 8631-d output when used for face identification. For face verification, the 8631-d FC layer is replaced by a 2048-d FC layer which will generate the face embeddings, based in [116] and in [117]. More details of the face verification model architecture are presented in Table 4 and Figure 35.

Table 4 – *Structure of the ResNet50 architecture. Each residual block is composed of residual units that contain 3 convolutional layers each, e.g., residual block 1 has 3 residual units.*

	Input Size	Block	Root	Residual block 1	Residual block 2	Residual block 3	Residual block 4	
ResNet50	(3,224,224)	**Output Size (spatial)**	56x56	56x56	28x28	14x14	7x7	avg. pool, 2048-dim fc [output: 1x2048]
		Arch.	[7x7, 64] 3x3 max pool, s=2	$\begin{bmatrix} 1x1, 64 \\ 3x3, 64 \\ 1x1, 256 \end{bmatrix}$ x3	$\begin{bmatrix} 1x1, 128 \\ 3x3, 128 \\ 1x1, 512 \end{bmatrix}$ x4	$\begin{bmatrix} 1x1, 256 \\ 3x3, 256 \\ 1x1, 1024 \end{bmatrix}$ x6	$\begin{bmatrix} 1x1, 512 \\ 3x3, 512 \\ 1x1, 2048 \end{bmatrix}$ x3	

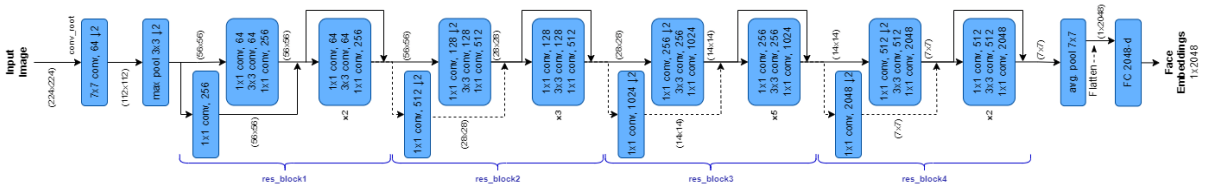

Figure 35 – *Framework illustration of the ResNet50 final architecture for face verification. Dashed lines represent downsampling paths of the residual blocks input which is added to the residual block's first residual unit output.*

5.5.2. ResNet50 Training

The ResNet50 implementation in [118], based in [116], has pre-trained weights available with the 8631-d FC layer for face identification. However, there are no implementation details about the training process, such as the number of epochs to train and fine-tune the model. Therefore, a training procedure for the ResNet50 was implemented. This solution could also be used as starting point to train the novel compressed domain face verification solution.

During the training procedure, the ResNet50 model has 2 phases: 1) training with a model which has a FC layer with 8631-d output (ResNet50_8631-d) as the output layer, for face identification, 2) the 8631-d FC layer is replaced by a randomly initialized 2048-d FC layer, to generate the face embeddings, obtaining the ResNet50 final architecture to be trained as shown in Figure 35. The ResNet50 with 2048-d embeddings is then fine-tuned for face verification, but only the FC layer is updated.

Train Dataset

The ResNet50 is trained with the VGGFace2 dataset [119] containing around 3.1M RGB images, converted to *png* format, from 8631 different identities. No face detection and alignment are performed to the images, since the samples already present the labelled face centered and with a small percentage of background information. For training, the whole dataset is used as training set, no validation is performed since in [118] it is used all the 8631 identities.. For fine-tuning the model, the dataset is split in train set and validation set at identity level, with around 10% of identities being selected for the validation set. This percentage is rounded so that during training, all batches have the same number of samples. Therefore, the train set has 7768 identities (~2.8M images) and the validation set has 863 identities (~310k images).

The training images are pre-processed before entering the network as performed in [116][118]. The dataset images have an original spatial size of 112x112, being resized with bilinear interpolation to a size of 256x256. The resized image is then randomly centered cropped to a spatial size of 224x224, i.e., it is randomly chosen a point (x,y) in the image to be the center of the cropping box. The cropping box has a spatial size of 224x224 with center in (x,y), hence the values of (x,y) must be chosen such that the cropping box is contained in the image borders. Afterwards, a grayscale conversion is applied

for some cropped images, with 20% probability. Finally, a pre-defined RGB vector [131.0912, 103.8827, 91.4953] is subtracted to the image, as described in Section 5.4. The pre-processing procedure is illustrated in Figure 36.

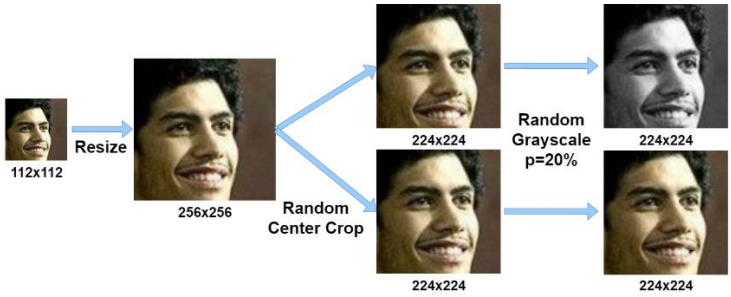

Figure 36 – *Illustration of the pre-processing for training the ResNet50. Face image obtained from the VGGFace2 dataset [119].*

Train ResNet50

The ResNet50 training is performed on the ResNet50_8631-d architecture, which has a FC layer with 8631-d output, with the goal of training the model for face identification. The whole VGGFace2 dataset is used to train the ResNet50 (3.1M images) and thus, no validation is performed. The model is trained with the Cross-Entropy Loss (CEL) with mean reduction, based on the softmax function. The CEL function is represented by (3), in which x is the input (output of the ResNet50) (8631-d vector), y is the target labels, N is the number of images in a batch, C is the number of classes and w is a pre-defined weight of the class. The loss of each image (l_n) is given in (3), where $\mathrm{y_n}$ is the one-hot vector of the input $\mathrm{x_n}$, meaning that $\mathrm{y_{n,c}}$ is the one-hot vector value and $\mathrm{x_{n,c}}$ is the input value of image n for class c. The softmax expression in (3) transforms the input values into probabilities, in which $\mathrm{x_{n,i}}$ represents the input value of image n for class i. The CEL function measures the performance of a classification model whose output is a set of probabilities between 0 and 1. In this case, the weight component is not used and thus is 1 for all classes. The model is trained with the mini-batch SGD optimizer, with mini-batches of 64 samples and a learning-rate of 0.1 for 10 epochs.

$$\boldsymbol{CEL}(x,y) = \frac{\sum_{n=1}^{N} l_n}{N}, \quad l_n = -\sum_{c=1}^{C} w_c \, log\big(softmax(x_{n,c})\big) \, y_{n,c}, \quad softmax\big(x_{n,c}\big) = \frac{e^{x_{n,c}}}{\sum_{i=1}^{C} e^{x_{n,i}}} \tag{3}$$

Fine-tune ResNet50

After training the ResNet50, the 8631-d FC layer is replaced by a randomly initialized 2048-d FC layer, which will output the face embeddings. Since this layer is initialized with random weights, they must be learned for the face verification task. Given that the rest of ResNet50 has already been trained, only the 2048-d FC layer was updated while fine-tuning. For this case, the VGGFace2 dataset is split into train set (90%) (~2.8M images) and validation set (10%) (310k images). The validation loss is used by the learning-rate scheduler, which decreases the learning-rate by 1/2 if there is a validation loss plateau (stable loss during some epochs). However, in practice, learning-rate reduction did not occur during the fine-tuning of the ResNet50 due to absence of validation loss plateaus.

The model is fine-tuned with the ArcFace Loss [117], which is based on a normalized hypersphere transformation of the face embeddings. This loss function uses the generated face

embeddings for multi-classification problems, such as face identification. However, the ArcFace Loss can be applied for face verification tasks since it promotes face embeddings that have higher intra-class similarity and higher inter-identity disparity. The ArcFace Loss uses an extra FC layer that takes the face embeddings as input and the output dimension corresponds to the number of train identity classes (7768), as shown in Figure 37. Since this MSc Thesis objective is face verification, it is not intended to assess the performance of this extra layer in face identification, being only used for fine-tuning the 2048-d FC layer.

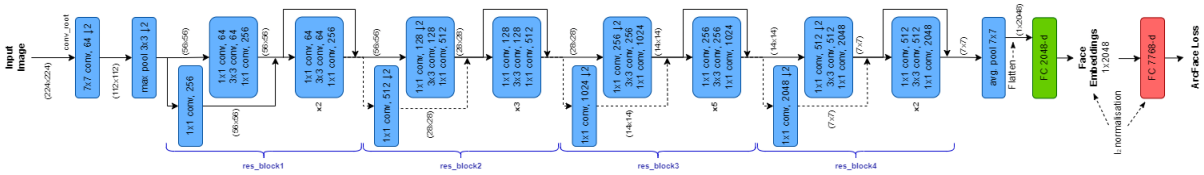

Figure 37 – *Framework illustration of the ResNet50 final architecture with the extra FC layer for fine-tuning the 2048-d FC layer with the ArcFace Loss. (Green) ResNet50 FC layer to be updated. (Red) Extra FC layer for fine-tune with ArcFace Loss. Dashed lines represent downsampling paths of the residual blocks input which is added to the residual block's first residual unit output.*

The ArcFace Loss is based in the softmax loss function given by (4), being N the number of images in a batch and n the number of train identity classes. Additionally, $x_i \in \mathbb{R}^d$ correspond to the face embedding of the i-th image with y_i identity, where d is the face embedding dimension (=2048), $W_j \in \mathbb{R}^d$ is the j-th column of the ArcFace extra FC layer weight matrix $W \in \mathbb{R}^{d \times n}$ and $b_j \in \mathbb{R}^d$ is the bias term.

$$\boldsymbol{Softmax\ Loss}(x, y, W, b) = -\frac{1}{N}\sum_{i=1}^{N} log\left(\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}}\right) \quad (4)$$

As described in [117], the bias term is fixed to 0 ($b_j = 0$) and it is performed a hypersphere transformation given by (5), in which θ_j represents the angle between weight j-th column W_j^T and the face embedding x_i.

$$W_j^T x_i = \left\|W_j^T\right\| \|x_i\| cos\ (\theta_j) \quad (5)$$

The face embedding (x_i) and the weights (W_j) are l_2-normalised, the face embedding is then re-scaled to a value of s, leading to $\left\|W_j\right\|$=1 and $\|x_i\|$=s. The normalization and re-scale steps make the face embeddings learn to be distributed in a hypersphere of radius s, by making predictions that are only dependent on the angle between the weights and the embeddings. Additionally, it uses a margin penalty (m) between x_i and W_{y_i} for intra-identity compactness and inter-identity discrepancy, obtaining the ArcFace Loss equation given by (6).

$$\boldsymbol{ArcFace\ Loss}(x, W, y) = -\frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{e^{s \cdot cos\ (\theta_{y_i} + m)}}{e^{s \cdot cos\ (\theta_{y_i} + m)} + \sum_{j=1, j \neq y_i}^{C} e^{s \cdot cos\ (\theta_j)}}\right) \quad (6)$$

The ArcFace Loss promotes face embeddings that have higher intra-class similarity and higher inter-identity disparity by updating the 2048-d FC layer and the ArcFace extra FC layer. Hence, the ArcFace Loss is also optimizing the 2048-d FC layer for face verification, even thought that the ArcFace

Loss is projected for face identification. It is used the default values of the ArcFace Loss parameters s and m defined in [117] (s=64, m=28.6º).

The model is fine-tuned with the mini-batch SGD optimizer, with mini-batches of 64 images and a learning-rate of 0.005 for 5 epochs.

5.6. Image Codecs

The decoded anchors use decoded images (lossy reconstructions) of the person whose identity is being verified as input. This Section describes the image codecs that were selected for the compression. As shown in Sections 3.2 and 3.3, deep-learning based image codecs have shown to be capable of competing against state-of-art conventional image coding solutions. For this reason, the following two deep-learning based image codecs were used: Ballé 2018 with hyperprior (Balle_2018) [82]and Cheng 2020 with attention (Cheng_2020) [121].

The pre-trained models of the Balle_2018 and Cheng_2020 codecs implemented by CompressAI in Python [122], [123] are used to encode and decode the images. The models were trained for 4-5M iterations with randomly selected images from the Vimeo90k dataset, the inputs were cropped to the shape (256,256) before compression [123]. The CompressAI group have made available these models with different optimization loss functions during training and different target qualities (which correspond to rate-distortion operating points). The different optimizations (MSE optimization and MS-SSIM optimization) and qualities lead to several decoded anchors.

Ballé 2018 with Hyperprior

The Balle_2018 codec is an end-to-end optimized image coding solution, which uses variational autoencoders and includes a hyperprior as side information to efficiently exploit the latent representation spatial dependencies [82].. After encoding the image, two latent representations are obtained: latent ($\hat{y}$) and the hyper-latent ($\hat{z}$). After decoding, the hyper-latent is considered side-information and only used to obtain probability models for the entropy decoder of the $\hat{y}$ latent. Hence, the bitstream has two parts that are created after fully compressing the input image, the main part of the bitstream, which contains most of the image compressed information, and an extra part of the bitstream, which has a much smaller percentage of the size of the entire bitstream.

The CompressAI group made available pre-trained models of the Balle_2018 codec for the MSE and MS-SSIM optimization losses. Additionally, for each loss, the codec is trained for 8 different target qualities (each quality represents a rate-distortion operating point), resulting in a total of 16 image coding solutions for the Balle_2018 codec. Each of these models, designated as Balle_2018_mse_x or Balle_2018_ms_ssim_x (x refers to the corresponding quality), are used in the Decoded Anchor solution.

Cheng 2020 with Attention

The Cheng_2020 codec is a learning-based image coding solution with discretized Gaussian Mixture Likelihoods and attention models. The codec uses Gaussian Mixture Likelihoods to parametrize the distribution of the latent representations, obtaining more flexible and accurate entropy models. Moreover, it employs attention modules at both encoder and decoder to achieve higher coding efficiency

[121]. Like in the Balle_2018 codec two latent representations (latent and hyper-latent) are generated and consequently the bitstream has two parts that represent the image in the compressed domain.

The pre-trained models of the Cheng_2020 codec are available (by the CompressAI group) in two different optimization losses: MSE and MS-SSIM. For each optimization loss the codec is trained in 6 different qualities that represent different estimated rate-distortion ratios. Consequently, 12 different pre-trained image coding solutions are available for the Cheng_2020 codec, which can be designated as Cheng_2020_mse_x or Cheng_2020_ms_ssim_x (being x the used corresponding quality).

Image Encoder

The image encoder implementation requires the input image to be an RGB image with H and W the spatial dimensions and the pixel values between [0,1]. Additionally, the H and W are expected to be at least 64 and a power of 2 [123]. Therefore, the image is pre-processed by padding with zeros (constant black colour) to increase the spatial dimensions. The padding process is performed uniformly across both sides of the image, e.g., if the image needs to be padded horizontally by a value of 20 pixels, then the image will be padded 10 pixels to the right and 10 pixels to the left.

From the bitstreams, the rate for each compression is calculated according to (7) in bits per pixel (bpp). Higher quality of the pre-trained models corresponds to larger rate values.

$$\boldsymbol{rate} = \frac{bitstream\ total\ size}{input\ image\ nº\ of\ pixels} \tag{7}$$

Image Decoder

The bitstreams obtained by the encoder are used as inputs of the decoder, resulting in a lossy reconstructed image. Since padding was performed over the image before entering the encoder, then there is the need to crop the image to its original spatial dimensions. Consequently, the original spatial dimensions need to be transmitted to the decoder.

The final decoded and cropped image is then used as input for face verification in the Decoded Anchor solutions. Since the procedure of decoding images for large datasets may require long time, it was decided to save all decoded images in *png* format as well as the obtained PSNR and MS-SSIM distortion metric values for each image, instead of executing the coding process for each test.

5.7. Similarity Score Computation and Decision Making

The face embeddings computed by the Face Recognition model (ResNet50) represent features that were extracted from the facial image. When performing face verification, it is desirable to compare the face embedding of the person whose identity is being verified against one or more face embeddings used as identity references. This comparation is done through a similarity score function, to perform the binary classification for a given threshold.

Similarity Score Computation

The selected similarity score function has a strong influence in the results that are obtained, since the scores it generates will be used directly to perform the identity verification. In this Thesis, the Cosine Similarity Score (COS), given by (8) ($\vec{a}$ and $\vec{b}$ are face embeddings) was used to obtain a

numerical comparison (-1 to 1) between two face embeddings. This function computes the relative difference of the face embeddings (vectors) which are used as input. The score is computed between a probe face embedding, which represents the face which identity is being verified, and an identity reference face embedding, that represents a face that is being used as reference for the identity. Although is not the case in this work, when there are multiple identity references, there are several options, namely a reference mean face embedding could be computed, a similarity score could be computed for each reference and then a Decision Making module that takes as input all scores could be employed to perform the final decision.

$$\textbf{\textit{Cosine Similarity Score}}(\vec{a},\vec{b}) = \frac{\vec{a}\cdot\vec{b}}{||\vec{a}||\cdot||\vec{b}||} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n {a_i}^2}\sqrt{\sum_1^n {b_i}^2}} \quad (8)$$

Decision Making

The similarity score obtained for a pair of face embeddings quantifies the similarity between the face of the person being verified and an identity reference face. However, it is not the final decision for the face verification problem (which is a binary classification problem). The decision making is performed with a threshold value that will classify the pair as True or False, as follow:

- **True –** the person's identity is verified as correct, i.e., the person's face when compared against a reference face results in a match. This occurs when the similarity score has a larger value than the specified threshold.
- **False –** the person's identity is verified as incorrect, i.e., the person's face and the used identity reference face do not match. This situation occurs when the similarity score is smaller than the threshold.

The threshold plays an important role on the decision making process, as illustrated in Figure 38, so multiple thresholds may need to be used for a more robust performance evaluation, and thus allow to assess the model's performance at different operating points.

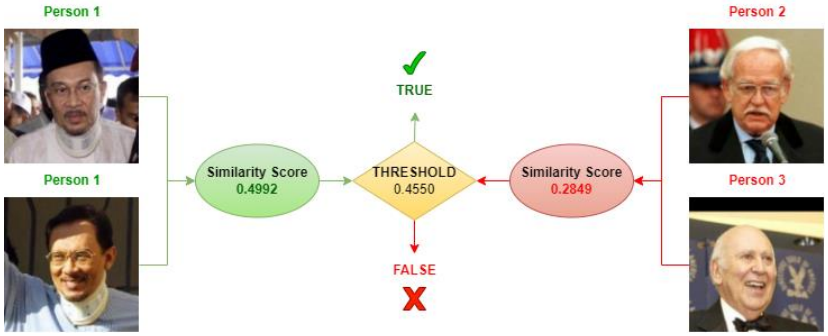

Figure 38 – *Illustration of the decision making process given a threshold of 0.4550 and two pairs of faces. One pair has the two images with the same identity (left) and the other pair has two images with different identities (right). The pairs were retrieved from the LFW dataset testing pairs and use the original images from the LFW dataset.*

5.8. Performance Metric Computation

A wide range of metrics with different characteristics can be applied to evaluate face verification performance. The selected performance metrics are extremely popular to evaluate face verification systems, and can be divided into two main groups:

- **Intermediate Performance Metrics –** Their values are computed in an intermediate step to calculate the final performance metrics, which will be used to evaluate the models. The values of these metrics are not directly presented in the models' assessment. This group include:
  - **False Acceptance Rate (FAR) –** Quantifies the acceptance of impostors, i.e., the number of false identity verifications that were classified as True over the total number of testing pairs. Also known as False Positive Rate (FPR) and expressed in percentage.
  - **False Recognition Rate (FRR) –** Quantifies the incorrect rejection of genuine users, i.e., number of True identity verifications that were classified as false over the total number of pairs. Also known as False Negative Rate (FNR) and expressed in percentage.
  - **True Acceptance Rate (TAR) –** Quantifies the correct acceptance of genuine user, i.e., the number of true identity verifications that were correctly classified as true over the total number of pairs. Also known as True Positive Rate (TPR) and expressed in percentage.
  - **Receiver Operating Characteristics (ROC) Curve –** Curve that represents the TAR vs FAR performance metrics, as shown in Figure 39. This curve allows for an analysis of the trade-off between TAR and FAR for a range of operating points (thresholds), which in some applications is important due to the different practical consequences that a false acceptance may have.
- **Final Performance Metrics –** These metrics are used to make the final evaluation of the models for face verification. The following metrics were defined based on previous work on face verification:
  - **TAR(@FAR=0.001) –** TAR value for a fixed FAR value of 0.001.
  - **TAR(@FAR=0.01) –** TAR value for a fixed FAR value of 0.01.
  - **Area Under the Curve (AUC) –** Measures the area under the ROC curve, as shown in Figure 39. The values of this metric are in the interval [0,1], being 1 the best performance, in which the TAR is 1 for all threshold values.
  - **Equal Error Rate (ERR) –** Represents the value of FAR where the FAR and FRR metrics are equal. Since the TAR metric is equal to 1-FRR, the ERR value can also be obtained by the FAR value in the intersection of the ROC curve with the line that connects the points (0,1) and (1,0) as shown in Figure 39. The smaller the ERR value the better performance is achieved for this metric.

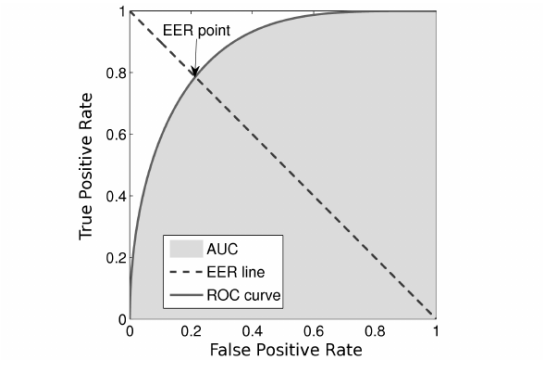

Figure 39 – *Illustration of the ROC curve, the AUC and the ERR metrics [127]. In face verification, True Positive Rate (TPR) is the same as True Acceptance Rate (TAR) and False Positive Rate (FPR) is the same as False Acceptance Rate (FAR).*

5.9. Performance Evaluation

The performance evaluation of the anchors represents the baseline results for the novel compressed domain face verification model. Each anchor is evaluated using the performance metrics described in Section 5.8. The baseline results are obtained for the following anchors:

- **Original_Anchor –** Original anchor solution described in Section 5.2.
- **Balle_2018_mse_Qn –** Decoded anchor solution described in Section 5.3, with the Balle_2018 codec [82] optimized for the MSE loss function and model n (1 to 8) for each target quality.
- **Balle_2018_ms_ssim_Qn –** Same as previous decoded anchors solution but trained with MS-SSIM as loss function.
- **Cheng_2020_mse_Qn –** Decoded anchor solutions described in Section 5.3, with the Cheng_2020 codec [121] optimized for the MSE loss function and model n (1 to 6) for each target quality.
- **Cheng_2020_ms_ssim_Qn –** Same as previous decoded anchors solutions but trained with MS-SSIM as loss function.

5.9.1. Datasets

The anchors solutions are evaluated for the Labelled Faces in the Wild (LFW) dataset [109][128], that was selected by its popularity in face verification performance evaluation. The LFW dataset contains labelled face images in non-controlled environments, across a wide range of conditions (position, lightning, camera quality, etc.) that represent everyday life situations [109]. The dataset is composed of 13233 images across 5749 different identities as shown in Table 5.

Datasets	#Identities	#Images	Image Size	Image Format	Publicly Available
LFW	5749	13233	250x250	jpg → png	Yes

Table 5 – *LFW dataset information.*

The dataset's images are only available in *jpg* format, hence the datasets do not totally represent original images, since they have been JPEG encoded. To simulate original images, it was converted all the JPEG images to the lossless *png* format, i.e. JPEG decoding followed by PNG encoding. The LFW dataset have available 6k labelled testing pairs for face verification (with 3k positive pairs and 3k negative pairs), each pair contains two images (probe face and identity reference face) and the label for the verification task (True/False).

5.9.2. Face Verification Protocol

As mentioned in Section 5.7, there are many protocols to evaluate performance in face verification tasks. Since the cosine similarity score function is used to compute the similarity between probe faces and reference faces and takes as input two vectors (face embeddings). However, there can be multiple probe images and multiple reference images for one identity verification. In these cases, different verification protocols can be used, e.g., mean face embeddings, multiple similarity scores used as inputs in a complex decision-making module, etc. The used dataset (LFW) has its own face verification testing protocol (LFW2 protocol) as described in [109], that consists on several triplets of images which include one probe image, one identity reference image and the ground-truth (True/False), as in Figure 38.

In all solutions, the similarity scores are computed for each pair and stored. The TAR and FAR values to generate the ROC curve and corresponding AUC are computed with *Python* functions from the *sklearn.metrics* library, in which the thresholds are chosen by the ROC curve function. The TAR@FAR=0.001 and TAR@FAR=0.01 are obtained by using the TAR and FAR values of the ROC curve. The ERR value is calculated with the FAR and FRR, in which the FRR values are computed by 1-TAR, in which ERR corresponds to the value where FAR is equal to FRR.

5.9.3. Experimental Results

The face verification performance metrics results for the implemented anchor solutions, given by bitrate-metric plots, are shown in Figure 40. For each decoded anchor solution (Balle_2018_mse_Qn, Balle_2018_ms_ssim_Qn, Cheng_2020_mse_Qn, Cheng_2020_ms_ssim_Qn), is computed the mean bitrate values of the probe images used in the test pairs. In the decoded anchors the probe face is a decoded image and the reference identity face is an original image; the reference images are considered original images and thus not used to calculate the rate. In the original anchor both probe and reference faces are original images, and since no bitrate can be computed for this case it is used a dashed horizontal line across all rates to represent the original anchor performance metrics values in Figure 40. Additionally, it was computed the distribution of the similarity scores obtained for the Balle_2018_mse_Q1, Balle_2018_mse_Q8 and original_anchor anchors for a more complete analysis of the performance of the face verification anchor solutions, presented in Figure 41.

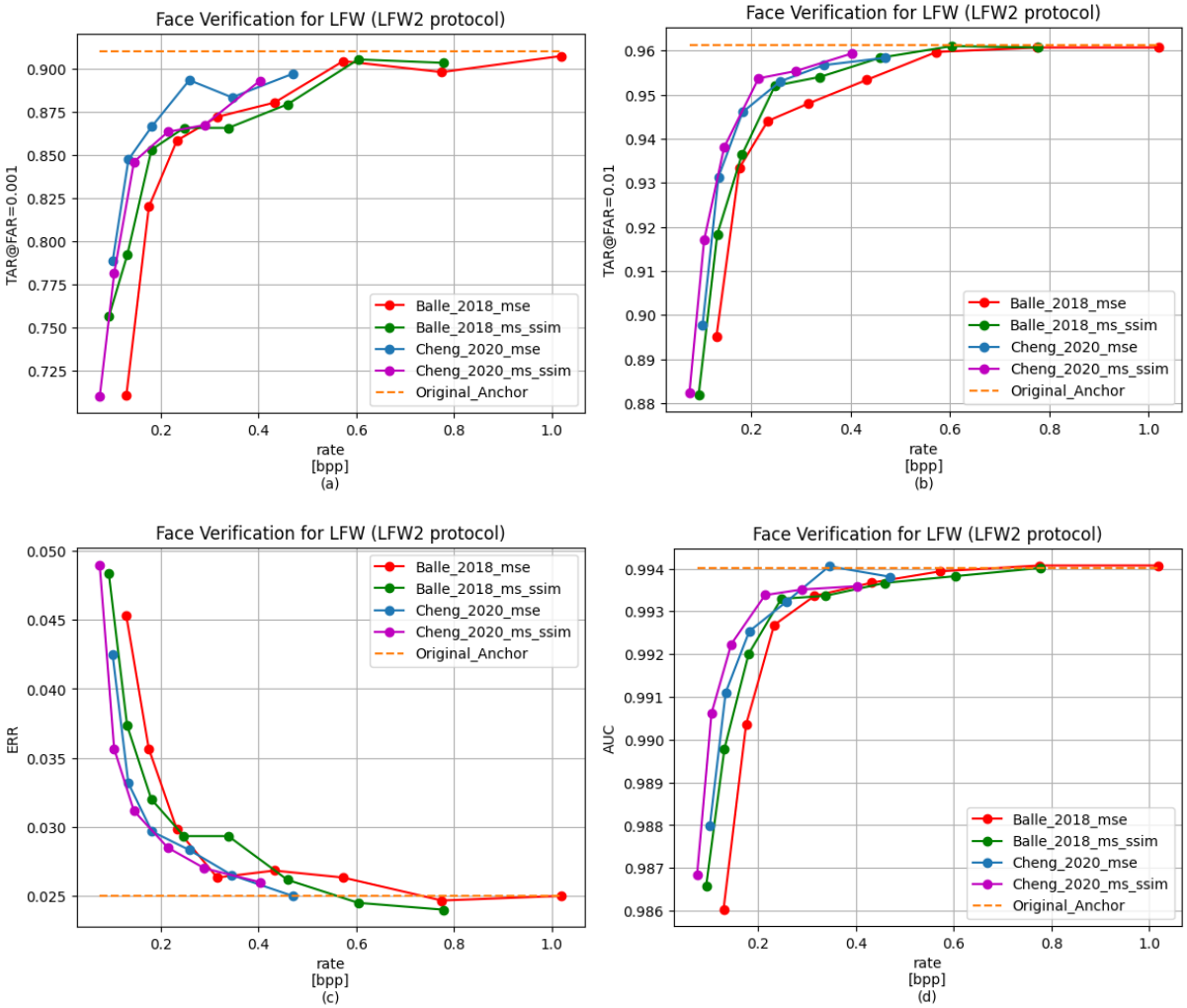

Figure 40 – *Face verification results for all anchors (ResNet50) for the different mean rate for the test images of the LFW dataset with the LFW2 protocol (6k pairs). (a) TAR@FAR=10^{-3}; (b) TAR@FAR=10^{-2}; (c) ERR; (d) AUC.*

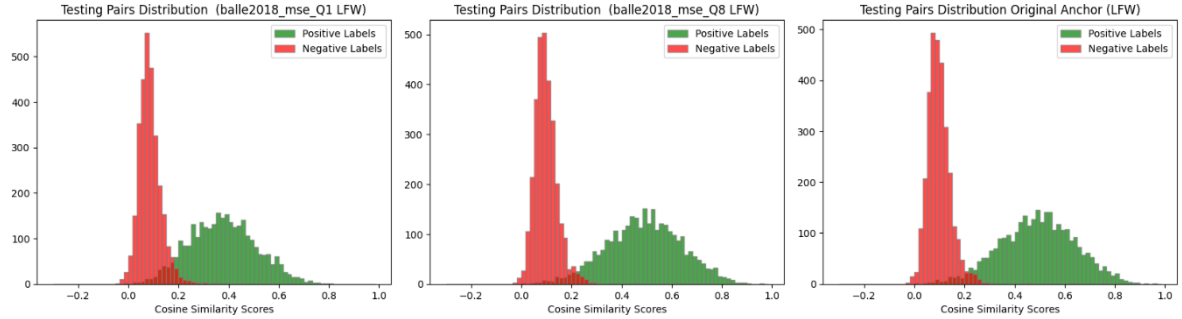

Figure 41 – *Similarity Scores distribution for the LFW dataset (6k pairs). Red: Negative labels (False); Green: Positive labels (True). (left) Decoded anchor with Balle_2018 codec, MSE loss and quality 1; (middle) Decoded anchor with Balle_2018 codec, MSE loss and quality 8; (right) original anchor.*

From Figure 40 and Figure 41 it can be observed that the anchor solutions present good results for the selected performance metrics, and is possible to visualize the increase in performance with the increase of the bitrate, as expected. Some other conclusions can be driven from the obtained results:

- **Performance results for TAR@FAR=0.01, ERR and AUC –** the TAR@FAR=0.01, ERR and AUC metrics do not have a major increase in the performance while increasing the bitrate, being contained

in the intervals [0.88, 0.97], [0.05, 0.02] and [0.986, 0.995] respectively. However, the AUC values are all close to the ideal value 1, which shows high performance results for all anchors in this metric. Likewise, the ERR and AUC also show high performance results. This means that these metrics are not very discriminative, especially compared to TAR@FAR=0.001.

- **Performance results for TAR@FAR=0.001 –** the experimental results for the TAR@FAR=0.001 metric presents low to high performance for the given bitrates. As expected, it clearly illustrates that for lower bitrates, where coding artifacts are more present, the percentage of true identities that are correctly verified for a fixed FAR (0.001) is lower than for higher bitrates. The FAR values simulate a desired maximum value for the percentage of false identities that are verified as true, being more demanding for TAR@FAR=0.001 than TAR@FAR=0.01. These results are expected since for lower bitrates there is a larger amount of lost information while encoding the images, resulting in a larger presence of artifacts in the decoded image which can lead to a worst performance, as verified in Figure 40 (a). Additionally, in the distributions presented in Figure 41, it is noticeable that the lowest bitrate has more difficulty in separate the positive and negative labels than the original anchor and the decoded anchor with highest bitrate.
- **Decoded Anchors' results close to the Original Anchor in higher bitrates –** in Figure 40 it can be observed that some slight performance gains of the decoded anchors are similar or even better than the original anchor in higher bitrates, which was not expected since the original images do not present coding artifacts. However, the LFW dataset does not contain actual original images, as mentioned in Section 5.9.1, and they are converted from JPEG format to the PNG format, which means that the original images will also present artifacts. When these images are encoded and decoded by deep-learning based codecs, the artifacts can sometimes better reconstruct some facial features, allowing higher performance results in some cases.
- **Cheng_2020 vs Balle_2018 –** from the results presented in Figure 40, it is possible to observe that the Cheng_2020 anchor solutions, in both optimization losses (MSE and MS-SSIM), have higher performance than the Balle_2018 anchor solutions in almost every bitrate in all metrics. For lower bitrates the difference in performance between the two codecs is larger, while for higher bitrates the performance of the decoded anchors become similar. Therefore, the Cheng_2020 codec is capable of better reconstruct the facial features than the Balle_2018 codec, especially in lower bitrates. However, the difference in performance is more noticeable in the TAR@FAR=0.001 metric, e.g., with the rate of ~0.16 bpp both Cheng_2020 anchor solutions (MSE and MS-SSIM) have a TAR@FAR=0.001 of ~0.850 while the Balle_2018 anchor solutions MSE and MS-SSIM have a value of ~0.71 and ~0.79.
- **MSE vs MS-SSIM (Optimization losses) –** in almost every bitrate the codecs optimized with MS-SSIM loss have higher performance than their corresponding codec optimized for MSE loss, i.e., the Cheng_2020 codec optimized for MS-SSIM has higher performance than the Cheng_2020 codec optimized for MSE (same for the Balle_2018 codec). This difference in performance is more significant in lower bitrates, as for higher bitrates the performance between decoded anchor solutions with different optimization losses becomes smaller. This means that the MS-SSIM optimization loss has higher capability for decode facial features than the MSE optimization loss. However, in Figure

40 (a) it can be seen that for the TAR@FAR=0.001 metric the previously made observation it is not fully accurate, since for the Cheng_2020, codec the MSE optimization has higher performance than the MS-SSIM optimization.

6. Compressed Domain Face Verification

This chapter presents the design, implementation and assessment of the novel compressed domain face verification solution, which is the main objective of this MSc Thesis.

6.1. Context and Objectives

The face verification task usually follows one of two schemes, as presented in Section 5: ATC scheme: face verification is applied over the original image, the obtained information and image can then be compressed (which corresponds to the original anchor); CTA scheme: where the original image is compressed, usually for storage/transmission, and then it is obtained a lossy reconstruction of the image after decoding. The decoded image is then compared to original identity reference images to perform face verification (Decoded Anchors). With the improvement of image quality and resolution, images are being represented with larger amount of data, leading to higher transmission delay and storage cost, which means that the ATC scheme can be difficult to perform (requires analysis before encoding) in many cases. The CTA scheme can overcome this problem by compressing the images for storage/transmission and later analysis is performed, but the decoded images are lossy reconstructions, which can lead to lower performance.

However, when deep learning image codecs are used, face verification can be performed over the latent representation of the image obtained after entropy decoding the bitstream, which can also be used for decoding the image, as presented in the JPEG AI paper [1] which is shown in Figure 25. This means that the facial features extracted while coding the image (with some quantization loss) will be used for face verification instead of the lossy decoded facial images. Therefore, this model is expected to achieve better performance than the decoded anchors at corresponding compression rates. The MSc Thesis main objective is the design, implementation and assessment of a novel compressed domain face verification model, which will use the latent representation of the image after entropy decoding to perform face verification.

The implemented model uses learning-based image codecs to encode images and to perform entropy decoding to obtain the latent representations. The latent representations are then used for face verification, by using a deep neural network to generate the face embeddings. The usage of different codecs, optimization losses (MSE and MS-SSIM) and qualities lead to multiple compressed domain solutions. However, due to the high amount of time needed to train these models, there was the need to only select one codec (Balle_2018 [82]), one optimization loss (MSE) and four target qualities (1, 3, 6, 8).

6.2. Compressed Domain Face Verification Architecture

In the compressed domain face verification model, the faces in the original images are initially cropped, aligned and resized. This was performed to evaluate the compressed domain face verification solution (the Thesis objective) without any influence of the face detection and processing step. Afterwards, the aligned faces are encoded, obtaining the respective bitstreams after entropy encoding, which are typically used for transmission/storage. In the decoder side, the latent representations are generated by entropy decoding the bitstreams. These probe latent representations are then used to perform face verification using the latent representations of original identity reference images.

In the decoded anchor solutions, in Section 5.3, the decoded probe images are compared with original reference images, by using the ResNet50 model trained with original images, which means that the performance of the decoded anchors will depend on the codec efficiency of reconstructing the faces. However, the compressed domain face verification model is trained with latent representations and thus the non-linear hyperspace in which the facial features are being represented, by the face embeddings, in the ResNet50 model and in the compressed domain NN model may not be the same. Therefore, it was selected to perform face verification by using the latents representations in both probe input and reference input, i.e., the probe and reference images are encoded and entropy decoded to obtain the probe latent representation and reference latent representation.

The compressed domain face verification model is mainly composed by a face detection model, a learning-based image codec, latent representation processing, face recognition model (compressed domain) and a similarity score and decision making process, as shown in Figure 42.

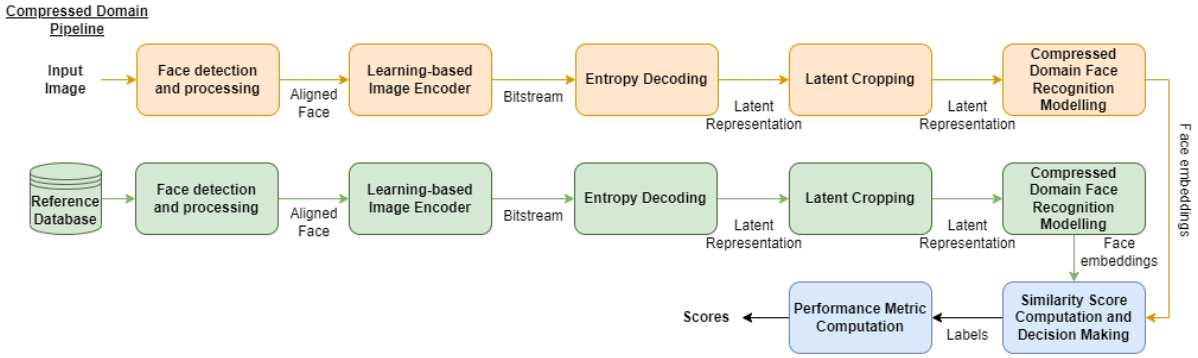

Figure 42 – *Compressed domain face verification solution.*

The compressed domain face verification architecture is composed by the following modules:

- **Reference Database –** database that contains all the original images of the faces used as references. These reference images are also encoded and entropy decoded to generate the reference latents representations.
- **Face Detection and Processing –** it was applied the same face detection model used in the original anchor described in Section 5.4 to obtain the face coordinates and landmarks. However, the face crop and alignment are performed before the encoding, which is different from the decoded anchors in Section 5.3. in which the face is cropped and aligned after reconstructing the images. It was chosen this approach since the cropping and alignment of the face in the latent representation is not the focus of this MSc Thesis and can be addressed in a future study. The processing of the faces after cropping is similar to the process in the original anchor, the faces are aligned, resized and padded to a final spatial size of 224x224 as described in Section 5.4 Since this processing occurs before the encoder, the faces are padded (with black) to a spatial size of 256x256 and the image values are normalized to values between 0 and 1 (instead of RGB vector subtraction).
- **Learning-based Image Encoder –** neural network encoder generates the bitstream for a desired reconstruction quality, same as in Section 5.3 For the compressed domain face verification model, it was selected the Balle_2018 codec optimized for MSE loss for qualities 1, 3, 6 and 8.
- **Entropy Decoding –** entropy encoding is a lossless process that receives as input the two latent representations (hyper-prior and the result of the synthesis transform), exploits their statistical

dependencies and serialize all information into a bitstream. At the decoder, the latent representations are restored, without artifacts, by entropy decoding. The obtained latent representations present a size equal to Nx16x16, being N the number of channels.

- **Latent cropping –** the face with spatial size 224x224 is padded to a size of 256x256 before encoding. However, this information it is not desirable for face verification, since it is similar to background information and thus can lower performance. Additionally, the compressed domain face recognition model accepts as input, latent representations with spatial size of 14x14. This means that there is the need to crop this extra information. The image was padded from a spatial dimension of 224x224 to 256x256, which corresponds to a pad of 16 pixels in all spatial directions. Also, the entropy decoder output (latent representations) has a spatial size of 16x16, which means that the image size was reduced 16x (256/16=16). Therefore, most of the padded region on the latent representation is represented by 1 value close to all borders, as shown in Figure 43 in red. This latent processing corresponds to the removal of this border by center cropping the latent representations with dimension 14x14. However, the spatial reduction performed by the encoder leads to portions of padding information being represented in spatial positions outside the cropped area. Therefore, the cropped latent representation still contains some padding information, although most of this information was removed during the latent cropping.

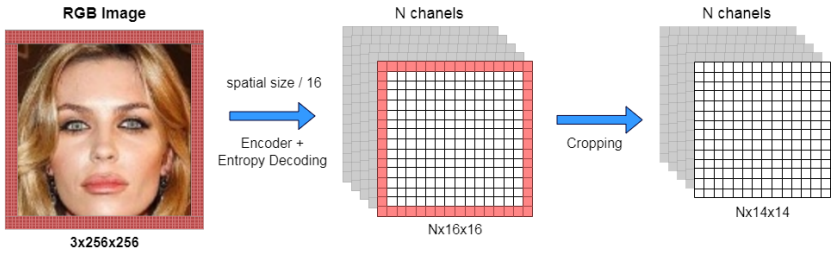

Figure 43 - *Illustration of the latent representation pre-processing for training the ResNet39. Face image obtained from the VGGFace2 dataset [119].*

- **Compressed Domain Face Recognition Modelling –** neural network model which takes a latent representation of a face (compressed) and generates a face embedding that can be used in face verification tasks. This model is described in Section 6.3.
- **Similarity Score Computation and Decision Making –** this module is the same as the one described for the original and decoded anchors (see Section 5.7) and thus, computes a binary label (True/False) for each face verification pair. In the compressed domain model described here, the probe and reference face embeddings are obtained from latent representations.
- **Performance Metric Computation –** this module is the same as the one used in the original and decoded anchors (Section 5.8) to obtain the quantitative performance results of the model.

6.3. Compressed Domain cResNet39

As shown in Section 5.5, deep learning face verification is a binary classification problem based on feature extraction. The feature extraction is performed by mapping the face to a lower-dimensional space, generating the face descriptors. Currently, many models have an extra FC layer as output layer to generate non-linear face embeddings by using the face descriptors. However, the learning-based encoder creates a latent representation with size Nx14x14 after entropy decoding and latent processing.

This means that the first blocks of the ResNet50, which perform convolutions and spatial downsampling, may be removed, since these operations are also performed by the encoder. The compressed domain deep-learning architecture is obtained by removing these blocks from the ResNet50, obtaining the cResNet39 (c corresponds to compressed), which has lower complexity.

6.3.1. Architecture

The cResNet39 architecture was implemented based in the ResNet50 implementation [118], presented in Section 5.5, and was inspired by the architecture modifications described in [90]. To obtain the cResNet39 architecture the blocks with higher spatial size than 28x28 of the ResNet50 are removed, i.e, the root block and the residual block 1 are removed. These modifications remove a total of 11 layers (10 convolutional layers + 1 max pool layer) from the ResNet50, as shown in Table 6.

The compressed domain face verification model takes latent representations of shape Nx14x14 as input, being N the number of channels. For this solution it was chosen the Balle_2018 codec pre-trained solutions optimized for MSE loss for target qualities 1, 3, 6 and 8 [122], [123]. The number of channels of the latent representation varies between the codec solutions accordingly to the used quality as follows:

- **Balle_2018 codec, MSE optimized, qualities 1 and 3 -** Number of channels (N) = 192
- **Balle_2018 codec, MSE optimized, qualities 6 and 8 -** Number of channels (N) = 320

There is the need to adjust the model's initial layers to support the latent representations as input. Since these representations have a spatial size of 14x14, then the cResNet39's residual blocks 2 and 3 will not perform downsampling, such that the output of residual block 3 has a spatial size of 14x14 as in the ResNet50, shown in Table 6. Hence, only the residual block 4 will apply downsampling in the cResNet39 model. Additionally, the first convolutional layer of residual block 2 is adjusted to receive inputs with 320 or 192 channels (depending on the used compression quality), instead of 256 channels like in the ResNet50. For face identification (only used for training), a 8631-d FC layer is used to generate the output for multi-classification problems, based in [116][118]. For face verification, the 8631-d FC layer is replaced by a 2048-d FC layer that will generate the face embeddings, based in [116] and in [117]. More details of the compressed domain face verification cResNet39 architecture are presented in Table 6 and Figure 44.

Table 6 – *Structure of the ResNet50 and cResNet39 architectures. Each residual block is composed of residual units that contain 3 convolutional layers each, e.g., residual block 1 has 3 residual units. N is the number of channels of the latent representation, which varies between codec solutions.*

		Input Size	Block	Root	Residual block 1	Residual block 2	Residual block 3	Residual block 4	
RGB	ResNet50	(3,224,224)	Output Size (spatial)	56x56	56x56	28x28	14x14	7x7	avg. pool, 2048-dim fc [output: 1x2048]
			Arch.	[7x7, 64] 3x3 max pool, s=2	$\begin{bmatrix} 1x1, 64 \\ 3x3, 64 \\ 1x1, 256 \end{bmatrix}$ x3	$\begin{bmatrix} 1x1, 128 \\ 3x3, 128 \\ 1x1, 512 \end{bmatrix}$ x4	$\begin{bmatrix} 1x1, 256 \\ 3x3, 256 \\ 1x1, 1024 \end{bmatrix}$ x6	$\begin{bmatrix} 1x1, 512 \\ 3x3, 512 \\ 1x1, 2048 \end{bmatrix}$ x3	
Compressed Domain	cResNet39	(N,14,14)	Output Size (spatial)	None	None	14x14	14x14	7x7	
			Arch.			$\begin{bmatrix} 1x1, 128 \\ 3x3, 128 \\ 1x1, 512 \end{bmatrix}$ x4	$\begin{bmatrix} 1x1, 256 \\ 3x3, 256 \\ 1x1, 1024 \end{bmatrix}$ x6	$\begin{bmatrix} 1x1, 512 \\ 3x3, 512 \\ 1x1, 2048 \end{bmatrix}$ x3	

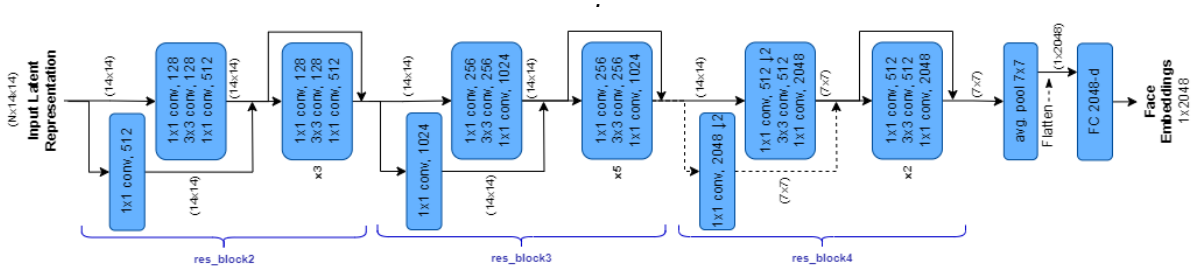

Figure 44 - *Framework illustration of the cResNet39 final architecture for face verification. N corresponds to the number of channels of the input latent representation, this value differs between the codec models. Dashed line represents the downsampling path of the residual block input which is added to the residual block's first residual unit output.*

6.3.2. Model Training

The cResNet39 training procedure is similar to the ResNet50 training in Section 5.5.2, and thus is composed by two phases: 1) training a model with an 8631-d output FC layer for face identification (cResNet39_8631-d); 2) replace the 8631-d FC layer for a randomly initialized 2048-d FC layer, which will generate the face embeddings for face verification. The cResNet39 with 2048-d face embeddings is then fine-tuned for face verification, only updating the FC layer.

The cResNet39 is trained with the VGGFace2 dataset [119] (~3.1M images, 8631 identities). As in Section 5.5.2, all the dataset is used in training, while for fine-tuning the model, the dataset is partitioned into training set (90%, 7768 identities, ~2.8M images) and validation set (10%, 863 identities, ~310k images). The validation loss is used by the learning-rate scheduler to reduce the learning-rate in the presence of loss plateaus, as in the ResNet50 model, however in practice no reduction was performed during training. The training images are preprocessed as in Section 5.5.2: 1) resize from 112x112 to 256x256; 2) random center crop to size 224x224; 3) grayscale conversion with 20% probability. Before the encoder, it is applied padding in the image to a spatial size of 256x256 and a normalization to values between 0 and 1. As described in Section 6.2, the latent representations obtained after entropy decoding have a spatial size of 16x16, which are then center cropped to a size of 14x14, as shown in Figure 43.

The training and fine-tuning processes are similar to the ones described in Section 5.5.2, operating in the following conditions:

- **cResNet39 Training –** the cResNet39 training for face identification is performed on the cResNet39_8631-d architecture with all images in the VGGFace2 (~3.1M images) and with the CEL function presented in (3) with mean reduction as in Section 5.5.2. The model is trained with the mini-batch SGD optimizer, with mini-batches of 64 images and a learning-rate of 0.1 for 6 epochs. The number of training epochs for the cResNet39 model is lower than for the ResNet50 model due to time limitation, since the models take a large amount of time to train. By analyzing the training loss of the ResNet50 train and of the cResNet39 train with the Balle_2018 codec (MSE optimized) with quality 8, it was noticeable that the loss has almost stabilized after epoch 6. Only the cResNet39 weights are updated, the encoder weights are kept constant.
- **cResNet39 Fine-tuning –** after training the cResNet39, the 8631-d output FC layer is replaced by the ResNet50's FC layer that generates 2048-d face embeddings, which was fine-tuned as described

in Section 5.5.2. Since the layer was fine-tuned with the ResNet50 using original images, there is the need for the weights to learn for face verification using the cResNet39 and latent representations as input. The VGGFace2 dataset is split into train set (90%) (~2.8M images) and validation set (10%) (~310k images). The cResNet39 model is fine-tuned with the ArcFace Loss [117][129], as described in Section 5.5.2. Only the 2048-d FC layer and the extra ArcFace FC layer are updated, the rest of the cResNet39 and encoder weights are kept constant. The model is fine-tuned with the mini-batch SGD optimizer, with mini-batches of 64 images and a learning-rate of 0.005 for 3 epochs. The number of epochs is also lower when comparing to the ResNet50 fine-tuning. After some analysis of the ResNet50 fine-tuning process, it was observed that the train loss starts to stabilize in epoch 3, and thus the number of fine-tuning epochs was reduced to 3 to save time.

The training and fine-tuning of the cResNet39 is performed with the Balle_2018 codec (MSE optimization), which uses the latent representation obtained after entropy decoding. In this case, it was selected 4 different models (1, 3, 6 and 8) for the assessment of the compressed domain model. These models cover a wide range of qualities and are described next:

- **cResNet39_Balle2018_mse_Q1 –** cResNet39 model solution with input of shape 192x14x14, trained with latent representations obtained by using the Balle_2018 codec optimized for MSE loss with quality 1. During training it was achieve a mean compression rate of 0.108 bpp over the VGGFace2 dataset.
- **cResNet39_Balle2018_mse_Q3 –** same as previous solution but trained for quality 3. The training dataset (VGGFace2) compression achieved a mean rate of 0.187 bpp.
- **cResNet39_Balle2018_mse_Q6 –** cResNet39 model with input of shape 320x14x14, trained with latent representations obtained by using the Balle_2018 codec optimized for MSE loss with quality 6. During training it was achieve a mean compression rate of 0.418 bpp over the VGGFace2 dataset.
- **cResNet39_Balle2018_mse_Q8 –** same as previous solution but trained for quality 8. The training dataset (VGGFace2) compression achieved a mean rate of 0.695 bpp.

6.4. Performance Evaluation

The performance of the compressed domain face verification model is evaluated with the same performance metrics presented in Section 5.8. Naturally, the compressed domain model (cResNet39) is compared to the original and decoded anchors solutions (ResNet50) under the similar conditions. Additionally, it was made a computational assessment of the anchors solutions and compressed domain solution.

In the compressed domain face verification model the faces are cropped, aligned and resized to a spatial size of 224x224 as in Section 5.4, but this is performed before encoding. Additionally, the image is padded (with black) to a size of 256x256, which corresponds to a padding of 16 pixels in all directions, and the resulting padded image is encoded. In the decoded anchors solutions presented in Section 5, the original probe image (250x250) is padded to a size of 256x256 before the encoder, corresponding to a padding of 3 pixels in all borders. The image is then encoded and decoded to obtain a lossy reconstruction, from which the face is cropped, aligned and resized to a size of 224x224, as in Section 5.4. Consequently, the decoded anchors solutions in Section 5 have different compressed representations (bitstreams) from the compressed domain model. Moreover, since the compressed

domain model only compresses resized faces with higher padding area, more area of the image with constant value 0 (black), a higher compression efficiency is expected compared to the decoded anchors. In order to have the similar conditions for all solutions, the decoded anchor model is adjusted such that the probe face cropping, alignment and resize is performed before the encoder. For a more complete performance evaluation, decoded anchor solutions based in conventional codecs are also implemented, i.e., the aligned and resized probe faces are encoded and decoded with conventional codecs and the lossy reconstructed face is then used as probe input for face verification. The selected conventional codecs are the JPEG [59][130], the JPEG2000 [63][131] and the HEVC Intra [70][132], which are presented in Section 3.1. For the decoded anchors, only the probe face is encoded and decoded, the reference identity input is an original image.

The performance evaluation results are obtained from the following anchors and for the novel compressed domain solutions:

- **ResNet50_Original_Anchor -** Original anchor solution described in Section 5.2
- **ResNet50_Balle2018_mse_Qn –** Decoded anchor solution described in Section 5.3 with the face cropping, alignment and resize before the encoder, as previously described, with the Balle_2018 codec [82] optimized for the MSE loss function and model n (1, 3, 6 or 8) that leads to different target qualities.
- **ResNet50_JPEG_Qn -** Decoded anchor solution described in Section 5.3 with the face cropping, alignment and resize before the encoder, with the JPEG codec [59] and quality n (1, 5, 25 or 62). The target rates for this codec are similar to the rates used in the Balle_2018 codec and were obtained by adjusting the compression factor.
- **ResNet50_JPEG2000_mse_Qn -** Decoded anchor solution described in Section 5.3 with the face cropping, alignment and resize before the encoder, with the JPEG2000 codec [63] optimized for MSE distortion and model n (1, 3, 6 or 8) for each target quality. The target qualities for this codec are similar to the qualities used in the Balle_2018 codec and were obtained by adjusting the compression factor.
- **ResNet50_JPEG2000_visual_Qn -** Same as previous decoded anchors solution but the codec used a perceptual optimized configuration.
- **ResNet50_HEVC_Qn -** Decoded anchor solution described in Section 5.3 with the face cropping, alignment and resize before the encoder, with the HEVC Intra codec [70] and quality n (41, 35, 26 or 19) for each target quality. Again, the target rates for this codec are similar to the compression rates used in the Balle_2018 codec and were obtained by adjusting the compression factor.
- **cResNet39_Balle2018_mse_Qn –** Novel compressed domain face verification solution described in Section 6.2 and Section 6.3, with the Balle_2018 codec [82] optimized for the MSE loss function and model n (1, 3, 6 or 8) that corresponds to different target qualities.

6.4.1. Dataset and Face Verification Protocol

The anchors solutions and novel compressed domain solution are evaluated for the LFW dataset [109][128], as in Section 5.9.1. The LFW test dataset has a total of 13233 images across 5749 identities. The performance evaluation is made in accordance with the LFW2 face verification protocol as described in Section 5.9.2 and in [109]. The LFW2 protocol has available 6k pairs (12k images), 3k

positive pairs (same identity) and 3k negative pairs (different identity), containing a probe image, an identity reference image and a ground-truth label in each sample. The decoded anchors only encode and decode the probe face, and, as in Section 5.9, the compression rate is calculated only using the probe faces, since the identity reference face is an original image. Consequently, the compression rate over the LFW dataset for the compressed domain model is also calculated only with the probe face of each pair.

The performance metrics and respective computations are the same as in Section 5.8 and Section 5.9.2. Hence, in the performance evaluation of the anchors solutions and novel compressed domain solutions it is used the TAR@FAR=0.001, TAR@FAR=0.01, ERR and AUC metrics.

Additionally, the computational complexities of the ResNet50_Original_Anchor, ResNet50_Balle2018_mse_Qn and cResNet39_Balle2018_mse_Qn solutions, for n in (1, 3, 6, 8), are computed, which will allow for a more detailed assessment of the presented solutions. The cResNet39 model have less layers than the ResNet50 and the latent representations have lower spatial size than the images. Moreover, in the decoded anchors solutions the latent representations are decoded to obtain the lossy reconstructed face, which increases the model's complexity. Hence, it is expected that the compressed domain solutions achieve lower computational complexity than the anchors solutions. The computational complexity is divided into two components: 1) number of Multiply-Accumulate (MAC) operations, which represents a proxy of the number of Floating-Point Operations (FLOPs), executed to obtain the face embedding of one input. It is also represented in number of MAC operations per pixel (MAC/px), considering an input image of 224x224; 2) mean execution/computation time of the face verification task on the LFW test dataset, using the LFW2 protocol, in milliseconds per batch (of size 64 pairs, i.e., 128 images/latent representations) and in milliseconds per pair (1 probe and 1 reference images/latent representations).

The MAC operation is the combination in one common step of the computation of the product between two numbers followed by an addition to an accumulator, as in (9).

$$\boldsymbol{a} \leftarrow \boldsymbol{a} + (\boldsymbol{b} \times \boldsymbol{c}) \tag{9}$$

The number of MAC operations is a very popular metric used in deep learning to compare models' computational complexities, as it represents the most fundamental operation in neural networks. It is also used in the common test conditions of the JPEG AI codec. Additionally, the MAC operations are composed of 2 FLOPs, meaning that the MAC value is a proxy of the number of FLOPs performed in the model. For the decoded anchors and compressed domain solutions the number of MAC operations are calculated after the entropy decoding since it is only relevant to obtain this information at the decoder side (encoder is always the same). For the original anchor solution, it is only considered the ResNet50 model when calculating the number of MAC operations. In all solutions it is not included the pre-processing of images or latent representations since the MAC value is only applicable for neural networks' operations. The evaluation of the models' complexity in number of MAC operations is performed based on the implemented method available in [130].

For the execution time, the pre-processing of images or latent representations is taken into consideration, since it is desirable to include the process of input preparation before entering the models. However, the face detection process described in Section 5.4, which is performed at the encoder side,

is not included in the solutions' execution times, since it is not the objective of this MSc Thesis to evaluate the face detection tasks. Although, the facial data (face boundaries coordinates and landmarks) obtained from the face detection model is used to crop the face from the image and perform face alignment. The execution times for each solution consider the following steps:

- **ResNet50_Original_Anchor –** face cropping, alignment and resize of both probe and face images followed by the face verification task.
- **ResNet50_Balle2018_mse_Qn –** for the probe image, it is considered the process after entropy decoding, i.e., the generation of the image lossy reconstruction by fully decoding the latent representation, followed by the face verification task. For the identity reference image, it is considered the face cropping, alignment and resize, since it is an original image, and the face verification task. Hence, this solution testing conditions differ from the compressed domain model, since only the probe image is encoded and decoded.
- **cResNet39_Balle2018_mse_Qn –** cropping of both probe and reference latent representations followed by the face verification task.

The test process on the LFW dataset is performed 50x for each solution in order to obtain the mean execution time values, with batches of 64 pairs. The test is performed with the 64-bits NVIDIA GeForce RTX 3080 GPU [133], with 12 GB of memory (memory type: GDDR6X), clock frequency of 33 MHz (clock boost 1.71 GHz) and 8960 CUDA cores.

6.4.2. Experimental Results

The face verification performance metrics results for the anchors and the novel compressed domain solution is given by rate-metric plots which are shown in Figure 45. Additionally, in Figure 46 it is presented the MS-SSIM and PSNR reconstruction performance results for the selected codec solutions on the LFW dataset, by using the probe images of LFW2 protocol. The MS-SSIM and PSNR are obtained for two different processing methods: 1) face cropping, alignment and resize before encoding, as in the decoded anchors described in Section 6.4; 2) face cropping, alignment and resize after decoding, as in Section 5.3. For the face processing methods of item 2) it is only computed the compression metrics results for the Balle_2018 codec with MSE optimization codec. Finally, the computational complexities of the ResNet50_Original_Anchor, ResNet50_Balle2018_mse_Qn and cResNet39_Balle2018_mse_Qn solutions, for n in (1, 3, 6, 8), are presented in Table 7. Additionally, in Table 7 is also presented the TAR@FAR=0.001 face verification performance metric, since it is the most discriminative metric between the selected face verification metrics.

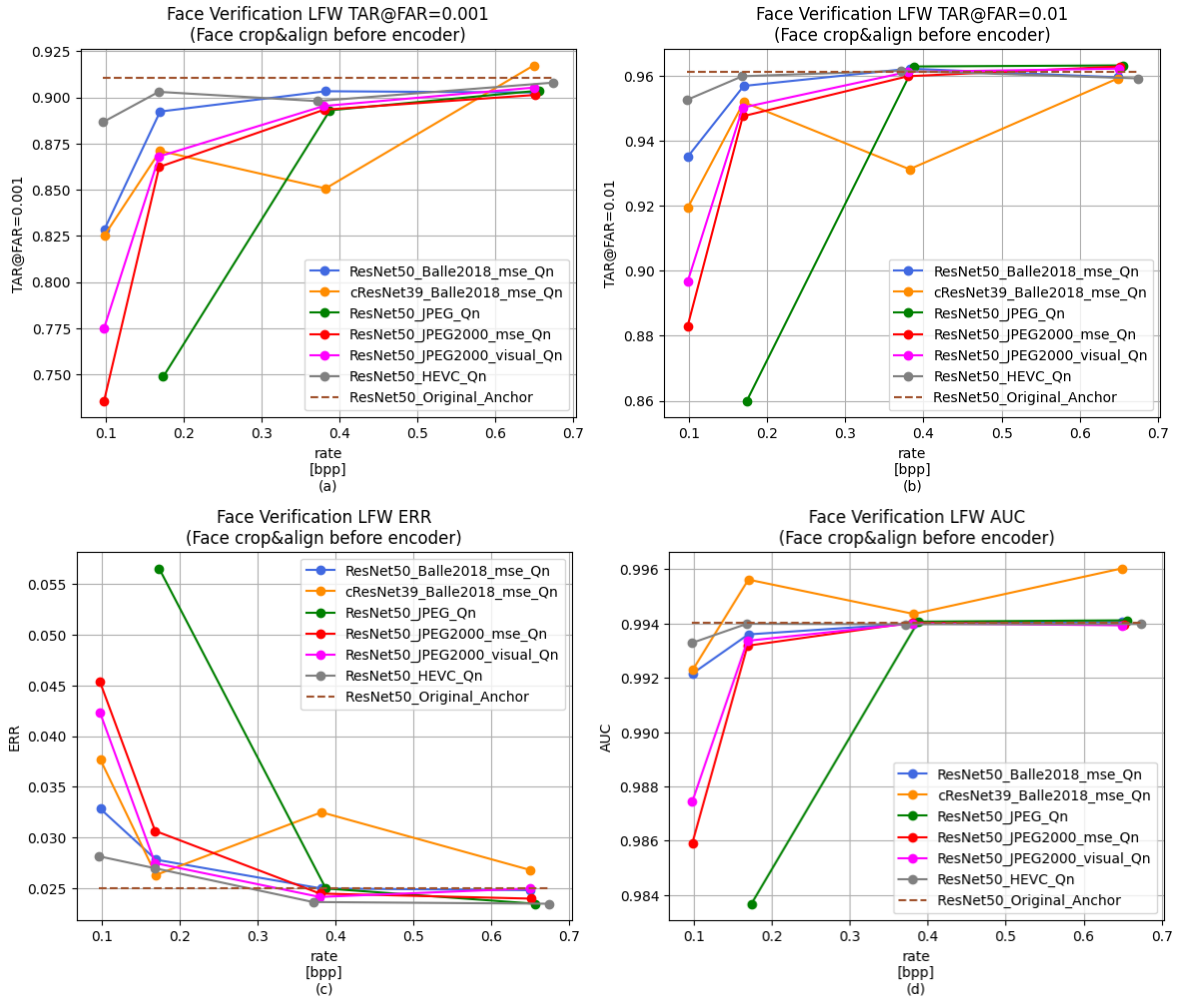

Figure 45 – *Face verification results for all anchors (ResNet50) and novel compressed domain solutions (cResNet39) for the different mean rate for the test images of the LFW dataset with the LFW2 protocol (6k pairs). (a) TAR@FAR=10^{-3}; (b) TAR@FAR=10^{-2}; (c) ERR; (d) AUC.*

Table 7 – *Computational complexity and face verification performance results on the LFW dataset (LFW2 protocol) of the ResNet50_Original_Anchor, ResNet50_Balle2018_mse_Qn and cResNet39_Balle2018_mse_Qn solutions, for n in (1, 3, 6, 8). Computational complexity is measured with number of operations in GMAC and in kMAC/px (per pixel) and with computational time in s/batch and in s/pair. Each batch contains a total of 64 pairs which corresponds to 128 images. The face verification performance is measured with the TAR@FAR=10^{-3}. (*) the solutions' computational times are mean values obtained by repeating the test process 50x with the NVIDIA RTX 3080 GPU [133].*

	Computational Complexity				**Face Verification** (LFW)	
	Operations		**Time***			
Models	Total [GMAC]	Per pixel [kMAC/px]	Batches [ms/batch]	Pairs [ms/pair]	TAR@ FAR=10^{-3}	Rate [bpp]
ResNet50_Original_Anchor	3.89	78	205.3	3.216	0.910	
ResNet50_Balle2018_mse_Q1	13.55	270	215.0	3.368	0.829	0.10
ResNet50_Balle2018_mse_Q3	13.55	270	215.1	3.370	0.892	0.17
ResNet50_Balle2018_mse_Q6	25.3	504	240.0	3.760	0.903	0.38
ResNet50_Balle2018_mse_Q8	25.3	504	240.8	3.772	0.903	0.65
cResNet39_Balle2018_mse_Q1	2.36	47	42.7	0.669	0.825	0.10
cResNet39_Balle2018_mse_Q3	2.36	47	42.5	0.666	0.871	0.17
cResNet39_Balle2018_mse_Q6	2.38	47	42.9	0.672	0.851	0.38
cResNet39_Balle2018_mse_Q8	2.38	47	42.9	0.673	0.917	0.65

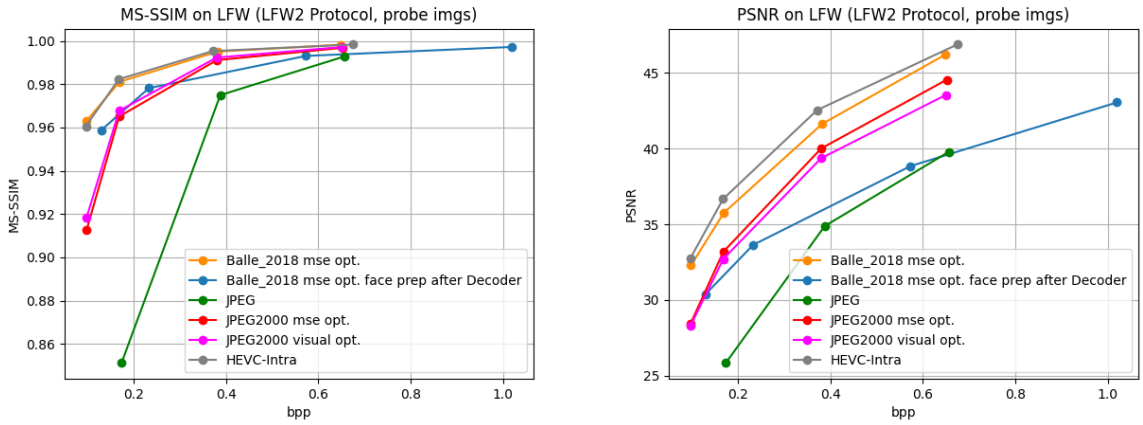

Figure 46 – *Image reconstruction testing results on the LFW dataset, using the images included in the LFW2 protocol probe images, for the codecs JPEG (Green), JPEG2000 with MSE optimization (Red), JPEG2000 with visual optimization (Purple), HEVC-Intra (Grey), Balle_2018 with MSE optimization (Orange), Balle_2018 with MSE optimization with target qualities 1, 3, 6 and 8 with face crop, align and resize after the decoder, as in Section 5.3 (Blue): (left) MS-SSIM; (right) PSNR. All solutions perform face crop, align and resize before the encoder, as in Section 6.4, with the exception of the solution with the blue line.*

From Figure 45 it is also noticeable the increase in performance with the increase of the bitrate, however some unexpected deviations were observed. Some other conclusions can be drawn from the obtained results:

- **Performance results for TAR@FAR=0.01, ERR and AUC –** the TAR@FAR=0.01, ERR and AUC metrics do not have a major increase in the performance while increasing the bitrate, similar to the results obtained in Section 5.9.3; the values of the performance metrics are contained in the intervals [0.86, 0.965], [0.06, 0.02] and [0.983, 0.996] respectively. However, the AUC values are all close to the ideal value 1, which shows high performance results for all solutions in this metric. Likewise, the ERR and TAR@FAR=0.01 also show high performance results. This means that these metrics are not very discriminative, especially compared to TAR@FAR=0.001.
- **cResNet39_Balle2018_mse_Qn vs ResNet50_Balle2018_mse_Qn –** the decoded anchors solutions using the Balle_2018 codec with MSE optimization (ResNet50_Balle2018_mse_Qn) achieve higher performance than the compressed domain model (cResNet39_Balle2018_mse_Qn) in almost all metrics and bitrates. Although the ResNet50_Balle2018_mse_Qn model performance results are better than the cResNet39_Balle2018_mse_Qn, the difference in performance can be considered small, since in all of the selected metrics the results do not vary more than a value of 0.06. These performance results are unexpected, since it was foreseen that the compressed domain model would achieve higher performance than the decoded anchor solutions when using the same codec. However, in this assessment, it was used a model with lower complexity (cResNet39), as shown in Table 7, for the compressed domain model than for the decoded and original anchors solutions (ResNet50). Additionally, the cResNet39 model was trained for fewer epochs (9 epochs total: 6 train epochs + 3 finetune epochs) than the ResNet50 model (15 epochs total: 10 train epochs + 5 finetune epochs), due to time limitation. At last, the decoded anchors ResNet50 model is trained with original images of spatial size 224x224, in which downsampling is applied in residual blocks 2, 3 and 4, obtaining the size 7x7 after the last convolutional layer. Hence, the ResNet50 performs a

spatial downsampling of the image of x32 (224/32=7), meaning that when training the ResNet50, the network learns this compact representation which can be beneficial for face verification tasks. Consequently, the decoded anchors solutions performance results only depend on the codec's efficiency in the reconstruction of the original face. However, the cResNet39 is trained with latent representations with spatial size 14x14, only performing downsampling in the residual block 4 as described in Section 6.3.1. After the cResNet39's last convolutional layer the output has a size of 7x7, which means a spatial reduction of x2. The downsampling process in the compressed domain solution is mainly carried out by the encoder, which is not trained alongside the cResNet39. This means that the compressed domain model performance depends on the encoder efficiency for face verification, a task which was not trained for (but for lossy reconstruction). The fact that the encoder was not trained along with the cResNet39, that the model is more simple and trained for fewer epochs explains the gap in performance in some bitrates. The highest gap in performance between these solutions is in the TAR@FAR=0.001 metric, in which the ResNet50_Balle2018_mse_Qn and cResNet39_Balle2018_mse_Qn solutions have 0.903 and 0.851 respectively for the target bitrate 0.38 bpp. Additionally, the cResNet39_Balle2018_mse_Q6 solution has significantly lower performance results in all metrics in its corresponding target bitrate, except for the AUC metric. However, even the AUC result of the cResNet39_Balle2018_mse_Q6 solution shows outlier behavior, i.e., the AUC metric plot decreases for the cResNet39_Balle2018_mse_Qn in the bitrate 0.38 bpp when increasing the target bitrate. This means that the weights random initialization of the cResNet39_Balle2018_mse_Q6 model may have influenced the performance of the model, since the training process is the same for all compressed domain models However, for the highest target bitrate (0.649 bpp) the compressed domain model presents higher performance in the TAR@FAR=0.001 than all anchors, including the original anchor.

- **Computational complexity –** from the computational complexity results presented in Table 7 it can be observed that all compressed domain solutions have similar computational complexity (47 kMAC/px and ~0.67 ms/pair). The original anchor solution has higher complexity than the compressed domain solutions, with the values 78 kMAC/px and 3.216 ms/pair. This corresponds to an increase in computational complexity of 66% (x1.66) in the number of MAC operations per pixel and of 380% (x4.80) in the execution time. The learning-based decoded anchor solutions (ResNet50_Balle2018_mse_Qn) is divided into two types in terms of computational complexity:
  - **n = 1 or 3 –** number of MAC operations equal to 270 kMAC/px and execution time of 3.370 ms/pair, hence there is an increase in computational complexity of these solutions of 474% (x5.74) and 402% (x5.02) respectively, when comparing with the compressed domain solutions.
  - **n = 6 or 8 –** 504 kMAC/px and execution time of 3.770 ms/pair. This means that there is an increase in the number of MAC operation per pixel of 972% (x10.72) and an increase in the execution time of 463% (x5.63) in relation with the cResNet39_Balle2018_mse_Qn solutions.

  The original anchor has lower computational complexity than the decoded anchors, as expected, since in the decoded anchors solutions the latent representation are used to perform reconstruction before performing the face verification task. In the original anchor solution, the mean execution time includes the face cropping, alignment and resizing processes for both images in a pair. However,

this image processing is not included in the execution times of the compressed domain model. In addition, in the decoded anchor, the execution time only considers the image processing for the original reference image. By analyzing the image processing computational times, it was possible to observe that the execution of this process (face cropping, alignment and resizing) for one image in the pair adds up an execution time of 0.28 ms/pair and thus is rather small. This leads to new execution times of the decoded anchors (3.65 ms/pair for n=1, 3; 4.05 ms/pair for n=6, 8) and compressed domain (1.24 ms/pair) solutions. Consequently, the original anchor and decoded anchor solutions present an increase in execution time of 159% (x2.59) and of 227% (x3.27) respectively, with respect to the compressed domain solutions, when considering the image processing step. Finally, the novel compressed domain face verification solutions present significantly lower complexity than the anchors solutions while having similar face verification performance, with the exclusion of the cResNet39_Balle2018_mse_Q6 model (considered as an outlier due to the random initialization). Hence, although the compressed domain solutions have lower performance than the ResNet50_Original_Anchor and ResNet50_Balle2018_mse_Qn solutions, it achieves lower computational complexity, leading to a better performance-complexity operating point.

- **Decoded anchors (Conventional vs Learning-based) –** at higher bitrates, the conventional decoded anchors solutions (ResNet50_JPEG, ResNet50_HEVC, ResNet50_JPEG2000_mse, ResNet50_JPEG2000_visual) have similar results to the learning-based decoded anchor solutions (ResNet50_Balle2018_mse_Qn). However, at lower bitrates, the ResNet50_Balle2018_mse_Qn has higher results than all conventional decoded anchors solutions in all performance metrics, apart from the ResNet50_HEVC solution. The HEVC Intra decoded anchor solution (ResNet50_HEVC) presents the highest performance results at lower bitrates compared to all other solutions. These results were expected since the HEVC Intra is a very efficient codec, especially when compared to JPEG, JPEG2000 and Balle2018. Beside the HEVC Intra, the Balle_2018 optimized for MSE loss codec can better reconstruct the facial images for face verification tasks than the used conventional codecs. From Figure 46, which presents the image reconstruction performance of all relevant codecs, it can be observed that image reconstruction performance results are related to the face verification TAR@FAR=0.001 metric results. The codecs with higher reconstruction qualities for the same bitrate (HEVC-Intra and Balle_2018 with MSE optimization) are the ones which perform higher as decoded anchors for the face verification task. Hence, the decoded anchors solutions face verification performance mainly depends on the codec solutions efficiency in reconstruct the facial features.
- **Compressed domain model vs conventional decoded anchors –** for the lowest target bitrate 0.10 bpp, the compressed domain solution cResNet39_Balle2018_mse_Q1 has significantly higher performance results in all metrics than the conventional decoded anchors ResNet50_JPEG, ResNet50_JPEG2000_mse and ResNet50_JPEG2000_visual. However, the performance results of these solutions become similar with the increase of the bitrate, excluding the ResNet50_JPEG for target bitrate 0.17 bpp, which has the lowest performance for this rate, and the cResNet39_Balle2018_mse_Q6 (outlier). The ResNet50_HEVC has the highest performance results at lower bitrates, as observed before. As expected, the performance of the compressed domain

models is higher than the conventional decoded anchors, excluding the HEVC, for lower rates and similar performance for higher rates. However, it was expected the gap in performance between the compressed domain solution and the conventional decoded anchors to be larger. This can be explained as in the “cResNet39_Balle2018_mse_Qn vs ResNet50_Balle2018_mse_Qn” conclusion, i.e., cResNet39 has lower complexity, trained for less epochs than the ResNet50 and the encoder not trained alongside the cResNet39 model.

- **Face crop, align and resize (before encoder vs after decoder) –** in Figure 46, it is represented the reconstruction performance (MS-SSIM and PSNR) of the Balle_2018 codec optimized for MSE loss with target qualities 1, 3, 6 and 8 with two different image pre-processing for the LFW dataset (LFW2 protocol): the Balle_2018 codec with MSE optimization with face processing (crop, align and resize) before the encoder in orange; the Balle_2018 with MSE optimization with face processing after the decoder in blue. It is possible to observe that performing face cropping, alignment and resizing before the encoder leads to higher compression and reconstruction efficiency, especially for higher target qualities. This means that the codec is able to decode images with higher reconstruction quality for lower bitrates, which explains the increase of the decoded anchor solutions’ performance of this section when compared with the decoded anchor solutions of Section 5.3.

7. Final Remarks

In this chapter it will be summarized the content of this MSc Thesis and the conclusions obtained from this work and some suggestions for future study.

7.1. Summary and Conclusions

Image coding is nowadays a common step in the storage and transmission of images through a variety of communication networks. Although conventional image codecs have improved rate-distortion performance over the years, it has been recognized that there are many future challenges that need to be addressed [58]. As shown in Section 3.3, DL image coding solutions are nowadays very popular and have been able to compete with state-of-art conventional image codecs. Moreover, the explosion in image data led also to the need of performing machine vision tasks with lower complexity. In deep-learning image codecs, image processing and computer vision tasks can be performed over the latent representation. This way, the extracted features obtained while coding the original images can be used instead of using the features computed from the lossy decoded image. Thus, adapting image processing or computer vision solutions to the compressed domain seemed to be a very important step to reduce complexity and storage cost in many applications.

In Section 1 a brief context and motivation for this report is presented, as well as the objectives and structure of this MSc Thesis. The defined main objective for this MSC Thesis was the design, implementation and assessment of a novel compressed domain solution that performs face verification over the latent representations.

In Section 2, the basic concepts and tools of deep-learning image analysis are reviewed. It is described basic concepts and architectures of neural networks and deep-learning images analysis tasks and respective state-of-art models.

In Section 3, it is performed a brief review of the basic concepts of conventional image codecs. Also, relevant learning-based image codecs are reviewed and evaluated against conventional image codecs.

In Section 4, two relevant compressed domain image analysis solutions are presented, which perform image classification and image segmentation. Moreover, it is reviewed a solution which performs facial feature extraction (face descriptors) at the encoder side, to which is employed quantization and entropy encoding (base information) for transmission/storage. The face descriptors, before quantization, are used to reconstruct the face to obtain the residual information between the original image and the reconstructed face. This residual information is then encoded for transmission/storage alongside with the base information. At the decoder side, the reconstructed facial features are used for face recognition tasks and to reconstruct the facial image. Finally, the reconstructed residual information is added to the reconstructed facial image to obtain a better reconstruction of the original image. It was described the solutions' objectives, designs, implementations and the corresponding performance assessment.

In Section 5, the two face verification anchors solutions were defined: original anchor and decoded anchor. This section describes the design, implementation and assessment of these solutions, which were used as baseline for the proposed compressed domain solution. Therefore, the used face

verification test protocol and dataset was described, as well as the selected performance metrics. In addition to the face verification model, it was also presented the selected learning-based codecs and face detection model, which were not evaluated in compression performance and detection performance respectively.

In Section 6, the proposed compressed domain face verification solution architecture and implementation are described, alongside with the training and testing processes. The decoded anchors solutions' were adjusted to perform face cropping, alignment and resizing before the encoder, to replicate the compressed domain model. Additionally, new decoded anchors solutions were defined based in popular conventional codecs. The codec (more precisely the encoder) described in [82] (Balle_2018) trained with MSE loss optimization for four target qualities was selected to derive the compressed domain solutions. Due to the high amount of training time required for each model and due to time limitation, no other codecs were selected. The face verification performance and computational complexity evaluation of the proposed solution in comparison with the anchors solutions was also presented.

From the results presented in Section 6, it can be concluded that the objective of improving face verification performance with compressed domain models, when comparing to the decoded anchors solutions, was not totally fulfilled. When considering only the decoded anchors based in conventional codecs, the proposed compressed domain solution achieved higher performance results for lower bitrates, excluding the solution which uses the HEVC Intra codec; for higher bitrates this was not verified. However, all solutions (anchors and proposed solutions) shows similar performance results for higher compression rates, which was expected since the facial images reconstruction quality increases with the target bitrate; except for the bitrate 0.38 bpp, in which the compressed domain model showed a significantly lower performance. The proposed solution performance for the rate 0.38 bpp can be partially explained by the model's weights random initialization, which may have led the model to a local minimum while training. Additionally, for lower bitrates the compressed domain model achieved lower performance than the corresponding decoded anchor (using same codec). This situation was not desirable, since it was expected an improvement in performance. However, it is understandable that the proposed solution was not able to outperform the corresponding decoded anchor, since the compressed domain model (cResNet39) is more simple and was trained for less epochs than the ResNet50 model used in the anchors solutions. In addition, the encoder was not trained alongside with the cResNet39, which means that the latent representations that are created are not optimized for face recognition, while in the ResNet50 all layers were trained for face verification. Although, the proposed compressed domain solutions shows significantly lower computational complexity than the anchors solutions, while having a small difference in performance.

In conclusion, the decoded solutions are a better option when it is desirable to have a high face verification performance, since it was shown that they can better verify a person's identity. However, for high bitrates this statement is not entirely true since all solutions achieved similar performance. Although, if the most relevant application concern is memory space and/or computational power (computational complexity) then the proposed compressed domain solution is the best candidate from the evaluated solutions. Nevertheless, a solution such as the one proposed in this MSc Thesis may be

adopted (more convolutional layer or training the encoder) to overcome the decoded solutions face verification performance.

7.2. Future Work

Given the relatively new concept of utilizing compressed representations of images for face recognition, it was inevitable that some challenges would appear. Considering the existing limitations, future efforts can be directed towards enhancing and conducting more robust performance assessments of solutions operating in the compressed domain as suggested:

- **Training with more epochs –** the proposed solution model was trained for less epochs than the anchors solutions, due to the high training time of the models. Training the proposed model for more epochs may improve performance.
- **Re-training the Network with the Encoder –** train the encoder weights alongside with the face verification network, aiming to obtain latent representations that maximize face verification. This has the disadvantage of the possibility of deteriorating the performance for standard pixel-wise reconstruction.
- **Add layers to Network –** the cResNet39 has significantly lower computational complexity than the ResNet50 used in the anchors solutions. As previously mentioned, the cResNet39 only performs a spatial reduction of x2, while the ResNet50 applies downsampling of x32. However, the ResNet50 can be modified to use compressed representations as input while maintaining the number of layers, obtaining the cResNet50 model (c means compressed). To achieve this goal, the max pool layer and convolutional layer from the root block of the ResNet50 need to be removed. To maintain the number of layers and the spatial size at the input of the residual block 1, a pixel shuffle upsampling layer with factor x4 can be added, which will increase the latent representations spatial size by x4 (to 56x56) and decrease the number of channels by x16. The upsampling layer can then be followed by a convolutional layer, which will adjust the number of channels. Additionally, the residual blocks of the cResNet50 can be initialized with the ResNet50 weights since the residual blocks architecture of these models is the same. These modifications will probably improve the face verification performance in the compressed domain since more layers are used.
- **Compressed Domain Face Crop and Resize –** the proposed compressed domain solution face processing (crop, align and resize) is performed before the encoder. This situation is not ideal, since in some image applications it is desirable to encode the entire image, usually for transmission/storage. This topic is the most challenging one but will enable to reconstruct the entire image at the decoder side, as in Section 5.3. However, the face cropping and resizing and eventually alignment processes need to be applied over the latent representation to enable to perform face verification at the decoder side.

In summary, the idea of performing face verification on compressed representations of the faces is still not efficient enough for high demanding face verification systems; however, it may be the best option for applications which require low computational power such as those employed on smartphones nowadays. Moreover, even thought that the proposed compressed domain face verification solution is not as efficient as the selected original and decoded anchors, it shows interesting results which may lead to future studies in compressed domain image analysis. Finally, the proposed concept allows to

perform face verification on latent representations without compromising the facial image reconstruction, since the encoder is not trained alongside the model.

References

[1] ISO/IEC JTC 1/SC 29/WG1 N90049, “White Paper on JPEG AI Scope and Framework v1.0,” 2021.

[2] “What Is Machine Learning?.” Learning Computer Science. https://www.learncomputerscienceonline.com/what-is-machine-learning/ (accessed Jan. 9, 2022).

[3] Pixel Decoder. (2021). Z. Yi. [Online]. Available: https://github.com/Geoyi/pixel-decoder.

[4] A. Amini (2020). Introduction to Deep Learning [PowerPoint slides]. Available: http://introtodeeplearning.com/2020/slides/6S191_MIT_DeepLearning_L1.pdf.

[5] Y. Gavrilova. “Convolutional Neural Networks for Beginners. Serokell. Available: https://serokell.io/blog/introduction-to-convolutional-neural-networks (accessed Aug. 20, 2021).

[6] W. M. Cowan, W. M. Cowan, T. C. Südhof and C. F. Stevens, *Synapses*, Baltimore, MD, USA: JHU Press, 2003.

[7] E. Akgün and M. Demir, “Modeling Course Achievements of Elementary Education Teacher Candidates with Artificial Neural Networks,” *Int. J. of Assessment Tools in Educ.,* vol. 5, no. 3, pp. 491-509, 2018.

[8] T. Dettmers. “Deep Learning in a Nutshell: Core Concepts.” Nvidia Developer. Available: https://developer.nvidia.com/blog/deep-learning-nutshell-core-concepts/ (accessed Sep. 2, 2021).

[9] L. Strika. “Convolutional Neural Networks: A Python Tutorial Using TensorFlow and Keras.” Kdnuggets. Available: https://www.kdnuggets.com/2019/07/convolutional-neural-networks-python-tutorial-tensorflow-keras.html (accessed Sep. 3, 2021).

[10] IBM Cloud Education. “Recurrent Neural Networks.” IBM.com. Available: https://www.ibm.com/cloud/learn/recurrent-neural-networks (accessed Sep. 3, 2021).

[11] IBM Cloud Education. “Unsupervised Learning.” IBM.com. Available: https://www.ibm.com/cloud/learn/unsupervised-learning (accessed Sep. 11, 2021).

[12] Y. LeCun and I. Misra. “Self-supervised learning: The dark matter of intelligence.” Meta AI. Available: https://ai.facebook.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/ (accessed Sep. 15, 2021).

[13] P. Radhakrishnan. “What are Hyperparameters ? and How to tune the Hyperparameters in a Deep Neural Network?.” Towards Data Science. Available: https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a (accessed Sep. 3, 2021).

[14] IBM Cloud Education. “Neural Networks.” IBM.com. Available: https://www.ibm.com/cloud/learn/neural-networks (accessed Sep. 3, 2021).

[15] P. Pandey. “Understanding the Mathematics behind Gradient Descent.” Towards Data Science. Available: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e (accessed Sep. 3, 2021).

[16] IBM Cloud Education. “Convolutional Neural Networks.” IBM.com. Available: https://www.ibm.com/cloud/learn/convolutional-neural-networks (accessed Sep. 11, 2021).

[17] S. Saha. “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way.” Towards Data Science. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (accessed Sep. 15, 2021).

[18] U. Udofia, “Basic Overview of Convolutional Neural Network (CNN),” *Medium Corp.*, 2018. [Online] https://medium.com/dataseries/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17 (accessed Feb. 03, 2022).

[19] “A Deep Learning and Grad-CAM based Color Visualization Approach for Fast Detection of COVID-19 Cases using Chest X-ray and CT-Scan Images.” ResearchGate. https://www.researchgate.net/figure/A-sample-of-the-convolution-operation-with-filter-size-3-and-stride-2_fig4_343508866 (accessed Sep. 17, 2021).

[20] “Classification of cerebral microbleeds based on fully-optimized convolutional neural network.” ResearchGate. https://www.researchgate.net/figure/A-toy-example-of-average-pooling-and-max-pooling_fig5_328783817 (accessed Sep. 18, 2021).

[21] W. Badr. “Auto-Encoder: What Is It? And What Is It Used For? (Part 1).” Towards Data Science. https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726 (accessed Sep. 18, 2021).

[22] M. Allahyari. “Autoencoders.” Sci2lab. https://sci2lab.github.io/ml_tutorial/autoencoder/ (accessed Sep. 18, 2021).

[23] “Deep semi-supervised learning for DTI prediction using large datasets and H2O-spark platform.” ResearchGate. https://www.researchgate.net/figure/Structure-of-Stacked-Autoencoders_fig2_325025951 (accessed Sep. 18, 2021).

[24] H. Saleem. “What is generative AI and how much power does it have.” IBM Developer Blog. Available: https://developer.ibm.com/blogs/what-is-generative-ai-and-how-much-power-does-it-have/ (accessed Sep. 19, 2021).

[25] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, “Generative Adversarial Nets,” *Advances in neural inf. Process. syst.,* vol. 27, Jun. 10, 2014.

[26] “The Generator.” Google Developers. Available: https://developers.google.com/machine-learning/gan/generator (accessed Sep. 18, 2021).

[27] N. Lei. “Generative Adversarial Network technology: AI goes mainstream.” IBM. Available: https://www.ibm.com/blogs/systems/generative-adversarial-network-technology-ai-goes-mainstream/ (accessed Sep. 18, 2021).

[28] “GAN Training.” Google Developers. Available: https://developers.google.com/machine-learning/gan/training (accessed Sep. 18, 2021).

[29] P. Gavali and J. S. Banu, “Chapter 6 – Deep Convolutional Neural Network for Image Classification on CUDA Platform,” in *Deep Learning and Parallel Computing Environment for Bioengineering Systems*, A. K. Sangaiah, Ed., New York, NY, USA: Academic Press, 2019, pp. 99-122.

[30] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng and M. Chen, “Medical image classification with convolutional neural network,” in *13th Int. Conf. on Control, Automat., Robot. & Vision*, Marina Bay Sands, Singapore, 2014, pp. 844-848, doi: 10.1109/ICARCV.2014.7064414.

[31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition”, *Proc. of the IEEE*, vol. 86, no.11, pp. 2278–2324, 1998, doi: 10.1109/5.726791.

[32] A. Krizhevsky, I. Sutskever and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Inf. Process. Syst. (NIPS)*, F. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger, 2012, vol. 25, doi: 10.1145/3065386.

[33] K. He, X Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image Recognition,” in *Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770-778.

[34] A. Verma. “Understanding Residual Networks (ResNets) Intuitively.” Towards Data Science. https://towardsdatascience.com/understanding-residual-networks-resnets-intuitively-5afef9f089ea (accessed Sep. 29, 2021).

[35] M. R. Khan. “Residual Networks (ResNets).” Towards Data Science. https://towardsdatascience.com/residual-networks-resnets-cb474c7c834a (accessed Sep. 29, 2021).

[36] M. R. Khan. “Intuition behind Residual Neural Networks.” Towards Data Science. https://towardsdatascience.com/intuition-behind-residual-neural-networks-fa5d2996b2c7 (accessed Sep. 29, 2021).

[37] F. Ramzan, M. U. G. Khan, A. Rehmat, S. Iqbal, T. Saba, A. Rehman and Z. Mehmood, “A Deep Learning Approach for Automated Diagnosis and Multi-Class Classification of Alzheimer’s Disease Stages Using Resting-State fMRI and Residual Neural Networks,” *J. Med. Syst.*, vol. 44, no. 37, pp. 1-16, Oct. 2019, doi: doi.org/10.1007/s10916-019-1475-2.

[38] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, X. Tang, “Residual Attention Network for Image Classification,” in *Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognit. (CVPR)*, Jul. 2017, pp.3156-3164, doi: 10.1109/CVPR.2017.683.

[39] Y. Amit, P. Felzenszwalb, R. Girshick, “Object Detection”, in *Computer Vision: A Reference Guide*, L. Zhang, Ed., Cham, Switzerland: Springer Int. Publishing, 2020, pp. 1-9, doi: 10.1007/978-3-030-03243-2_660-1.

[40] R. Girshick, J. Donahue, T. Darrell and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” in *Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognit. (CVPR)*, Jun. 2014, pp. 580-587.

[41] C. L. Zitnick and P. Dollár, “Edge Boxes: Locating Object Proposals from Edges,” *Compt. Vision-ECCV*, D. Fleet, T. Pajdla, B. Schiele and T. Tuytelaars, Eds., Cham, Switzerland: Springer Int. Publishing, 2014, pp. 391-405.

[42] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt and B. Scholkopf, “Support vector machines,” *IEEE Intell. Syst. And their Appl.*, vol. 13, no. 4, pp. 18-28, Jul.-Aug. 1998, doi: 10.1109/5254.708428.

[43] “Getting Started with R-CNN, Fast R-CNN, and Faster R-CNN”, 2021. https://www.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html#mw_25d18973-df6c-48ef-aaa9-31a4ec9e6705.

[44] R. Girshick, “Fast R-CNN,” in *Proc. of the IEEE Conf. on Comput. Vision (ICCV)*, Dec. 2015, pp. 1440-1448.

[45] S. Ren, K. He, R. Girshick and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137-1149, Jun. 2017, doi: 10.1109/tpami.2016.2577031

[46] “Anchor Boxes for Object Detection”, 2021. https://www.mathworks.com/help/vision/ug/anchor-boxes-for-object-detection.html.

[47] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou and G. Cottrell, “Understanding Convolution for Semantic Segmentation”, in *2018 IEEE Winter Conf. on Appl. of Comput. Vision (WACV)*, 2018, pp. 1451–1460. Doi: 10.1109/WACV.2018.00163.

[48] K. He, G. Gkioxari, P. Dollár and R. Girshick, “Mask R-CNN”, in *2017’ Proc. of the IEEE Conf. on Comput. Vision (ICCV)*, 2017, pp. 2980-2988. Doi: 10.1109/ICCV.2017.322.

[49] “An automatic nuclei segmentation method based on deep convolutional neural networks for histopathology images.” ResearchGate. https://www.researchgate.net/figure/The-overall-network-architecture-of-Mask-R-CNN_fig1_336615317 (accessed Feb. 05, 2022).

[50] D. Parthasarathy, “A Brief History of CNNs in Image Segmentation: From R-CNN to Mask R-CNN”, *Athelas*, Apr. 22, 2017. https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4 (accessed Oct. 13, 2021).

[51] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “DeepFace: Closing the Gap to Human-Level Performance in Face Verification”, in *Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognit. (CVPR),* Jun. 2014, pp. 1701–1708.

[52] F. Schroff, D. Kalenichenko and J. Philbin, “FaceNet: A Unified Embedding for Face Recognition and Clustering”, in *Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognit. (CVPR),* Jun. 2015, pp. 815-823.

[53] B. Amos, B. Ludwiczuk and M. Satyanarayanan, “OpenFace: A general-purpose face recognition library with mobile applications,” CMU-CS-16-118, Jun. 2016.

[54] V. Kazemi and J Sullivan, “One Millisecond Face Alignment with an Ensemble of Regression Trees”, in *Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognit. (CVPR),* Jun. 2014, pp. 1867-1874.

[55] H. Li, Z. Lin, X. Shen, J. Brandt and G. Hua, “A Convolutional Neural Network Cascade for Face Detection”, in *of the IEEE Conf. on Comput. Vision and Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5325-5334.

[56] A. Geitgey, “Machine learning is fun! Part 4: modern face recognition with deep learning,” *Medium Corp.*, 2016. [Online] https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78#.ds8i8oic9 (accessed Oct. 16, 2021).

[57] M. Antonini, M. Barlaud, P. Mathieu and I. Daubechies, “Image coding using wavelet transform,” *IEEE Trans. on Image Process.*, vol. 1, no. 2, pp. 205-220, Apr. 1992, doi: 10.1109/83.136597.

[58] S. Ma, X. Zhang, C. Jia, Z. Zhao, S. Wang and S. Wang, “Image and Video Compression With Neural Networks: A Review,” *IEEE Trans. Circuits and Syst. For Video Technol.*, vol. 30, no. 6, pp. 1683-1698, Jun. 2020, doi: 10.1109/TCSVT.2019.2910119.

[59] G. K. Wallace, “The JPEG still picture compression standard,” *IEEE Trans. Consum. Electronics*, vol. 38, no. 1, pp. 18-34, Feb. 1992, doi: 10.1109/30.125072.

[60] J. F. R. Herrera, “The JPEG standar (ISO/IEC 10918-1),” Jun. 24, 2014. https://w3.ual.es/~vruiz/Docencia/Apuntes/Coding/Image/03-JPEG/index.html (accessed Dec. 10, 2021).

[61] N. Ahmed, T. Natarajan and K. R. Rao, “Discrete Cosine Transform,” *IEEE Trans. Computers*, vol. C-23, no. 1, pp. 90-93, 1994, doi: 10.1109/T-C.1974.223784.

[62] D. A. Huffman, “A Method for the Construction of Minimum-Redundancy Codes,” *Proc. IRE*, vol. 40, no. 9, pp. 1098-1101, Sep. 1952, doi: 10.1109/JRPROC.1952.273898.

[63] C. Christopoulos, A. Skodras and T. Ebrahimi, “The JPEG2000 still image coding system: an overview,” *IEEE Trans. Consum. Electronics*, vol. 46, no. 4, pp. 1103-1127, Nov. 2000, doi: 10.1109/30.920468.

[64] “Basic architecture of JPEG2000 encoder and decoder.” ResearchGate. https://www.researchgate.net/figure/Basic-architecture-of-JPEG2000-encoder-and-decoder_fig3_328593488 (accessed Feb. 03, 2022).

[65] M. W. Marcellin, M. J. Gormish, A. Bilgin and M. P. Boliek, “An overview of JPEG-2000,” *Data Compression Conf.*, 2000, pp. 523-541, doi: 10.1109/DCC.2000.838192.

[66] D. Santa-Cruz, T. Ebrahimi, J. Askelof, M. Larsson and C. Christopoulos, “JPEG 2000 Still Image Coding Versus other Standards,” *SPIE 4115 : Appl. of Digit. Image Process. XXIII,* Dec. 2000, doi: 10.1117/12.411564.

[67] E. Aboufadel. “IV. How does JPEG 2000 work?.” JPEG 2000: The Next Compression Standard using wavelet technology. Dec. 2001. https://faculty.gvsu.edu/aboufade/web/wavelets/student_work/EF/how-works.html (accessed Dec. 10, 2021).

[68] M. Antonini, M. Barlaud, P. Mathieu and I. Daubechies, “Image coding using wavelet transform,” *IEEE Trans. Image Process.*, vol. 1, no. 2, pp. 205–220, Apr. 1992, doi: https://doi.org/10.1109/83.136597.

[69] D. Taubman, “High performance scalable image compression with EBCOT,” *IEEE Trans. Image Process*, vol. 9, no. 7, pp. 1158-1170, Jul. 2000, doi: 10.1109/83.847830.

[70] V. Sze, M. Budagavi and G. J. Sullivan, “High efficiency video coding (HEVC),” in *Integrated circuit and systems, algorithms and architectures*, A. P. Chandrakasan, Ed., Switzerland: Springer, 2014, vol. 39, ch. 1, pp. 1-4.

[71] F. Bossen, B. Bross, K Suhring and D. Flynn, “HEVC Complexity and Implementation Analysis,” *IEEE Trans. Circuits and Syst. For Video Technol.*, vol. 22, no. 12, pp.1685-1696, Dec. 2012, doi: 10.1109/TCSVT.2012.2221255.

[72] H. Schwarz, D. Marpe and T. Wiegand, “Overview of the Scalable Video Coding Extension of the H.264/AVC Standard,” *IEEE Trans. Circuits and Syst. For Video Technol.*, vol. 17, no. 9, pp.1103-1120, Sep. 2007, doi: 10.1109/TCSVT.2007.905532.

[73] G. J. Sullivan, J. R. Ohm, W. J. Han and T. Wiegand, “Overview of the High Efficiency Video Coding (HEVC) Standard,” *IEEE Trans. Circuits and Syst. For Video Technol.*, vol. 22, no. 12, pp.1649-1668, Dec. 2012, doi: 10.1109/TCSVT.2012.2221191.

[74] T. Nguyen and D. Marpe, “Performance analysis of HEVC-based intra coding for still image compression,” *Picture Coding Symp.*, 2012, pp. 233-236, doi: 10.1109/PCS.2012.6213335.

[75] T. Hinz, P. Helle, H. Lakshaman, M. Siekmann, J. Stegeman, H. Schwarz, D. Marpe and T. Wiegand, “An HEVC Extension for Spatial and Quality Scalable Video Coding,” *Proc. SPIE*, vol. 8666, pp. 866605- , Feb. 2013, doi: 10.1117/12.2009393.

[76] I. K. Kim, J. Min, T. Lee, W. J. Han and J. Park, “Block Partitioning Structure in the HEVC Standard,” *IEEE Trans. Circuits and Syst. For Video Technol.*, vol. 22, no. 12, pp.1697-1706, Dec. 2012, doi: 10.1109/TCSVT.2012.2223011.

[77] D. Marpe, H. Schwarz and T. Wiegand, “Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard,” *IEEE Trans. Circuits and Syst. for Video Technol.*, vol. 13, no. 7, pp. 620-636 , Jul. 2003, doi: 10.1109/TCSVT.2003.815173.

[78] B. Bross, Y. K. Wang, Y. Ye, S. Liu, J. Chen, G. J. Sullivan and J. R. Ohm, “Overview of the Versatile Video Coding (VVC) Standard and its Applications,” *IEEE Trans. Circuits and Syst. for Video Technol.*, vol. 31, no. 10, pp. 3736-3764 , Oct. 2021, doi: 10.1109/TCSVT.2021.3101953.

[79] E. Upenik, M. Testolina, J. Ascenso, F- Pereira and T. Ebrahimi, “Large-Scale Crowdsourcing Subjective Quality Evaluation of Learning-Based Image Coding,” in *IEEE Visual Commun. and Image Process. (VCIP 2021)*, Munich, Germany, Dec. 5-8, 2021.

[80] J. Ascenso, P. Akyazi, F. Pereira and T. Ebrahimi, “Learning-based image coding: early solutions reviewing and subjective quality evaluation,” *SPIE* in Optics, Photonics and Digital Technologies for Imaging Applications VI, vol. 11353, p. 113530S, Apr. 2020, doi: https://doi.org/10.1117/12.2555368.

[81] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor and M. Covell, “Full Resolution Image Compression With Recurrent Neural Networks,” *Proc. IEEE Conf. on Comput. Vision and Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017.

[82] J. Ballé, D. Minnen, S. Singh, S. J. Hwang and N. Johnston, “Variational image compression with a scale hyperprior,” *Int. Conf. Learn. Representations*, Vancouver, Canada, Apr. 2018.

[83] D. Minnen, J. Ballé and G. Toderici, “Joint Autoregressive and Hierarchical Priors for Learned Image Compression,” *32nd Conf. Neural Inf. Process. Syst.*, Montréal, Canada, Sep. 2018.

[84] J. Lee, S. Cho and S. K. Beack, “Context-adaptive Entropy Model for End-to-end Optimized Image Compression,” *Int. Conf. Learn. Representations*, New Orleans, LA, USA, May 2019.

[85] G. Ginesu, M. Pintus and D. D. Giusto, “Objective assessment of the WebP image coding algorithm,” *Signal Process.: Image Commun.*, vol. 27, no. 8, pp. 867-874, Feb. 2012, doi: doi.org/10.1016/j.image.2012.01.011.

[86] *Kodak Lossless True Color Image Suite (PhotoCD PCD0992)*, E. Kodak, Jan. 2013. [Online]. Available: http://r0k.us/graphics/kodak/.

[87] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth and L. J. Li, “YFCC100M: The New Data in Multimedia Research,” *Assoc. for Comput. Machinery: Commun. ACM*, vol. 59, no. 2, pp. 64-73, Feb. 2016, doi: 10.1145/2812802.

[88] A. Horé and D. Ziou, “Image Quality Metrics: PSNR vs. SSIM,” *20th Int. Conf. Pattern Recognit.*, 2010, pp. 2366-2369, doi: 10.1109/ICPR.2010.579.

[89] D. M. Rouse and S. S. Hemani, “Understanding and simplifying the structural similarity metric,” *15th IEEE Int. Conf. Image Process.*., 2008, pp. 1188-1191, doi: 10.1109/ICIP.2008.4711973.

[90] *Data compression in TensorFlow*. (2020). J. Ballé, S. J. Hwang, N. Johnston and D. Minnen. [Online]. Available: https://github.com/tensorflow/compression.

[91] R. Torfason, F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte and L. Van Gool, “Towards image understanding from deep compression without decoding,” *arXiv:1803.06131*, Mar. 2018.

[92] K. He, X Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image Recognition,” in Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognit. (CVPR), Jun. 2016, pp. 770-778.

[93] L. Theis, W. Shi, A. Cunningham and F. Huszár, “Lossy Image Compression with Compressive Autoencoders,” *arXiv:1703.00395v1*, Mar. 2017.

[94] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini and L. Van Gool, “Soft-to-hard vector quantization for end-to-end learned compression of images and neural networks,” CoRR, arXiv:1704.00648, vol. 3, Apr. 2017.

[95] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg and F. F. Li, “Imagenet large scale visual recognition challenge,” *CoRR*, abs/1409.0575, 2014.

[96] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille., “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs,” *CoRR*, abs/1606.00915v2, May 2017.

[97] *DeepLab-ResNet-TensorFlow*. (2021). Vladimir and A. Chaudhry. [Online]. Available: https://github.com/DrSleep/tensorflow-deeplab-resnet.

[98] M. A. Rahman and Y. Wang, “Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation,” in *Advances in Visual Computing*, ISVC 2016, G. Bebis, R. Boyle, B. Parvin, D. Koracin, F. Porikli, S. Skaff, A. Entezari, J. Min, D. Iwai, A. Sadagic, C. Scheidegger and T. Isenberg, Ed., 2016, pp. 234-244, doi: 10.1007/978-3-319-50835-1_22.

[99] M. Everingham, S. M. A. Eslami and L. Van Gool, “The PASCAL Visual Object Classes Challenge: A Retrospective,” *Int. J. Comput. Vision*, vol. 111, pp. 98-136, Jan. 2015, doi: 10.1007/s11263-014-0733-5.

[100] B. Hariharan, P. Arbelez, L. Bourdev, S. Maji, and J. Malik, “Semantic contours from inverse detectors,” in *2011 Int. Conf. Comput. Vision*, pp. 991–998, Nov. 2011, doi: 10.1109/ICCV.2011.612634.

[101] S. Ruder, “An overview of gradient descent optimization algorithms,” *CoRR*, abs/1609.04747, Jun. 2017.

[102] S. Wang, S. Wang, X. Zhang, S. Wang, S. Ma and W. Gao, “Scalable facial image compression with deep feature reconstruction,” *2019 IEEE Int. Conf. Image Process. (ICIP)*, Aug. 2019, pp. 2691-2695.

[103] F. Schroff, D. Kalenichenko and J. Philbin, “FaceNet: A Unified Embedding for Face Recognition and Clustering,” *Proc. IEEE Conf. Comput. Vision and Pattern Recognit. (CVPR)*, Jun. 2015, pp. 815-823.

[104] G. Mai, K. Cao, P. Yeun and A. K. Jain, “On the Reconstruction of Face Images from Deep Face Templates,” *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 41, no. 5, pp. 1188-1202, May 2019, doi: 10.1109/TPAMI.2018.2827389.

[105] J. Ballé, V. Laparra and E. P. Simoncelli, “End-to-end Optimized Image Compression,” *CoRR*, abs/1611.01704, Aug. 2018.

[106] Q. Cao, L. Shen, W. Xie O. M. Parkhi and A. Zisserman, “VGGFace2: A Dataset for Recognising Faces across Pose and Age,” *2018 13th IEEE Int. Conf. Autom. Face and Gesture Recognit.*, May 2018, pp. 67-74, doi: 10.1109/FG.2018.00020.

[107] K. He, X. Zhang, S. Ren and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, Dec. 2015, pp. 1026-1034.

[108] D.P Kingma and J. Ba, “Adam: A method for stochastic optimization,” *ICLR,* arXiv:1412.6980, 2015.

[109] G. B. Huang, M. Mattar, T. Berg and E. Learned-Miller, “Labeled Faces in the Wild: A Database forStudying Face Recognition in Unconstrained Environments,” *Workshop Faces in ‘Real-Life’ Images: Detection, Alignment and Recognit.*, Oct. 2008.

[110] S. C. Park, M. K. Park and M. G. Kang, “Super-resolution image reconstruction: a technical overview,” *IEEE Signal Process. Mag.*, vol. 20, no. 3, pp. 21-36, May, 2003, doi: 10.1109/MSP.2003.1203207.

[111] J. Deng, J. Guo, Y. Zhou, J. Yu, I. Kostia and S. Zafeiriou, “RetinaFace: Single-stage Dense Face Localisation in the Wild,” *CoRR*, vol. abs/1905.00641, pp. 1-10, May, 2019.

[112] retinaface. (2022). S. I. Serengil. [Online]. Available: https://github.com/serengil/retinaface.

[113] RetinaFace-tf2. (2020). S. Bertrand. [Online]. Available: https://github.com/StanislasBertrand/RetinaFace-tf2.

[114] S. Yange, P. Luo, C.C. Loy X. Tang, “WIDER FACE: A Face Detection Benchmark,” in *IEE Conf. Comput. Vision and Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5525-5533.

[115] WIDER FACE: A Face Detection Benchmark. (2017). S. Yang. [Online]. Available: http://shuoyang1213.me/WIDERFACE/.

[116] Q. Cao, L. Shen, W. Xie, O. M. Parkhi and A. Zisserman, “VGGFace2: A dataset for recognising faces across pose and age,” *arXiv*, vol. arxiv.1710.08092v2, May, 2018, doi: 10.48550/ARXIV.1710.08092.

[117] J. Deng, J. Guo, J. Yang, N. Xue, I. Kotsia and S. Zafeiriou, “ArcFace: Additive Angular Margin Loss for Deep Face Recognition,” *arXiv*, vol. arxiv.1801.07698v3, Feb, 2019, doi: 10.48550/ARXIV.1801.07698.

[118] VGGFace2-pytorch (2018). cydonia. [Online]. Available: https://github.com/cydonia999/VGGFace2-pytorch.

[119] VGGFace2 Dataset. Q. Cao, L. Shen, W. Xie, O. M. Parkhi and A. Zisserman. [Online]. Available: https://www.robots.ox.ac.uk/~vgg/data/vgg_face2/.

[120] Y. Guo, L. Zhang, Y. Hu, X. He and J. Gao, “MS-Celeb-1M: A Dataset and Benchmark for Large-Scale Face Recognition,” *arXiv*, vol. abs/1607.08221, Jul, 2016, doi: 10.48550/ARXIV.1607.08221.

[121] Z. Cheng, H. Sun. M. Takeuchi and J. Katto, “Learned Image Compression with Discretized Gaussian Mixture Likelihoods and Attention Modules,” *arXiv*, pp. 1-15, Mar., 2020, doi: 10.48550/ARXIV.2001.01568.

[122] CompressAI. (2022). InterDigitalInc. [Online]. Available: https://github.com/InterDigitalInc/CompressAI/.

[123] Image Compression. (2022). CompressAI. [Online]. Available: https://interdigitalinc.github.io/CompressAI/zoo.html.

[124] COSINEEMBEDDINGLOSS (2022). PyTorch. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.CosineEmbeddingLoss.html.

[125] F. Schroff, D. Kalenichenko and J. Philbin, “FaceNet: A Unified Embedding for Face Recognition and Clustering,” in Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognit. (CVPR), Jun. 2015, pp. 815-823.

[126] O. M. Parkhi, A. Vedaldi, A. Zisserman, “Deep Face Recognition,” in Brit. Mach. Vision Conf., 2015, pp. 1-12.

[127] “Dynamic Score Combination: A Supervised and Unsupervised Score Combination Method.” ResearchGate. https://www.researchgate.net/figure/An-example-of-a-ROC-curve-its-AUC-and-its-EER_fig1_225180361 (accessed Jul. 03, 2022).

[128] Labeled Faces in the Wild, University of Massachusetts Amherst, 2018. [Online]. Available: http://vis-www.cs.umass.edu/lfw/.

[129] Losses. (2022). PyTorch Metric Learning. [Online]. Available: https://kevinmusgrave.github.io/pytorch-metric-learning/losses/#arcfaceloss.

[130] JPEG AI Quality Assessment Framework. (2022). JPEG AI. [Online]. Available: https://gitlab.com/wg1/jpeg-ai/jpeg-ai-qaf/-/tree/main/examples.

[131] Downloads. (2020). Kakadu Software. [Online]. Available: https://kakadusoftware.com/documentation-downloads/downloads/.

[132] HEVC HM reference software. (2023). Fraunhofer HHI. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/HM.

[133] NVIDIA Corporation. “Compare GeForce Graphics Cards”. NVIDIA. https://www.nvidia.com/en-eu/geforce/graphics-cards/compare/?section=compare-specs (accessed May 24, 2023).

Appendix A

A. Performance Evaluation of Learning-based Image Codecs

This appendix reports the selected images from the JPEG AI database for evaluation of the learning-based image coding solutions FRICwRNN, FactMSE, FactMS-SSIM, HyperMSE and HyperMS-SSIM, in section 3.3.

Figure 47 – *Selected images from JPEG AI database: TE00 (left), TE03 (middle) and TE04 (right) [80].*

Figure 48 - *Selected images from JPEG AI database: TE08 (left) and TE16 (right) [80].*

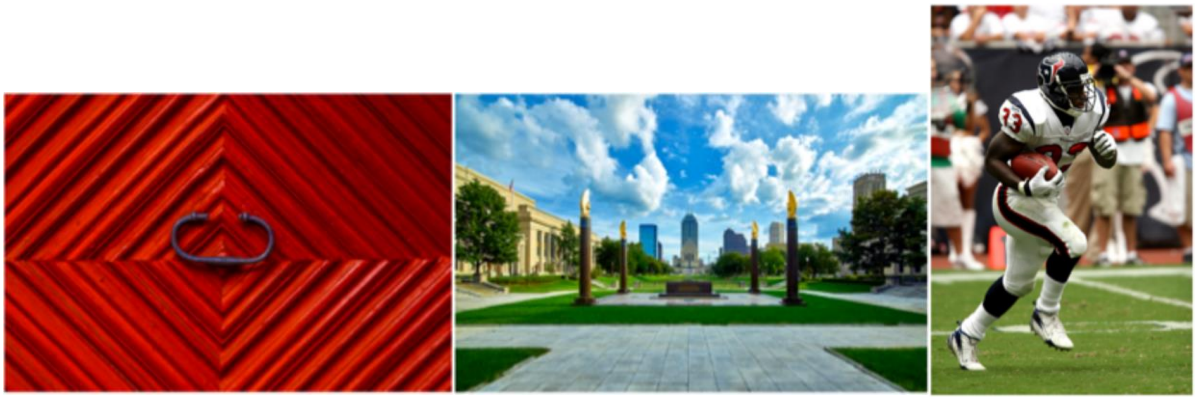

Figure 49 - Selected images from JPEG AI database: TE20 (left), TE25 (middle) and TE39 (right) [80].