Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Eletrotécnica
Mestrado em Eng.ª Eletrotécnica

LEARNING-BASED IMAGE COMPRESSION USING MULTIPLE AUTOENCODERS

RÚBEN DUARTE ANTÓNIO

Leiria, February 2023

Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Departamento de Engenharia Eletrotécnica
Mestrado em Eng.ª Eletrotécnica

LEARNING-BASED IMAGE COMPRESSION USING MULTIPLE AUTOENCODERS

RÚBEN DUARTE ANTÓNIO

Dissertation performed under the supervision of Professor Pedro Antonio Amado de Assunção, Professor Sérgio Manuel Maciel de Faria, and Professor Luís Miguel de Oliveira Pegado de Noronha e Távora

Leiria, February 2023

3

ACKNOWLEDGMENTS

First and foremost, I would like to thank God for guiding my entire journey as a student because I recognise that nothing would be possible without Him.

I would like to thank my advisers Dr. Pedro António Amado Assunção, Dr. Luís Miguel de Oliveira Pegado de Noronha e Távora, and Dr. Sérgio Manuel Maciel de Faria for their guidance and support which were crucial for this research.

A special thanks to my parents, Virgilio Rosa Antonio and Carla Sofia da Cunha Duarte, my brother Rodrigo Duarte António and my girlfriend Ane Caroline de Matos, for their unconditional support.

Finally, I would like to thank Instituto de Telecomunicações for hosting this work, and for providing the fundamental research environment and all my colleagues at the Multimedia Signal Processing group from Leiria Delegation of Instituto de Telecomunicações, for all the knowledge shared and support given.

RESUMO

Aplicações de vídeo em ambientes avançados (por exemplo, cidades inteligentes) fazem emergir novos desafios associados a sistemas cada vez mais complexos com requisitos exigentes em várias áreas, tais como vigilância, visão computacional na indústria, medicina, entre outras. Como consequência, uma enorme quantidade de dados é adquirida para ser analisada por máquinas orientadas para tarefas. Devido ao enorme volume de dados gerados é necessária a implementação de métodos de compressão eficientes para reduzir o seu armazenamento. Para este objetivo têm vindo a ser investigadas novas técnicas para compressão de imagens que não seguem a arquitetura dos codificadores clássicos.

Neste âmbito, esta dissertação apresenta um trabalho de pesquisa sobre métodos de compressão de imagens utilizando algoritmos de aprendizagem automática. Realizando uma analise das as características dos diferentes algoritmos existentes na literatura, descrevendo um estudo de simulação e resultados comparativos de desempenho na compressão de imagens. Em particular, é dado maior ênfase a implementações com redes convolucionais, nomeadamente aos *autoencoders*.

Foram estudadas, implementadas e avaliadas duas abordagens de compressão imagens utilizando *autoencoders*. Uma compressão orientada a elementos da imagem e outra orientada a imagens de alta resolução (UHD e imagens 360º). Como referido, numa primeira abordagem, foi considerado um cenário de vídeo vigilância que inclui diversos elementos da imagem, como pessoas carros, rostos, bicicletas e motas, e foi desenvolvido um método de compressão utilizando *autoencoders* orientada a aplicações de visão computacional, isto é as imagens descodificadas serem processadas por uma máquina. Nesta primeira abordagem foram analisados o desempenho em termos da qualidade da imagem descodificada e do desempenho algoritmico no seu processamento ou extração de informação. A segunda abordagem focou-se na utilização de imagens de alta resolução evoluindo o método utilizado na abordagem anterior para a utilização de características da imagem, como a variância, os gradientes ou a PCA de *features*, ao invés do conteúdo que a imagem representa.

Considerando a primeira abordagem em comparação com a norma de VVC (Versatile Video Coding), o método proposto atinge uma eficiência de codificação significativamente melhor do que a do VVC, por exemplo, até 46,7% de redução

do BD-rate. A precisão das tarefas de visão computacional é também significativamente maior quando realizadas sobre objetos visuais comprimidos com o método proposto, em comparação com as mesmas tarefas realizadas sobre os mesmos objetos comprimidos com o VVC. Estes resultados demonstram que a abordagem utilizada com base em *autoencoders* é uma solução mais eficiente para a compressão de objetos visuais do que a codificação com a norma VVC. Considerando a segunda abordagem, embora sejam obtidos melhores resultados que o VVC nos subconjuntos de imagem de teste, a abordagem apresentada apenas apresenta ganhos significativos considerando imagens 360º.

Palavras-chave: Compressão baseada na aprendizagem, *autoencoders*, objetos, vídeo vigilância, imagens UHD, imagens 360º, redes convolucionais

ABSTRACT

Advanced video applications in smart environments (e.g., smart cities) bring different challenges associated with increasingly intelligent systems and demanding requirements in emerging fields such as urban surveillance, computer vision in industry, medicine and others. As a consequence, a huge amount of visual data is captured to be analyzed by task-algorithm driven machines. Due to the large amount of data generated, problems may occur at the data management level, and to overcome this problem it is necessary to implement efficient compression methods to reduce the amount of stored resources.

This thesis presents the research work on image compression methods using deep learning algorithms analyzing the properties of different algorithms, because recently these have shown good results in image compression. It is also explained the convolutional neural networks and presented a state-of-the-art of autoencoders.

Two compression approaches using autoencoders were studied, implemented and tested, namely an object-oriented compression scheme, and algorithms oriented to high resolution images (UHD and 360º images). In the first approach, a video surveillance scenario considering objects such as people, cars, faces, bicycles and motorbikes was regarded, and a compression method using autoencoders was developed with the purpose of the decoded images being delivered for machine vision processing. In this approach the performance was measured analysing the traditional image quality metrics and the accuracy of task driven by machine using decoded images. In the second approach, several high resolution images were considered adapting the method used in the previous approach considering properties of the image, like variance, gradients or PCA of the features, instead of the content that the image represents.

Regarding the first approach, in comparison with the Versatile Video Coding (VVC) standard, the proposed approach achieves significantly better coding efficiency, e.g., up to 46.7% BD-rate reduction. The accuracy of the machine vision tasks is also significantly higher when performed over visual objects compressed with the proposed scheme in comparison with the same tasks performed over the same visual objects compressed with the VVC. These results demonstrate that the learning-based approach proposed is a more efficient solution for compression of visual objects

than standard encoding. Considering the second approach although it is possible to obtain better results than VVC on the test subsets, the presented approach only presents significant gains considering 360º images.

Keywords: Learning-based compression, autoencoders, visual objects, video surveillance, UHD images, 360º images, convolutional neural networks

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AE	Autoencoder.
BPP	Bits Per Pixel.
CABAC	Context-Adaptive Binary Arithmetic Coding.
CL	Convolutional Layers.
CNN	Convolutional Neural Network.
DCT	Discrete Cosine Transform.
HEVC	High Efficiency Video Codding.
JPEG	Joint Photographic Experts Group.
KLT	Karhunen-Loève Theorem.
MSE	Mean Squared Error.
MS-SSIM	Multi-Scale Structural Similarity.
PCA	Principal Component Analysis.
PSNR	Peak Signal-to-Noise Ratio.
ROI	Region Of Interest.
UHD	Ultra High Definition.

VAE	Variational Autoencoder.
VMAF	Video Multimethod Assessment Fusion.
VVC	Versatile Video Coding.

1

INTRODUCTION

1.1 CONTEXT AND MOTIVATION

In recent years, the utilisation of video applications in smart environments has grown and brought different challenges associated with increasingly intelligent systems with specific requirements in emerging fields such as urban surveillance, computer vision in industry, medicine, among others. As a consequence, a huge amount of visual information is captured, requiring to develop efficient storage and delivery systems capable of supporting the corresponding huge amount of digital data. Such requirements increased the importance of the study and the development of efficient compression algorithms for this kind of environments.

Considering the application environment of smart surveillance, the use of 360º video is an interesting approach because with only one camera it is possible to view all angles of a scene. A 360º camera is equipped with, at least, two cameras, allowing to simultaneously capture single images or videos. Then, the images captured by each lens are stitched together to generate 360-degree videos and photos.

Recently deep learning algorithms have shown good results in image compression compared to the traditional encoders [1] [2] [3] [4]. This fact, explains the gradual growth of their applications in image compression envolving different algorithms such as convolutional neural network (CNN), autoencoders (AE) and variational autoencoders (VAE). In this context, two different approaches have been under research: either adding learning-based models and coding tools and/or substituting existing ones in conventional hybrid block-based encoders, or developing end-to-end learning-based compression architectures using deep neural networks to find a whole compact representation of the visual content.

1.2 OBJECTIVES

The main objective of this research consisted on the development and implementation of a pipeline to code images more efficiently using autoencoders. In the course of this work, the following tasks were carried out to accomplish such objective:

- Study and implementation of an autoencoder and networks to detect objects.
- Study and implementation of a framework to detect and coding objects with autoencoders.
- Study of the different functional blocks in an autoencoder, in order to improve the coding performance such as quantization, architecture and entropy coding.
- Study and implementation of a coding framework using different autoencoders simultaneously.
- Performance evaluation of the proposed methods.

1.3 DISSERTATION STRUCTURE

This document is organized as follows: Chapter 2 gives an overview of on machine learning algorithms, with a particular focus on CNNs, as well as the state-of-art of autoencoders. Chapter 3 describes the proposed framework for compression of visual objects, presents the used datasets and the obtained results. Chapter 4 presents the proposed framework for compression of 360º and ultra high definition (UHD) images, describes the dataset and the graphical image properties used to create the different subsets, and presents the obtained results. Chapter 5 presents the conclusions of the developed work and several suggestions of improvement for future work.

2

BACKGROUND ON CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE CODING

In this chapter, the basics of convolutional neural networks (CNN) are presented as these are structural elements in deep learning architectures used in image compression. The chapter starts with an overview of deep learning algorithms related to image compression. Then, JPEG AI is presented and finally state-of-the-art autoencoders are described.

2.1 OVERVIEW

Many machine learning algorithms are inspired by the human brain using a structure known as artificial neural networks. In this way, computers can perform assignments similarly to humans, such as learning by example. This allows the possibility of using computers to learn complex functions such as image identification, speech recognition, prediction making and others. Deep learning was first theorized in the 1980s [5], but only recently deep learning has achieved high levels of accuracy because it is necessary a large amount of labelled data and substantial computing resources.

Deep learning has changed the way of representing some problems because the algorithms are trained by using large sets of labelled data and neural network architecture that learn complex features from data without human interaction. The main difference between deep learning and machine learning is the type of input data and the methods used in the computational learning process. Machine learning in general might require that features are previously extracted from images, while deep learning eliminates some of this data pre-processing because the relevant features are automatically identified. Furthermore, deep learning algorithms estimate the model parameters by means of different types of learning, such as supervised learning which uses labelled datasets to make predictions, or unsupervised learning which does not require labelled datasets because the patterns in the data are detected, or clustering by any distinguishing features.

2.2 DEEP LEARNING ALGORITHMS

To use deep learning to solve a problem it is possible to choose different algorithms, such as CNNs, autoencoders, transformers, generative adversarial networks and others. Each of these algorithms has different characteristics and depending on the problem under consideration some kind of analysis is needed to determine which one is the best. In this section an overview of some of these algorithms is presented, starting by CNNs and then autoencoders and transformers. A specific example of a standard compression algorithm based on autoencoders is also presented, the JPEG AI [6].

2.2.1 *Convolutional Neural Network*

A CNN is a specific processing structure used by deep learning algorithms mostly for image recognition, classification and processing. For instance, the identification of a certain visual object using CNNs goes through the analysis of specific characteristics (features) along the image. The features are determined by performing several convolutions on the image with multiple filters (kernels), where each one tries to identify a different feature, also considering geometric variations (e.g rotation, translation) that increase the robustness of the results provided by the network. To understand the operations performed in a CNN it is important to take into account that a colour image is represented by a 3D matrix where the combinations of red, green and blue pixels [R,G,B] are stored, as represented in Figure 1.

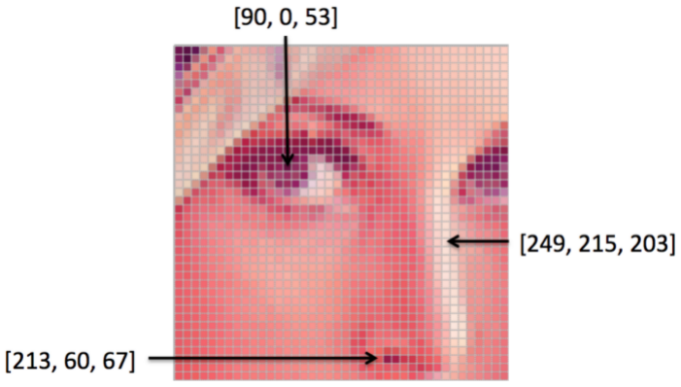

Figure 1: Organisation of an image, showing the (R,G,B) values of 3 pixels[7].

Typically, a CNN architecture is composed of multiple layers such as convolutional layers, activation, pooling and fully connected layers. CNNs are mainly composed of

convolutional layers because the features of the input images are captured by this type of layers. The convolution is a linear process that involves the multiplication and addition of the image samples by a set of weights with dimensions $K_S \times K_S$, i.e. the kernel. Figure 2 shows the process previously described using a 3×3 kernel. The result of convoluted features after the convolution process is then delivered as input to an activation layer.

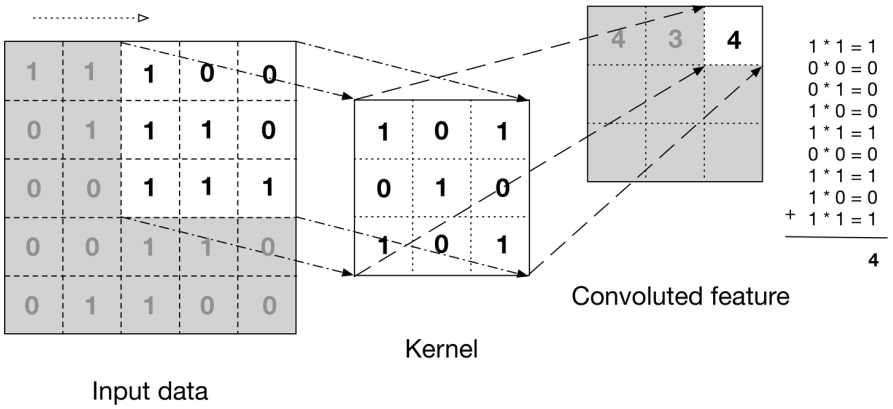

Figure 2: Convolution layer [8].

Some parameters directly characterise a convolutional layer and consequently the shape of the convoluted features:

- Stride is the size of the step that the kernel is shifted in each iteration. If the stride is two then the kernel is shifted two pixels at a time;
- Padding is the process where some pixels are added around the image before the convolution operation, the value of the added pixels is usually 0;
- Dilation is the space between kernel elements, the default value is 1. If the dilation is greater than 1, $dilation - 1$ number of "0" is added between all values of the considered kernel;
- Depth is the number of filters used in a convolution layer;

The size of the output of the convoluted feature is formulated in [9], as

$$H_{out} = \frac{H_{in} + 2 \times P[0] - D[0] \times (K_S - 1) - 1}{S[0]} + 1 \tag{1}$$

$$W_{out} = \frac{W_{in} + 2 \times P[1] - D[1] \times (K_S - 1) - 1}{S[1]} + 1 \tag{2}$$

where P is the number of the padding, D is the value of dilation, K_S denotes the size of the kernel, S is the value of the stride and H_in W_in, H_out W_out represents the height and the width of the input data and convoluted features, respectively. A large number of specific filters can be learned in parallel on a given training dataset and very specific features will be obtained that can be detected anywhere on the input images.

Activation layers are placed directly after the convolutional layer to introduce non-linearity because the convolution is a linear operation. It is possible to introduce non-linearity using different types of activation functions such as sigmoid, tanh, ReLU, Leaky ReLU, Maxout and ELU. The sigmoid function curve is similar to an S shape, as shown in Figure 3, it restricts the output value between 0 and 1.

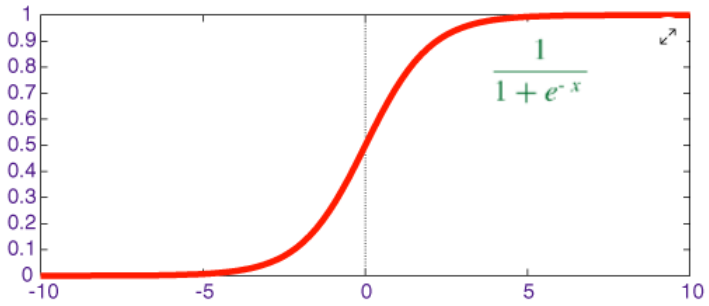

Figure 3: Sigmoid function [10].

The tanh function curve is similar to the sigmoid function but the input values are limited between -1 and 1, as shown in Figure 4.

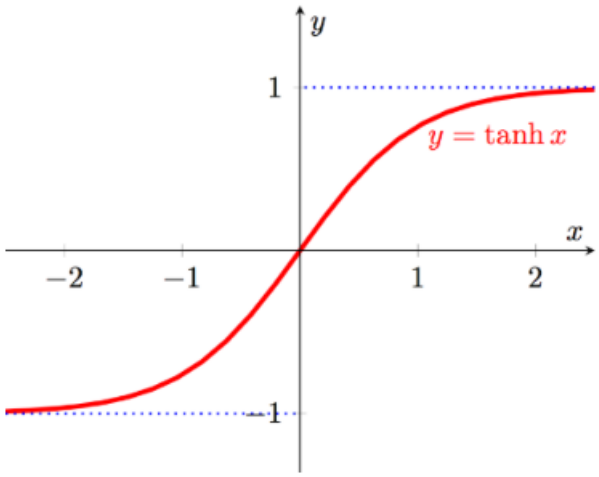

Figure 4: Tanh function [11].

The ReLU function is one of the most popular functions and its greatest advantages is that it does not activate all neurons at the same time, because the negative input values are passed to zero and consequently the neuron does not get activated, as shown in Figure 5. The disadvantage of ReLU is in the negative region where the gradient is zero, which does not allow updating the weights in the backpropagation process. To overcome this problem an alternative was created, named Leaky ReLU, as shown in Figure 6. In this case, the gradient of the curve on the negative side is not zero, which favours the convergence of the training process.

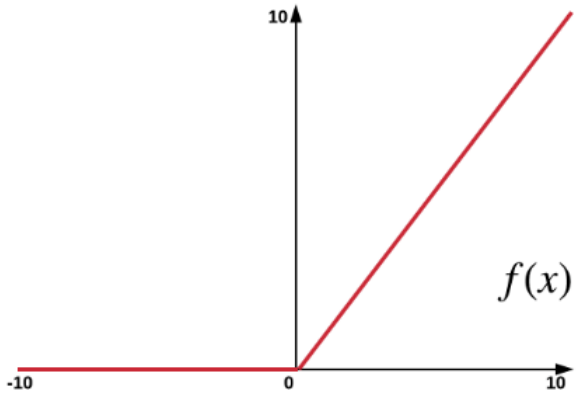

Figure 5: ReLU function [12].

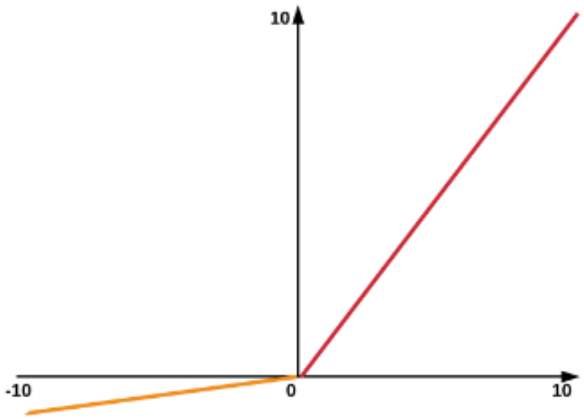

Figure 6: Leaky ReLU function [12].

The pooling layer can be placed after the activation layer to reduce the spatial size of the representation, and the number of parameters to learn and consequently reduce the amount of computation performed in the network. So the main goal of the pooling layers is to summarise the features in a feature map generated by a convolution layer. There are two types of pooling layers known as Average Polling and Max Pooling. The Average Pooling computes the average of the elements present in the region of the feature map covered by the filter. The output of the average

pooling layer would be a feature map containing the average of features present in a filter. Figure 7 shows an example of average pooling using a filter 2 by 2 and a stride of 2.

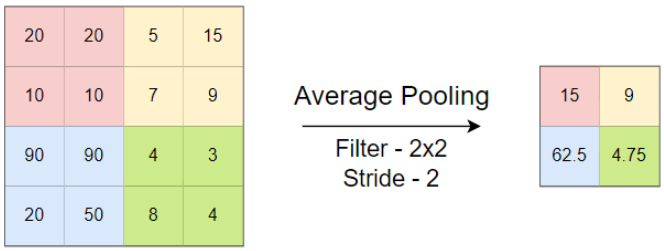

Figure 7: Example of Average Pooling.

The Max Polling selects the maximum element from the region of the feature map covered by a filter. While average pooling gives the average of features present in a patch, max pooling returns a feature map containing the most prominent features. Figure 8 shows an example of average pooling using a filter 2 by 2 and a stride of 2.

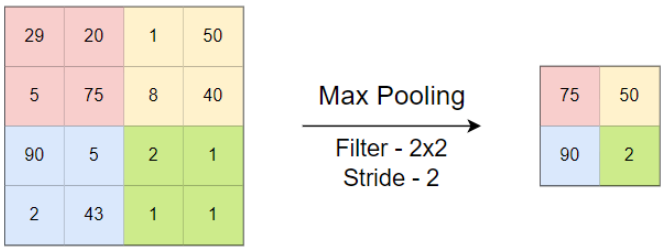

Figure 8: Example of Max Pooling.

Unlike the convolution layer, the pooling layer does not change the depth of the feature map, and the dimensions of its output using the default values of dilation and padding, are given in [13] by,

$$H_{out} = \frac{H_{in} - K_S}{S[0]} + 1 \tag{3}$$

$$W_{out} = \frac{W_{in} - K_S}{S[1]} + 1 \tag{4}$$

where K_S denotes the size of the kernel and S the value of the stride.

The fully connected layer is usually the last layer in the convolutional neural network, comprising the so called classification block. This layer receives the output

of the flatten layer, which is an one-dimensional layer, applies a linear transformation to this vector through a weighting matrix and then introduces a non-linear operation. Multiple fully connected layers can be added together, based on the depth of the classification model. Finally the output of the fully connected layer is linked to a Softmax or a Sigmoid function for probability distribution over the total number of classes considered. Figure 9 shows an example of a fully connected layer to classify three different objects.

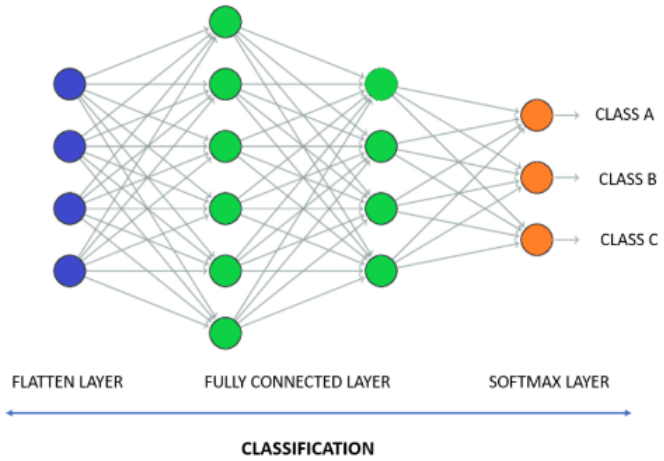

Figure 9: Example of fully connected layer [14].

2.2.2 *Autoencoder*

An autoencoder (AE) is a deep learning architecture that compresses the input data into a latent-space representation and reconstructs an estimated output from this latent representation. Since the latent representation uses less data than the original, autoencoders have been investigated for image compression as an alternative to classical standard schemes. Figure 10 represents a generic autoencoder, which is comprised of three main components: the encoder, the latent space and the decoder.

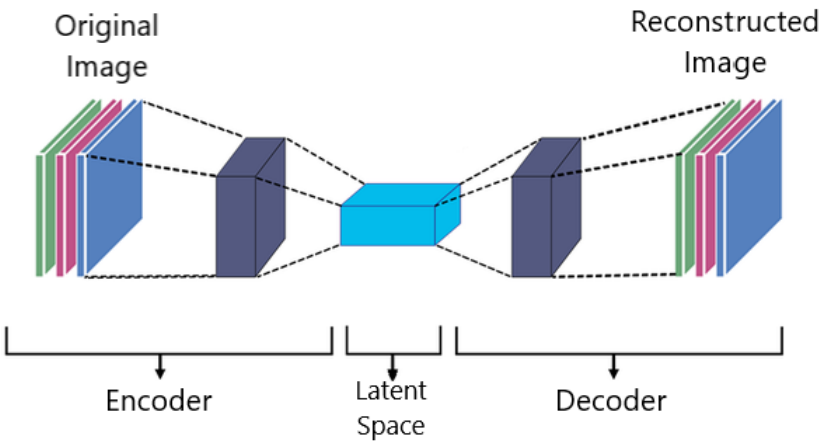

Figure 10: Generic autoencoder [15].

The encoder is the network component responsible for learning how to compress efficiently the original image into the smaller latent representation. This latent space is therefore a compact representation of the original image, and the decoder part is responsible for learning how to reconstruct the original image from the latent space information. In most cases the decoded image presents some loss compared to the original image because the latent space itself is not a lossless representation and also the reconstruction is not perfect. The main characteristic of autoencoders is their capability of finding a compact representation that provides compression but they can be used for a variety of functions, such as removal of image noise [16] and anomaly detection [17] among others.

There are different types of autoencoders depending on their architecture and also intended applications, where the main types are denoising autoencoder, sparse autoencoder, convolutional autoencoder and variational autoencoder. The architecture of these autoencoders are slight variations of Figure 10. The goal of denoising autoencoder is to remove noise of the input image. In this case, the autoencoders are optimised to remove noise from similar data (images of the same class for example), and for this purpose is added a block before the encoder introduces random noise in the input data [18]. A sparse autoencoder is an autoencoder whose training criterion involves a sparsity penalty [19]. The loss function penalizes activations of hidden layers and with this a smaller number of neurons are activated. The main goal of this method is to eliminate redundant information guaranteeing the autoencoder learns latent representation with reduced size. Convolutional Autoencoders use convolution layers in the encoder and deconvolution layers in the decoder. They are state-of-the-art for unsupervised learning of convolutional filters [20]. The variational autoencoders use a variational approach to generate the latent representation, which

results in an additional loss component and an estimator for the training network named the stochastic gradient variational bayes estimator [21]. For applications in image compression, the most popular autoencoder is the convolutional autoencoder.

2.2.3 *Learning-based image encoder used in JPEG AI*

The JPEG AI is a framework developed by the JPEG standardisation group and its main purpose is create a learning-based image coding standard. JPEG AI targets a wide range of applications like visual surveillance, image collection storage and management, autonomous vehicles and devices, media distributions and others [6]. Figure 11 shows high-level JPEG AI framework, comprising three main pipelines, each one compressing images for a different task. In fact, besides the expected standard image decoding scheme, it includes image processing and computer vision tasks where the latent representation of the original image is optimised to be a feature-rich representation that allows good performance in these tasks. Examples of computer vision tasks are image classification, object identification, region segmentation whilst image processing tasks, might involve noise removal, upscaling, quality enhancement or other types of processing that have classical image processing analytical counterparts, i.e., non-learning approaches.

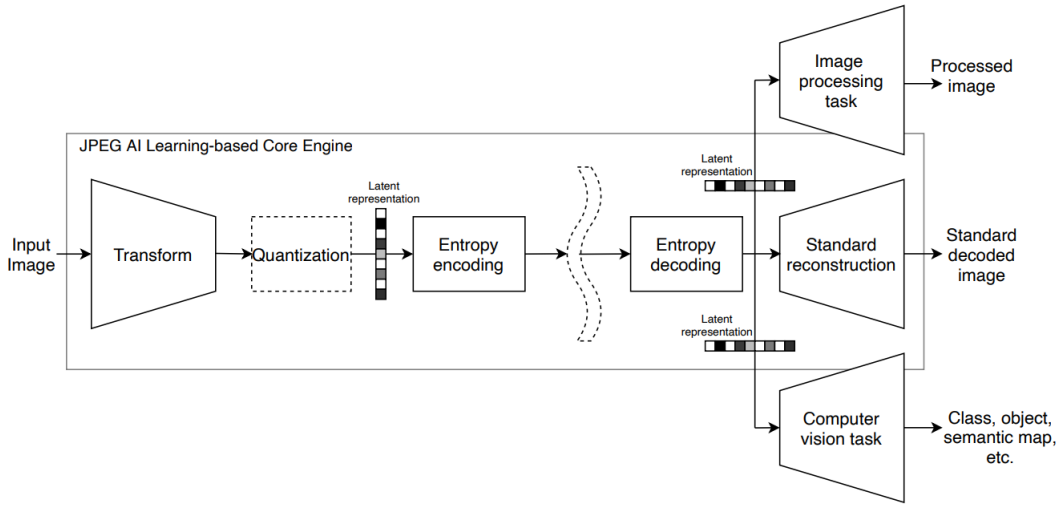

Figure 11: JPEG AI learning-based image coding framework [6].

A JPEG AI dataset was established in order to assess the performance of state-of-the-art learning-based image coding algorithms. It comprises specific subsets for training, validation and test, containing a wide diversity of images, namely in terms of their characteristics, such as content type and spatial resolution. In terms of data,

it has PNG images (RGB color components, non-interlaced), with spatial resolution from 256x256 to 8K (8 bits); the training set has 5264 images, while the validation and test have 350 images [22].

2.2.4 *Transformers*

The transformer neural network is a novel architecture created to solve sequence-to-sequence tasks while handling long range dependencies [23]. The development of the transformer model was motivated to respond to some limitations presented by CNNs such as the convolution filter only being able to characterize short-range spatial correlation within the receptive field. Then, if the input images have different type of content than those used in the training process, worse performance is obtained despite the intensive computation [24]. Figure 12 represents the architecture of a transformer, consisting of a stacked self-attention, point-wise and fully connected layers for both the encoder and decoder in left and right sides of figure, respectively.

Recently transformer neural networks have achieved promising results compared to the architectures that only use CNNs. This has been investigated for different vision tasks, as demonstrated in [25] and [26]. This is justified by the fact the transformers can well capture the long-range correlation for better information embedding, by dividing the input images into patches and treating them as sequential words as in natural language processing for computation [23]. In [24] and [27] a hybrid architecture is used, based on CNNs with transformers with the main focus in image compression. It was shown that the performance obtained by these architectures is similar to CNNs.

2.3 IMAGE COMPRESSION WITH AUTOENCODERS

Recently autoencoders have achieved promising results in image compression. The first learning-based image compression scheme that achieved better performance than JPEG2000 in terms of peak signal -to-noise ratio (PSNR) and multi-scale structural similarity (MS-SSIM) was presented in [1]. In [28] it is described the baseline learning-based model shown in Figure 13(a). The image compression process is formulated by

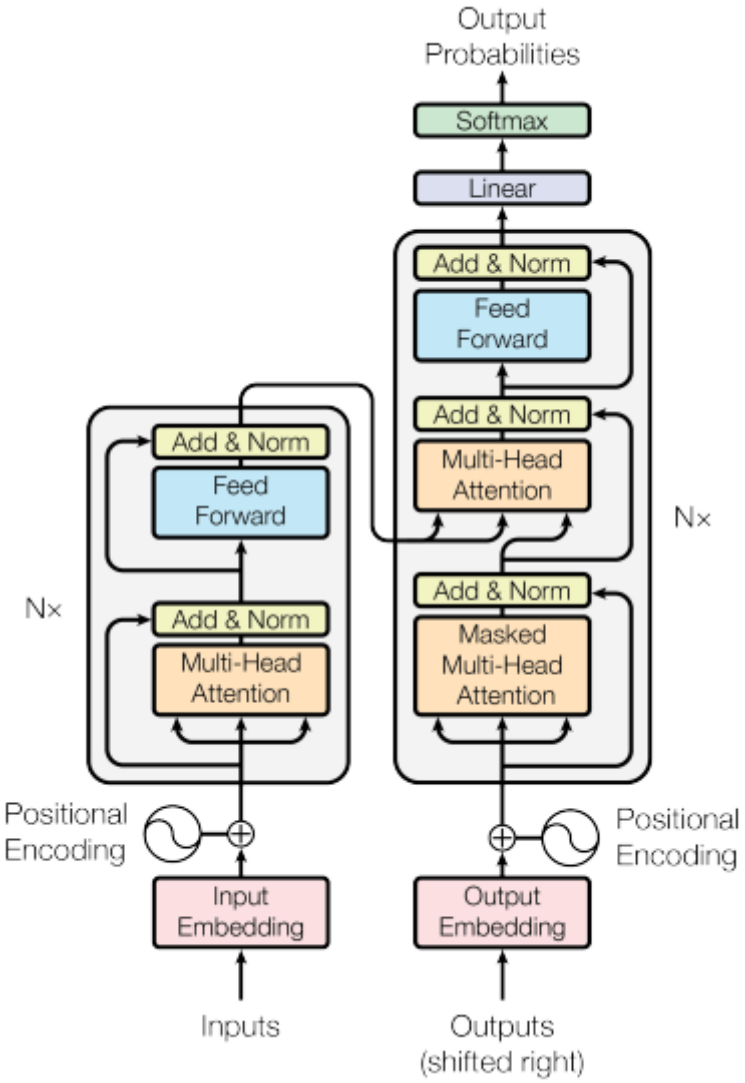

Figure 12: The Transformer - model architecture [23].

$$
\begin{aligned}
y &= g_a(x;\phi) \\
\hat{y} &= Q(y) \\
\hat{x} &= g_s(\hat{y};\theta)
\end{aligned}
\tag{5}
$$

where x, $\hat{x}$, y and $\hat{y}$ are the original images, reconstructed images, the latent representation before quantization, and latent representation after quantization, respectively. ϕ and θ are optimisation parameters of analysis and synthesis transforms. $U|\,Q$ are the quantization and entropy coding steps, which will be later addressed in 2.3.1 and 2.3.2, respectively.

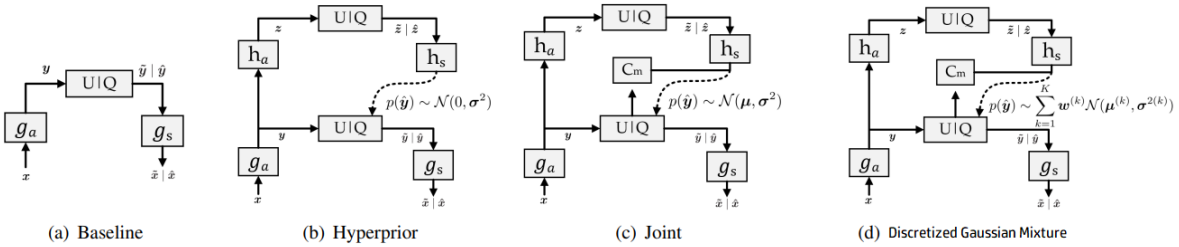

Figure 13: Operational diagram of learned compression [4].

In [2] a hyperprior model was presented as shown in Figure 13(b), by introducing a side information z to capture spatial dependencies among the latent space (y), formulated by

$$\begin{aligned} z &= h_a(y;\phi_h) \\ \hat{z} &= Q(y) \\ p_{\hat{y}|\hat{z}}(\hat{y}|\,\hat{z}) &\leftarrow h_s(\hat{z};\theta_h) \end{aligned} \tag{6}$$

where h_a and h_s denote the analysis and synthesis transform in the auxiliary autoencoder, ϕ_h and θ_h are optimisation parameters and $p_{\hat{y}|\hat{z}}(\hat{y}|\,\hat{z})$ are estimated distributions conditioned on $\hat{z}$. A zero-mean Gaussian distribution with scale parameters ($\sigma^2 = h_s(\hat{z};\theta_h)$) is used to estimate the $p_{\hat{y}|\hat{z}}(\hat{y}|\,\hat{z})$. In [29] a jointly model was presented as shown in Figure 13(c), by introducing a more accurate entropy model using an autoregressive context model (denoted as C_m), a mean and scale hyperprior. In [4] a Discretized Gaussian Mixture model was presented, as shown in Figure 13(d), by introducing a more accurate entropy model using a Gaussian distribution model to estimate the $p_{\hat{y}|\hat{z}}(\hat{y}|\,\hat{z})$, where K is the number of the mixture gaussian used and $w^{(k)}$ is the weights of the gaussian.

For the construction of an autoencoder it is necessary to take into account the number of convolutional layers in both the encoder and the decoder (usually the architecture of the decoder is symmetrical to that of the encoder), the number of filters per layer, the size of the latent space and the loss function (measures the image information that is lost when comparing the original with the output).

The loss function that is used to optimise the autoencoders during the training phase is formulated by,

$$\mathcal{L} = \mathcal{R} + \lambda * \mathcal{D}(x,\hat{x}) \tag{7}$$

where $\mathcal{R}$ and $\mathcal{D}(x,\hat{x})$ are the bit rate and the distortion terms respectively, and λ controls the rate-distortion trade-off. In the baseline diagram, the $\mathcal{R}$ only depends on the $\hat{y}$ because the hyperprior does not exist in the architecture, otherwise the $\mathcal{R}$ would depend on $\hat{y}$ and $\hat{z}$.

In recent research studies to improve the compression performance of autoencoders, two main components have been investigated such as the network architectures and the entropy models used to encode the latent representation.

In terms of the architecture of the autoencoders, some blocks have been introduced in order to improve their performance. In [30] was introduced the residual block which was shown to have comparable performance with JPEG2000. An attention module was proposed in [4] to force learning-based models pay more attention to complex regions (different parts of the image be adapted to local content), and also achieved improved performance for image restoration [30] and compression [30]. In [31] a concatenated residual module was proposed to further remove the spatial correlation in the latent space, facilitating the information flow, and improving the training of the network.

2.3.1 *Quantization*

Quantization is a necessary component in image compression to reduce the amount of data required to represent the latent space. However, since this is an irreversible operation, it introduces unavoidable distortion. This step produces similar effects as in the discrete cosine transform (DCT) coefficients in standard JPEG compression. In autoencoder, the quantization during the training phase must be differentiable to allow the back-propagation over the gradients, but due to the inherent non-differentiability of round-based quantization, a conventional quantizer cannot be directly incorporated into an autoencoder. In this case it is possible approximate the quantisations of the latent space by using different methods such as a uniform noise approximation [1], soft histogram [32], a stochastic form of binarization [33], and in some studies [30],[32], the derivation was replaced in the backpropagation, but it was guaranteed that the quantized value was correct in forward propagation.

2.3.2 *Entropy Encoder*

In this step the quantized values and the prediction side-information are lossless compressed into a bit-stream using entropy coding methods, like arithmetic encoder (CABAC [34]), context-adaptive entropy models [2, 29, 35], relative entropy coding [36]. The entropy encoder is a crucial module to reduce statistical redundancy within the latent space and is possible to implement in autoencoders because discrete values are considered. The main idea is to exploit the statistics of latent representation with a conditional probability model. As mentioned above, the performance of the autoencoder increases when side information was introduced to capture the spatial dependencies among the latent space, so it is very important to fit the parameterized distribution model to the marginal distribution of the quantized latent space. This was investigated in the recent past and some articles are published trying to model the conditional probability distribution $p_{\hat{y}|\hat{z}}(\hat{y}|\,\hat{z})$ after decoding $\hat{z}$. For instance, in [4] a discretized Gaussian Mixture likelihood was proposed for entropy coding which is characterised by 3 parameters, weight, mean and variance. Afterwards, an improved work was presented in [31] by using a discretized Gaussian-Laplacian-Logistic model distribution. Overall, the impact on compression efficiency was demonstrated to be better when such models are used.

2.4 SUMMARY

This chapter presented an overview of deep learning algorithms, detailing their purpose and characteristics, with a particular focus on convolutional neural networks. The different types of autoencoders are also described in terms of their characteristics and functions. The JPEG AI framework and transformer models were also discussed, alongside with an overview of the state of the art autoencoders using convolutional autoencoders

On the whole, the background for the main objective of this thesis was presented, which is developing a framework to efficiently compress images using multiple autoencoders trained to different classes of objects. The overview carried out showed that deep learning algorithms are a relevant research topic and they can be used for different applications. Therefore this is also a promising approach for new contributions in emerging fields of image compression.

3

LEARNING-BASED COMPRESSION OF VISUAL OBJECTS

This chapter describes the proposal of an efficient learning-based approach to compress relevant visual objects, based on the use of multiple autoencoders, that is one for each object class. First, a detailed description of the proposed compression approach and the autoencoder's model are explained. Then, the datasets used to training the autoencoders and the different training approaches are presented. Finally, the experimental results are analysed by measuring different rate-distortion metrics and the accuracy of some tasks driven by machines.

3.1 PROPOSED APPROACH

To implement this approach a surveillance scenario has been considered because it presents a limited number of objects in the scene, such as people, vehicles and some animals (e.g. dogs). The functional scheme of the proposed approach for compression and delivery of visual objects in smart surveillance applications is shown in Figure 14. For the sake of simplicity only the compression of visual objects is represented. The background is treated as stationary, so it can be extracted, encoded and transmitted using a traditional encoder. The relevant objects, delimited by bounding boxes, are first classified into one of the i predefined classes of interest, which depend on the requirements of the surveillance scenario. As the background in the bounding box itself is not relevant, object segmentation is performed in order to enhance and extract the region of interest, this is explored in section 3.3.1. Then for each class, an end-to-end optimised encoder is used to obtain its compressed representation for transmission to the corresponding decoder and processing through the corresponding machine vision task.

Each encoder-decoder pair (Cj, j=1, 2, ... i) is optimised for a unique object class by training the autoencoder (AE) with visual objects of that class. Since objects of the same class have similar features, this strategy favours the network to learn how better to model those specific features, thus reducing the latent representation's

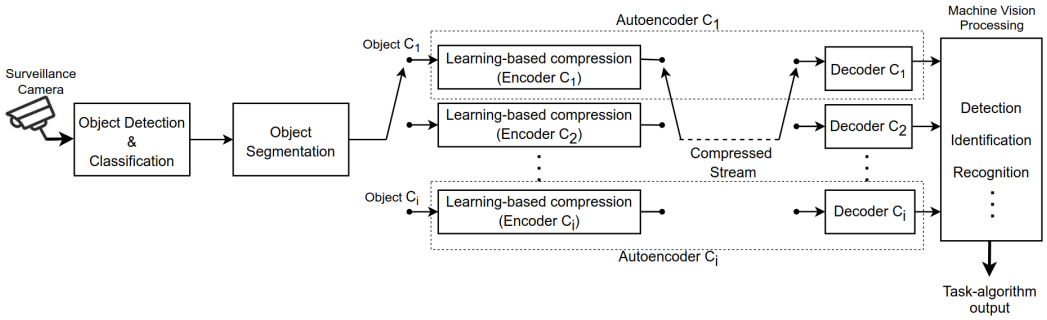

Figure 14: Proposed object-based compression scheme for smart surveillance.

entropy. At the end of the pipeline, after decoding, visual objects are processed by some task whose level of success measures the system's performance.

Within the scheme proposed any architecture can be considered for the learning-based compression module. In the case, the choice followed closely a model proposed by Cheng and collaborators [4], and is presented in Figure 15. The main difference to Cheng's original model is that the attention module were not considered, since their impact on performance was herein limited, as later discussed in section 3.3.1.

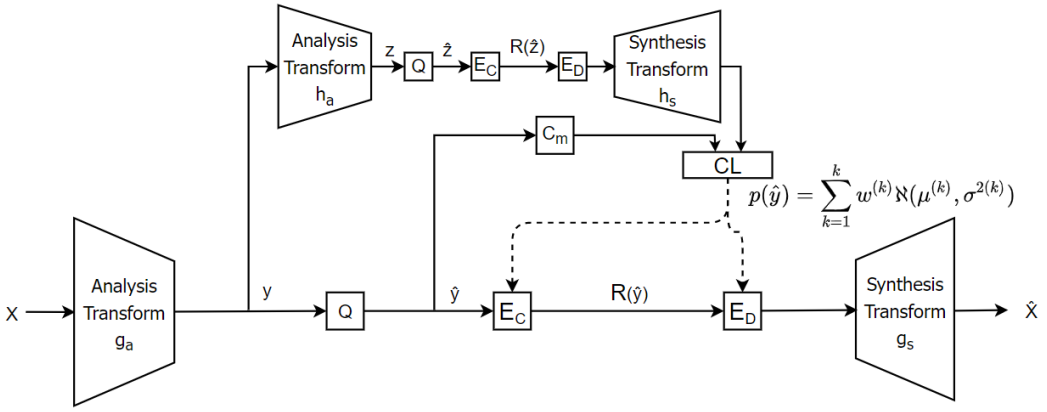

Figure 15: Learning-based network architecture for image compression. E_C: Entropy coding, E_D: Entropy decoding, Q: Quantisation, C_m: Context model, CL: convolutional layers.

In the adopted model (Figure 15), x, $\hat{x}$, y, $\hat{y}$ are the input visual objects, reconstructed objects, latent space before quantization and coded stream, respectively. The expressions $y = g_a(x;\phi); \hat{y} = Q(y); \hat{x} = g_s(\hat{y};\theta)$ represent the analysis, quantization and synthesis transform composed by convolutional layers and activation functions. The set of parameters, ϕ and θ, of the analysis and synthesis transforms are optimised during the training phase. Quantization is approximated by adding uniform noise to keep the operation differentiable during the training phase, while in inference a rounding-based operation is used followed by entropy coding, e.g., arithmetic coding. Besides the main encoder-decoder pipeline, the auxiliary network

(top of the diagram) comprises the analysis synthesis transforms h_a and h_s, respectively, providing a hyperprior by generating the side information $z = h_a(y; \phi_h)$, which captures the spatial correlations of y [2]. The quantized version of z is $\hat{z} = Q(z)$ and the synthesis transform produces an estimate of distribution $p(\hat{y}/\hat{z})$, i.e., $h_s(\hat{z}; \theta_h) \rightarrow p(\hat{y}/\hat{z})$. The parameters ϕ_h, θ_h are jointly optimised with ϕ and θ during training.

A discretised Gaussian mixture is used for the entropy model, where each Gaussian distribution is characterised by 3 parameters: weight, mean and variance. Accordingly, this Gaussian mixture model requires 3×N×K channels for the output of the auxiliary autoencoder, where N represents the number of filters and K is the number of Gaussian distributions considered.

The study conducted in [4] showed that varying the number of mixtures used in the model resulted in minimal gains, with only a 0.2dB difference observed between using one or three Gaussian mixtures. Given these results, and the fact that more complex approaches did not significantly improve performance, a simpler entropy model using only one Gaussian mixture was implemented to keep the model as straightforward as possible. To improve the entropy coding efficiency, an autoregressive model, C_m, is used to predict each latent representation from its causal context [37]. By concatenating the output of the autoregressive model (C_m) and the output of the synthesis transform (h_s) the estimated probability distribution of $\hat{y}$ is obtained after convolutional layers (CL) and given to the entropy encoder and decoder. The bitrate of compressed images is given by $R = R(\hat{y}) + R(\hat{z})$, where the last term is side information, obtained by encoding the entropy model parameters required for arithmetic decoding.

3.2 DATASETS

Five datasets were adopted to evaluate the proposed compression approach using five different object classes: faces, people, cars, motorbikes and bikes. The faces dataset was created by joining the LFW Face Database and Flickr-Faces-HQ Dataset available respectively in [38] and [39]. The people, cars, motorbikes and bikes datasets were created using tools made available in Detectron2 library [40], by cropping and resizing the bounding boxes of people, cars, motorbikes and bikes in different positions, taken from several videos available online.

The concept under study is that better compression performance can be achieved if we training our autoencoders using only features that represent a specific object.

To this aim, for each class four different datasets were considered representing the object with a black background, the object with blurred background, the object with the original background and the object in the centre of the image with a black background. In the latter case, the number of the specific object in the image was limited to one, while for the other cases it was allowed images with more than one object. Figures 58, 57, 56 and 59 (Appendix A) show some examples of image datasets for each considered class using different background approaches. Then, in Figure 60 shows some images of the faces dataset with the original and with a black background.

The datasets with objects of class people, cars, motorbikes and bikes have a total of 94500 images, resized to 128×128 pixels, while for class faces the dataset has a total of 78761 images, also resized to 128×128 pixels. Object detection and segmentation were performed by using the following models: faster_rcnn_R_101_FPN_3x and mask_rcnn_R_50_FPN_3x, respectively. Specific details about these models and their implementation can be found in [40].

3.3 EXPERIMENTAL RESULTS

The performance evaluation study was carried out through simulation of the proposed pipeline by measuring the compression efficiency using five relevant visual objects in surveillance applications: people, cars, motorbikes, bikes and faces, as described in the previous section. Learning-based compression was implemented using the autoencoder architecture presented in Section 3.1. The software implementation is available in the CompressAI framework [41].

The pre-trained implementations available in CompressAI were first validated by confirming that the results presented in [41] could be accurately reproduced. These provided pre-trained models were trained for 4-5M steps on 256×256 image patches, randomly extracted and cropped from the Vimeo-90K dataset [42]. A batch size of 16 was used and the initial learning rate was 1e-4, for approximately 1-2M steps. The learning rate was then divided by 2 whenever the evaluation loss reached a plateau (patience of 20 epochs).

For the performance evaluation of the proposed approach, two versions of the learning network were used: (i) the pre-trained and (ii) the re-trained one, obtained through transfer learning over the pre-trained models. There are two different types of transfer learning over a pre-trained model: fine-tuning, where all of the model's parameters for a new task are updated, and feature extraction, where only the

final layer weights are updated. In this process, the parameters of the pre-trained models were first loaded and then fine-tuning was carried out by further learning the specific features of each visual object class.

For transfer learning the people, cars, motorbikes and bikes datasets were divided into 90000 images for training and 4500 for testing, while the faces dataset was divided into 74822 images for training and 3939 for testing. The models were trained for 1M steps with a batch size of 8, and an initial learning rate of 1e-4. The learning rate is divided by 10 whenever the evaluation loss reaches a plateau (patience of 10). The training of the models is stopped if the loss does not decrease in the following 24 epochs. The loss function used for training is formulated as

$$L = \lambda \times 255^2 \times D_{MSE} + R \tag{8}$$

where λ controls the rate-distortion tradeoff, D_{MSE} denotes the distortion term between the original and decoded images (Mean Squared Error in the case), and R is the estimated bit-rate. When optimized by mean square error (MSE), λ was chosen from the set {0.00008,0.0009,0.0018,0.013,0.0483,0.8}. The number of filters (N) was 128 for the five lower rates and 192 for the higher rate models.

These two learning-based compression networks used in the experiments are identified in Table 1, including the notation used in the Figures ahead. For comparison, the same visual objects were also compressed as intra-coded images with four standard encoders: joint photographic experts group (JPEG), JPEG2000, high efficiency video coding (HEVC) (HM) and versatile video coding (VVC) (VTM).

Table 1: Learning-based image compression models used in the visual objects

Model	**Description**
pretrained-<class>	pre-trained model: pretrained model using “class” for testing.
transfer-learn-<class>	transfer learning: retrained model using “class” for training and testing

3.3.1 *Rate-distortion performance*

The rate-distortion performance was evaluated by measuring three different quality metrics: PSNR, MSSIM and video multimethod assessment fusion (VMAF), against the corresponding bits per pixel (bpp) achieved after compression.

As mentioned before, it is possible to transfer learning over a pre-trained model using two different approaches, fine-tuning and feature extraction. Figures 16, 64,

and 65 (the last two in B) show the PSNR results for faces dataset with blurred background, original background and with a black background using different transfer learning approaches, respectively. The number present in the legend refers to the fraction of the frozen parameters in the encoder of the considered autoencoder. The blue line represents the fine-tuning process where the pre-trained model is retrained, and all weights updated, and the red line represents the case of feature extraction frozen the first 85% parameters of the pre-trained model (it only updated the last 15% of the weights of the encoding module). The comparison between various datasets and percentages of frozen weights shows that models which underwent fine-tuning achieved better performances than those using feature extraction only. This improvement can be observed through the higher values of PSNR achieved for the same value of bpp. Therefore, it was decided to use fine-tuning approach to transfer learning of the considered pretrained models.

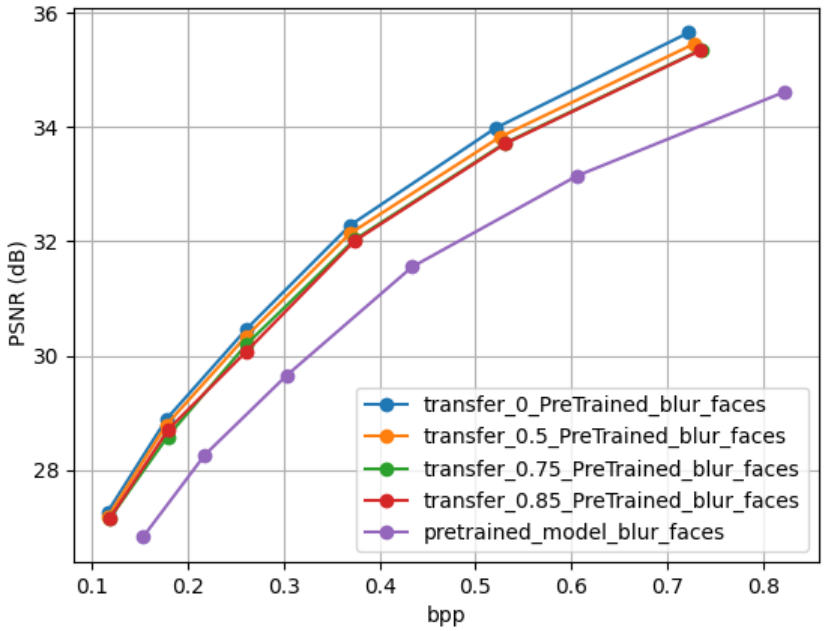

Figure 16: PSNR - faces dataset with blurred background using different transfer learning approaches.

The PSNR results for people, cars, motorbikes and bikes datasets, considering different approaches to the background, are shown in Figures 17, 18, 19 and 20, respectively. The blue line represents the model trained using dataset with the original background, the orange line represents the model trained using dataset with a black background, the green line represents the model trained using dataset with blurred background and the red line represents the model trained using the dataset with one object in each image center with a black background.

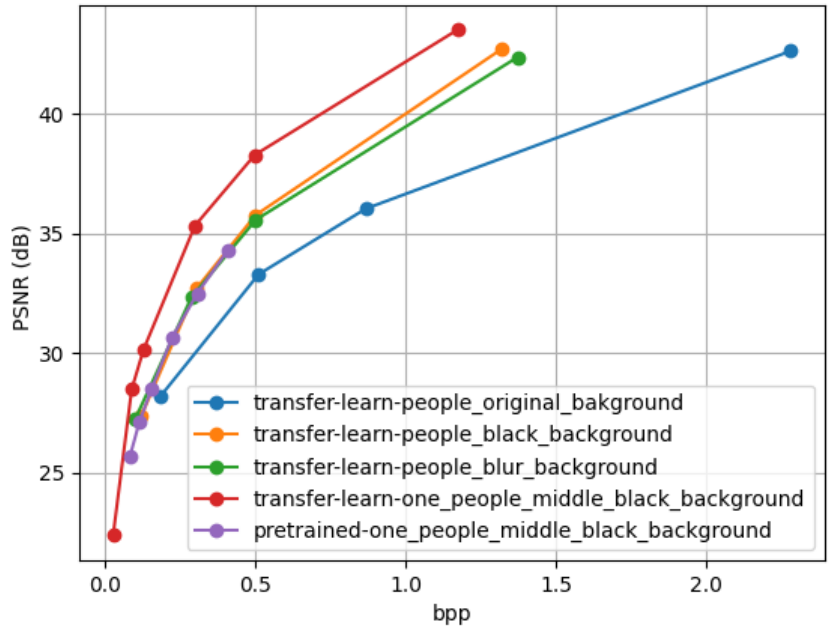

Figure 17: PSNR - people dataset with different approaches to the background.

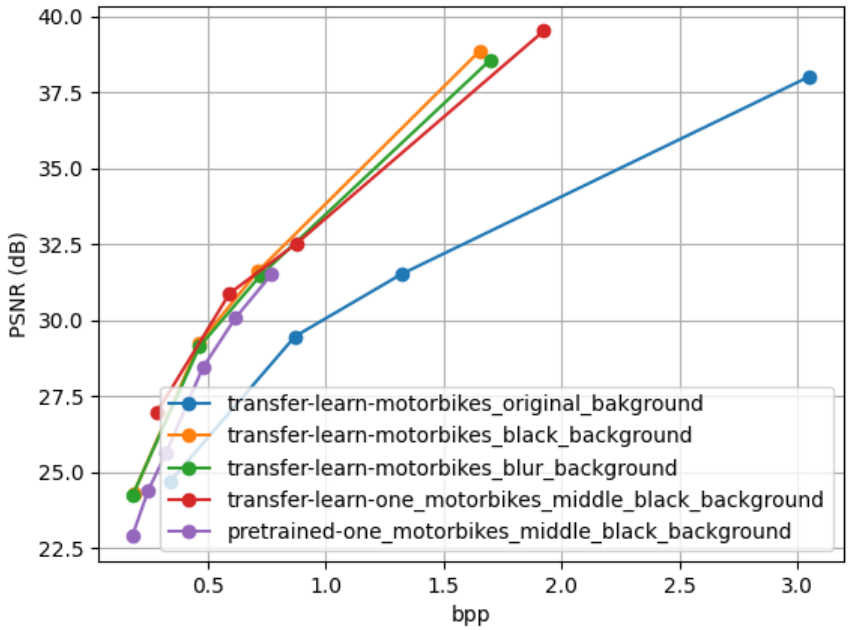

Figure 19: PSNR - motorbikes dataset with different approaches to the background.

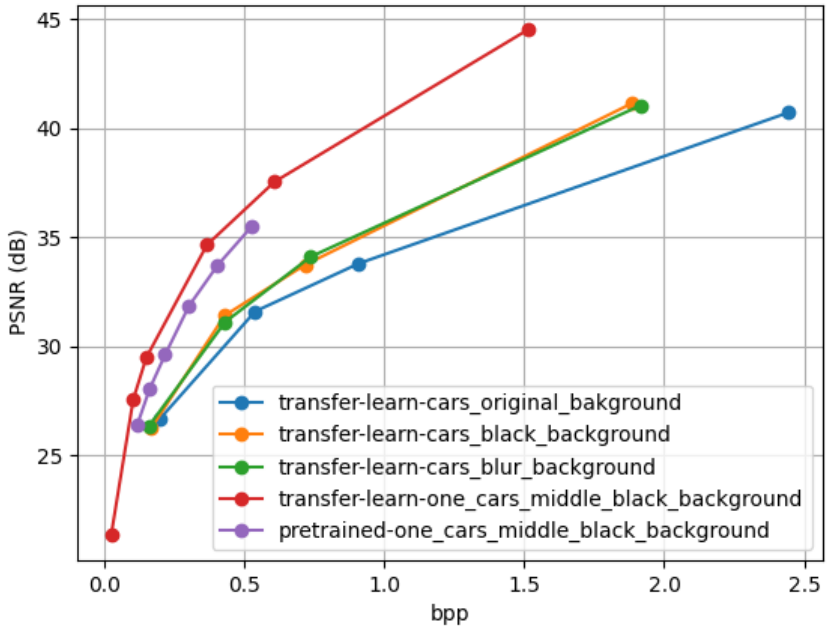

Figure 18: PSNR - cars dataset with different approaches to the background.

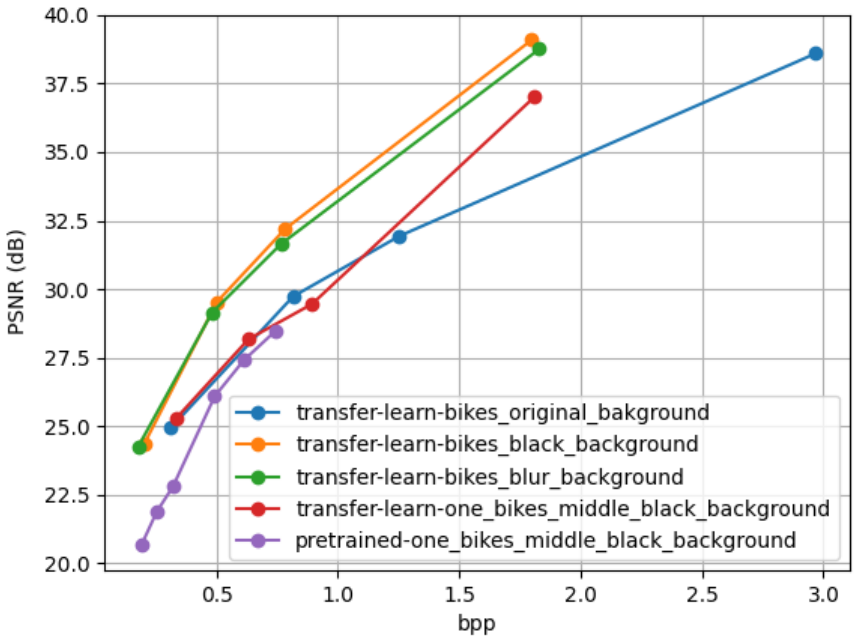

Figure 20: PSNR - bikes dataset with different approaches to the background.

Looking at the blue, green and orange line it is possible to infer that the datasets with blurred images and images with a black background are those where the best

performance is obtained. This can be explained on the grounds that, in their images, the objects are the in spotlight and so the model can better learn the features that characterize the considered class. It can also be observed, especially in the cars and people datasets and basically for the same reasons, the high performances with images containing only one centered object with a black background.

A different behavior is observed with the motorbikes and bikes datasets. As it can be seen, the best performances obtained are not with images containing only one object centered in the image with a black background. This is most likely due to the very limited segmentation performance of Detectron2, which often led to incomplete objects, as shown in for example in Figure 59.

Based on the results obtained so far, the next set of experiments involved images with a black background and one object centered in the image, and the object classes people, cars and faces.

As mentioned in 3.1, the attention module proposed in [4] was removed of the autoencoder architecture because the images present in datasets contain a black background. To validate this option, an experiment was conducted in order to compare the performance of the autoencoder with and without the attention module. The PSNR results considering different approaches are shown in Figures 21, 22 and 23 for people, cars and faces datasets, respectively. The blue and green line represent the results obtained with (*transfer-learning*) in each class and the pretrained model respectively, both without the attention model. The orange line and the red line are the corresponding results obtained now with the attention module.

On the whole, the results show that there are no significant performance differences for the configurations with and without the attention module. Therefore, with no improvements observed and for the sake of simplicity, it was decided to not use attention modules in the architecture of the autoencoders.

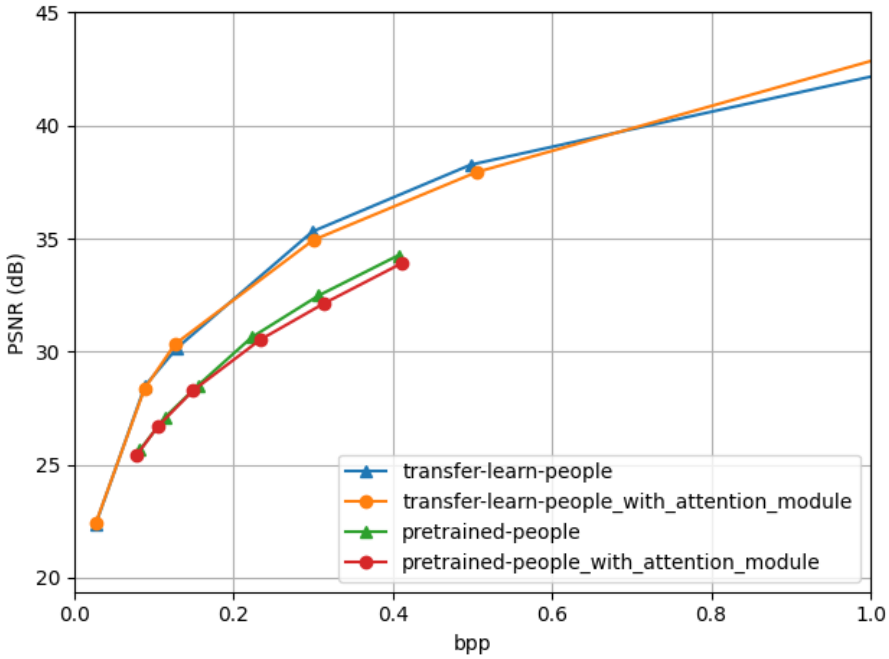

Figure 21: PSNR - people dataset with different architecture approaches.

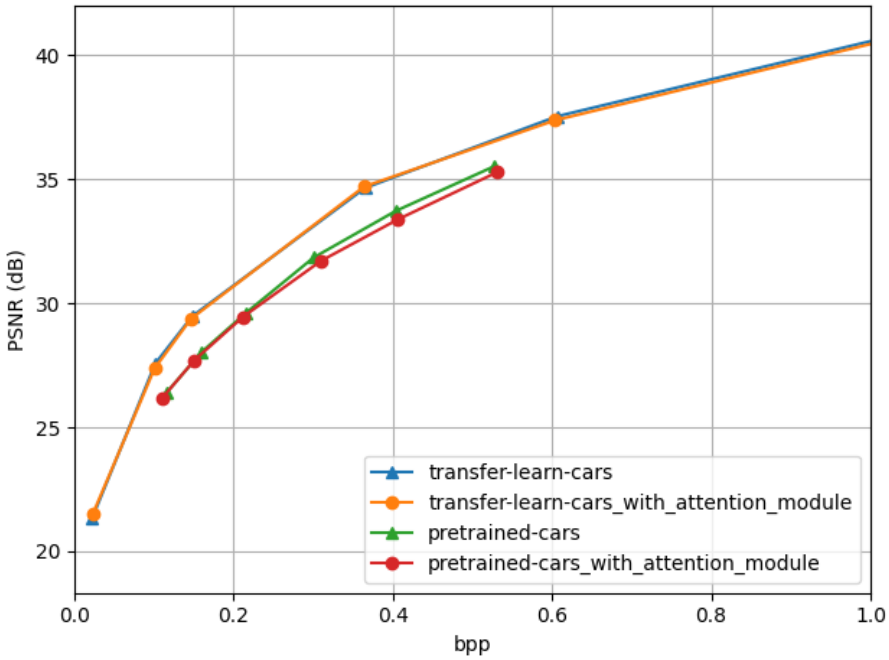

Figure 22: PSNR - cars dataset with different architecture approaches.

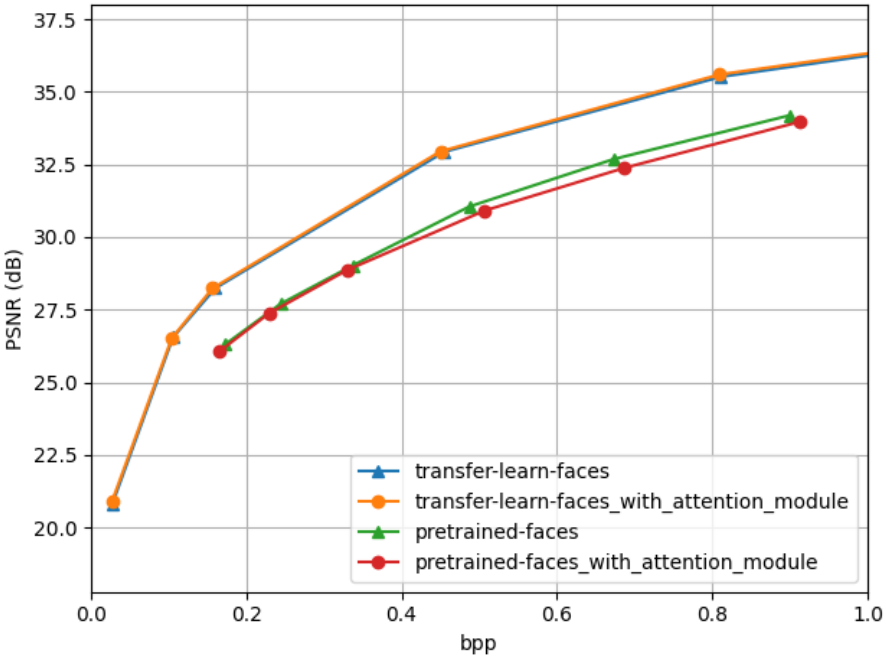

Figure 23: PSNR - faces dataset with different architecture approaches.

With the datasets and the models' architecture defined, the next study aimed at comparing the performace obtained against standard encoders. The PSNR results for the people, cars and faces datasets are shown in Figures 24, 25 and 26, respectively. The coding efficiency of the standard encoders is more or less inline with their expected relative performance: where JPEG (the oldest of them) with the lowest R-D performance and VVC (the most recent) the highest.

In regard to the proposed learning based approaches, for the people and cars dataset, the fine-tuned autoencoder (*transfer-learning*) outperforms the VVC, as well all other encoders up to 0.6bpp. With the faces dataset the result is even more impressive as transfer-learning exhibts consistently higher R-D performance in allover teh considered range. In the case of agnostic learning using the pre-trained model, (i.e.no fine-tuning), the R-D performance is comparable to that of VVC, but still slightly above for almost the bit rate range.

The BD-RATE(%) and BD-PSNR(dB) gains, using the VVC as reference, for the proposed learning-based model (*transfer-learning*) are present in Table 2. The BD-RATE quantifies the bitrate savings achieved at an equivalent quality level, whereas the BD-PSNR measures the quality improvements obtained considering an equivalent bitrate. As it can be seen, the coding gains are quite significant for the considered datasets. For the people dataset, the BD-RATE gain is 32.6% for a BD-PSNR of 2.8%, and for the cars dataset, 28.2% and 2.1%, respectively. This is

even higher for the faces dataset, i.e. 46.7% and 3.06% for BD-RATE and BD-PSNR, respectively.

Table 2: BD-Rate and BD-PSNR using VVC as reference.

Dataset	**Model**	BD-RATE(%)	BD-PSNR(dB)
People	transfer-learn	-32,64	2,84
Cars	transfer-learn	-28,16	2,07
Faces	transfer-learn	-46,71	3,06

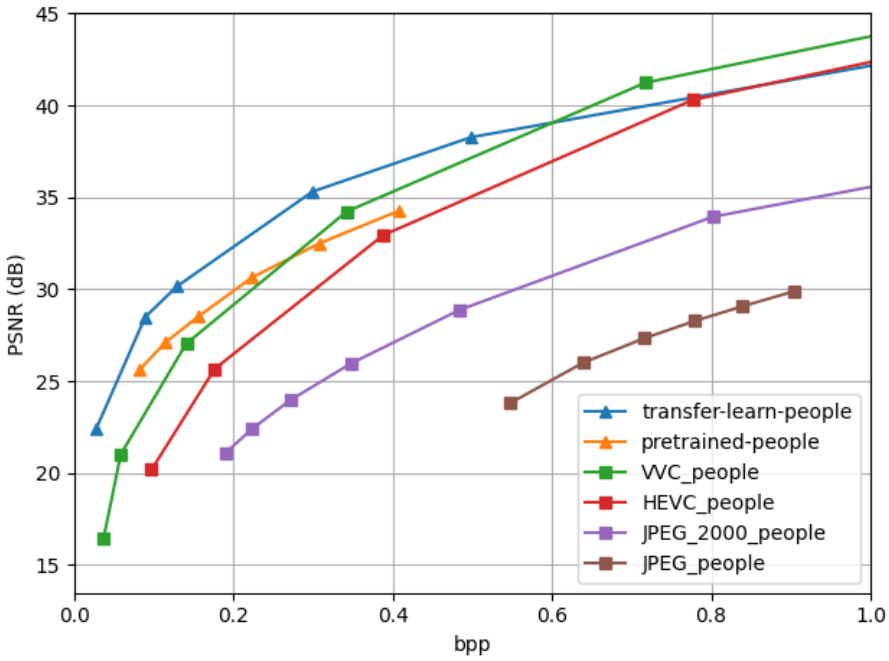

Figure 24: PSNR - people dataset.

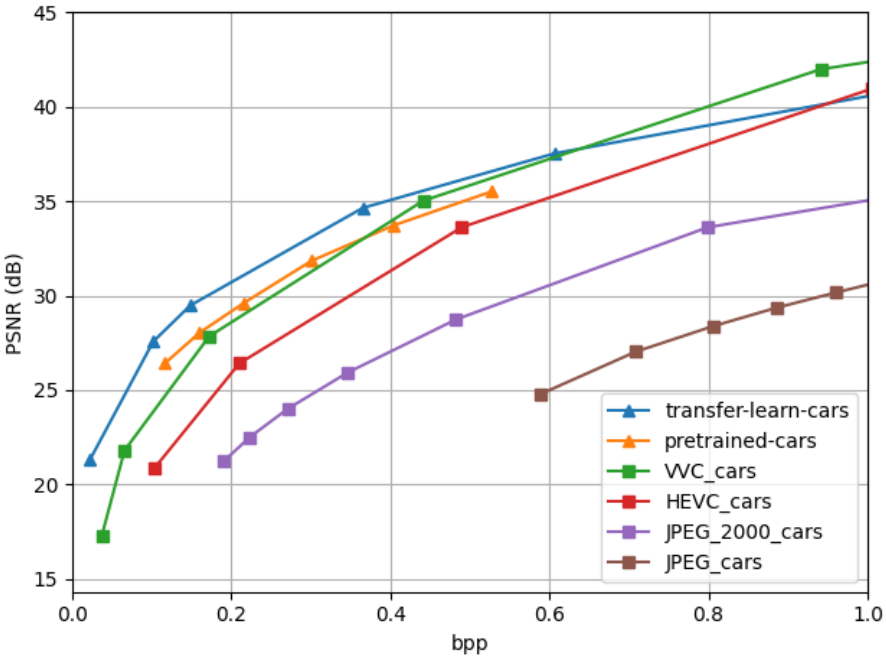

Figure 25: PSNR - cars dataset.

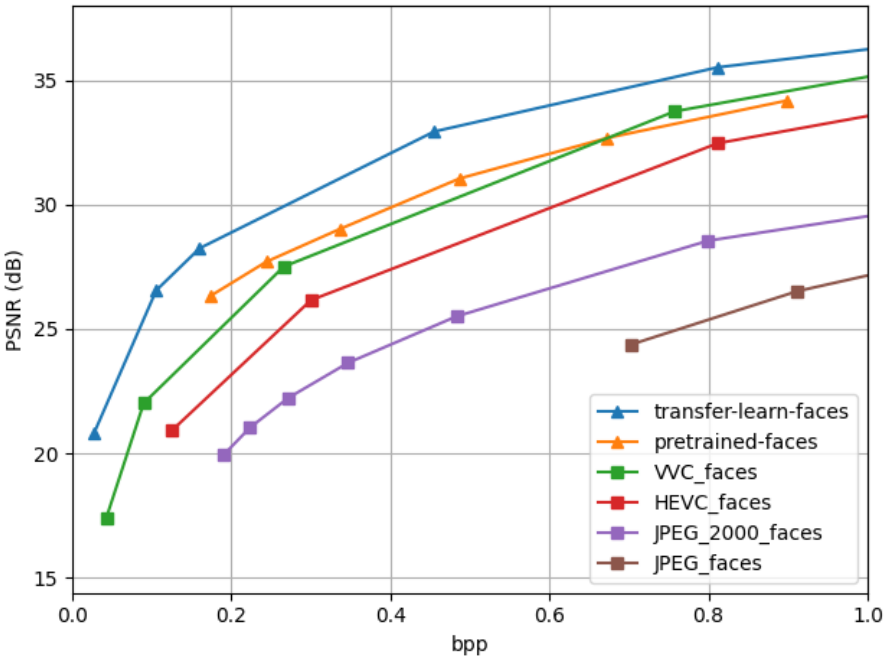

Figure 26: PSNR - faces dataset.

In addition to the PSNR results, the algorithm performance has also been evaluated in terms of MS-SSIM and VMAF [43]. The results are presented in Figures 66, 67 68, 69, 70 and 71, with the original MS-SSIM results are converted

to a dB scale ($-10\log_{10}(1 - MS-SSIM)$) in order to represent the differences more clearly. These results confirm that the proposed approach achieves better performance than the VVC, as well as the benefits of fine-tuning the parameters learnt from generic images. Overall, for the same bit rates, the quality obtained from the *transfer-learn* model is consistently higher than the state of the art VVC encoder.

3.3.2 *Task-driven algorithm performance*

The task-driven algorithm performance was also evaluated in two types of assignments: object classification and face recognition. The same range of coding rates was used, i.e., up to 1.0 bpp and the tasks were carried out using algorithms available in Detectron2 library.

3.3.2.1 *Object classification*

The classification accuracy of the visual objects was assessed before and after coding the images with the different schemes. Since the actual object class is known in advance, the result of this task is a binary output, indicating whether the correct class was identified or not. This is the class with the highest confidence score, provided it is at least 70%, positively identified or not. The same data test was used in all encoders, and the accuracy, measured as the percentage of objects correctly classified, and the results are presented in Figures 27, 28 and 29.

The data obtained shows that the autoencoder with *transfer-learning* outperforms all the others, particularly at lower bit rates. In the case of faces, the accuracy is quite high for very low bit rates. This is likely to happen because the shape of all faces are quite similar and the object detection algorithm is capable of identifying a face object from its shape, regardless the quality of the corresponding features. In the case of people and cars dataset this is not likely to happen because the diversity of shapes is much higher.

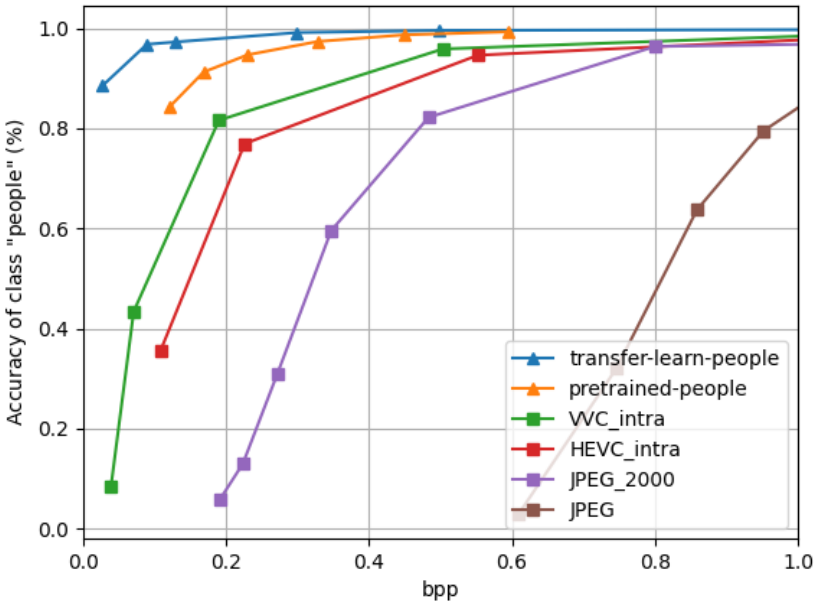

Figure 27: Classification accuracy - people dataset

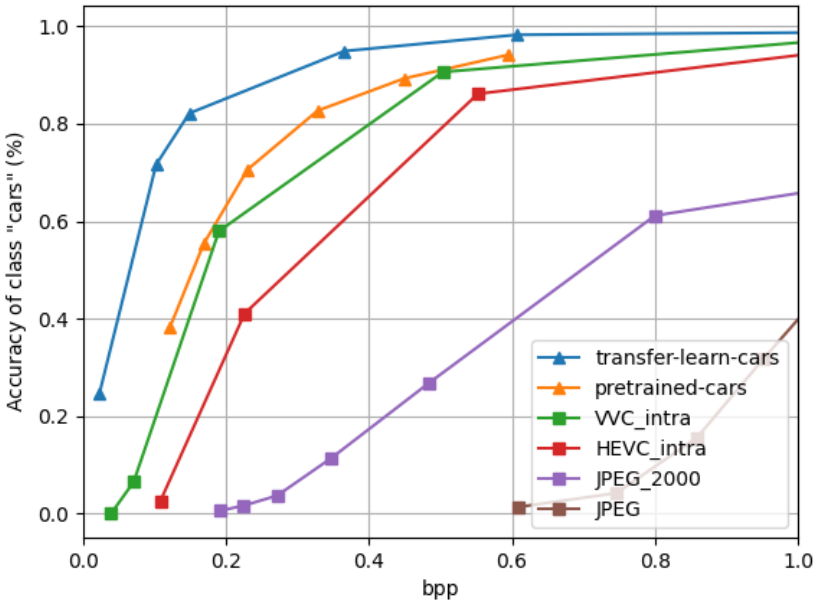

Figure 28: Classification accuracy - cars dataset

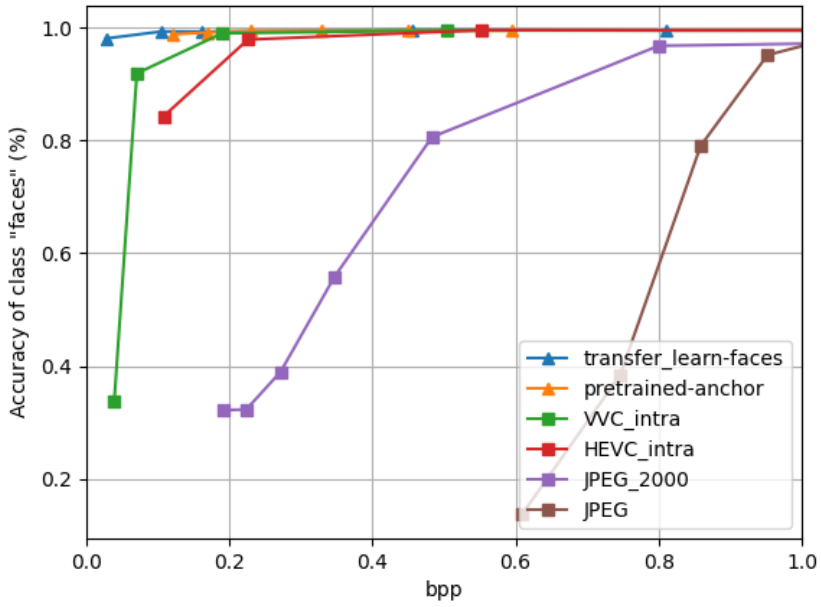

Figure 29: Classification accuracy - faces dataset

3.3.2.2 *Face recognition*

Face recognition was performed by using the Deepface tool created by Facebook [44]. To measure the performance of face recognition after compression with different encoders, a dataset with 2304 face images was used. Then a test group with 1000 matching faces in the database and 1000 non-matching faces was created. For a face to be correctly recognised, it was established that at least one, and no more than five, matching images should be found. The results obtained are presented in Figure 30, showing that the accuracy obtained from images encoded with the proposed autoencoder (*transfer-learning*) is again consistently higher than all other encoders (including the learning-based scheme based on pre-training with generic images).

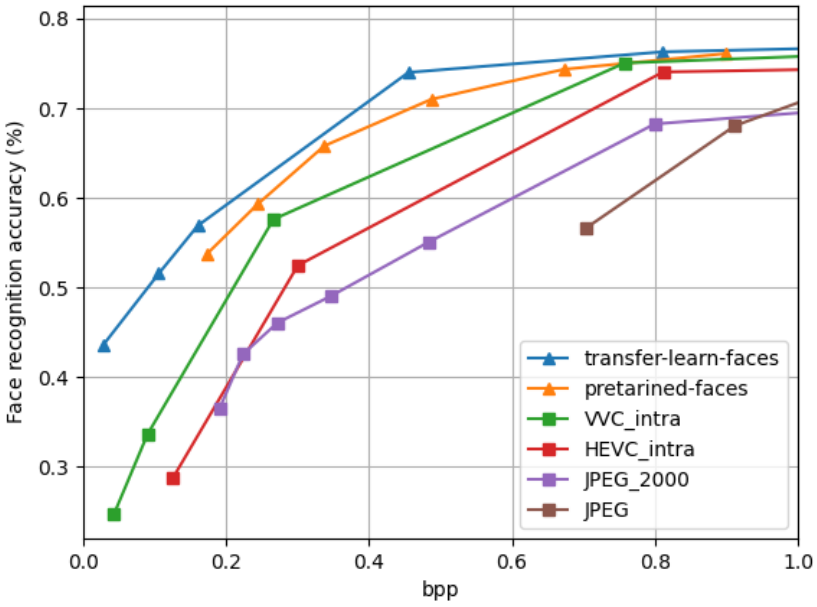

Figure 30: Accuracy in face recognition

3.4 SUMMARY

In this chapter, a novel coding approach based on multiple autoencoders was described, each one specifically optimised for a particular object class. The proposed encoding scheme consistently achieves better results than other standard encoders, including the state of the art VVC.

Initially, the learning-based network architecture and the datasets used to evaluate the proposed method were presented. The first experiment consisted on evaluating the impact of the background surrounding each object, considering different approaches. It led to the conclusion that resulted that removing the background from the objects in images conducted to better compression performance. The best results are obtained when the images of the datasets consider only show one object center with a black background.

It was also shown that the use of the attention module in the autoencoder architecture does not bring significant improvements when the objects of the datasets are already segmented. This is justified because the network can easily learn the relevant features of the considered objects when the images contain only the respective object.

Additionally, it was demonstrated that fine-tuning of generic optimised autoencoders through transfer learning, yields improved the compression efficiency and the task-driven algorithm performance (object classification and face recognition).

4

LEARNING-BASED COMPRESSION USING CLASSIFICATION FOR 360° AND UHD IMAGES

Based on the approach of using several autoencoders to compress different parts of an image, this chapter addresses an efficient method to compress 360° and UHD images using a set of autoencoders in order to take advantage of the characteristics of different regions of the images. Firstly, the proposed approach is detailed, and then the datasets used to training the autoencoders, the training parameters and the methods to classify image regions are described. Finally, the experimental results are analysed by measuring different rate-distortion metrics used to compress 360° and UHD images.

4.1 PROPOSED APPROACH

The main purpose of this approach is to compress 360° and UHD images more efficiently. To this end, several autoencoders are trained to tackle the different kinds of pixel-based or transform-based features like variance, horizontal/vertical gradient, DCT, Karhunen-Loève Theorem (KLT) and Fourier transform. Thus, instead of using a specific autoencoder to compress a particular class of objects, as proposed in Section 3, herein specific autoencoders are trained to compress regions of UHD and 360° images with particular properties (e.g. variances or gradients). For certain properties several autoencoders will be trained, considering different ranges (e.g. one autoencoder compresses regions with high variance and another compresses regions with low variance).

Each image to compress is divided into square regions of uniform size, as shown in Figure 31, wherein WR stands for width and HR for height. WR and HR are predefined sets of parameters, such that an integer number of regions is defined within the whole image. The visual content of these square regions is agnostic regarding their semantic meaning.

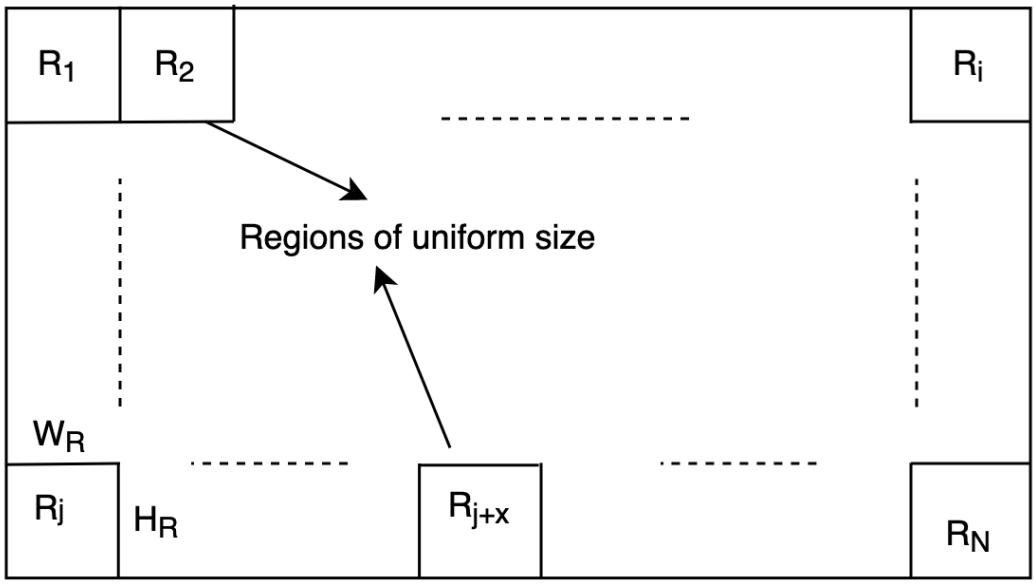

Figure 31: Division of 360° and UHD images in square regions.

4.2 DATASETS

To implement this approach one general dataset was created, using 438980 crops with 128x128 pixels from 360° and UHD images. First, several video sequences were joined, such as UVG Dataset [45], SJTU Dataset [46], MCML Dataset [47], NETFLIX - "Chimera" video sequence [48], Xiph.org test media short film Sintel [49] and JVET Dataset [50], with frames interval of approximately 1.5 seconds.

As mentioned before, the main purpose of this method is to compress UHD and 360° images using different autoencoders based on region-based properties of the image such as the variance, the gradients and the principal component analysis (PCA) over features extracted by CNNs. To achieve this, the original dataset are split considering each metric into four different subsets. In this way, it was created three copies of the general dataset divided each one in four various subset consider different ranges of each metric used, as explained in the following subsections.

4.2.1 *Variance*

In order to create 4 different subsets, each with a different variance range, the variance of all crops has been calculated after converting the crop to grayscale, according to

$$Variance = \frac{\sum_{n=1}^{128}\sum_{m=1}^{128}(X_{nm} - \mu)^2}{128 * 128} \tag{9}$$

where μ is the average of the crop in grayscale, and n and m are the coordinates of the pixel. A histogram of the variances for all crops in the dataset is shown in Figure 32. Based on this information, 4 subsets were created, each comprising the same number of crops. The ranges of variances of each group can be inferred with the aid of the dashed red lines in the figure.

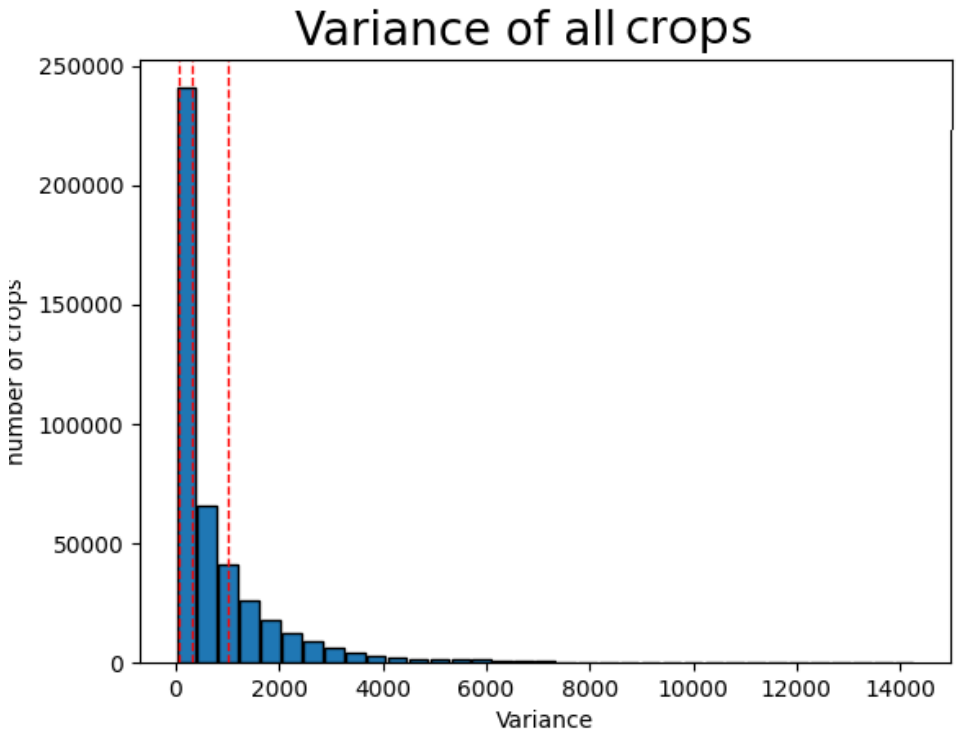

Figure 32: Histogram of variances for all crops.

In detail, the first subset of crops includes crops with low variance, in a range between 0 and 53.76. The second subset is composed by crops with a variance between 53.76 and 315.89. In the third subset crops have a variance between 315.89 and 1014.71, and the fourth subset contains crops with variance between 1014.71 and 14281.55. The total number of crops in these subsets is 109745. Some examples of crops representing different values of variance are shown in Figure 61 (Appendix A).

4.2.2 *Gradients*

Another criterion used to group crops was the gradient, namely the average horizontal and vertical gradient. To perform these operations, the following equations based on Sobel kernels were used,

$$Horizontal\_Gradient = \frac{\sum_{n=1}^{128}\sum_{m=1}^{128}(|Gx_{nm}|)}{128 * 128} \tag{10}$$

$$Vertical\_Gradient = \frac{\sum_{n=1}^{128}\sum_{m=1}^{128}(|Gy_{nm}|)}{128 * 128} \tag{11}$$

where n and m are the coordinates of the pixel, Gx_{nm} and Gy_{nm} are crop horizontal and vertical gradient magnitudes, respectively. The crops were converted to grayscale for the calculation of the gradients. These results, for all crops, are plotted in Figure 33, distributed by horizontal and vertical gradients. Each point represents one crop.

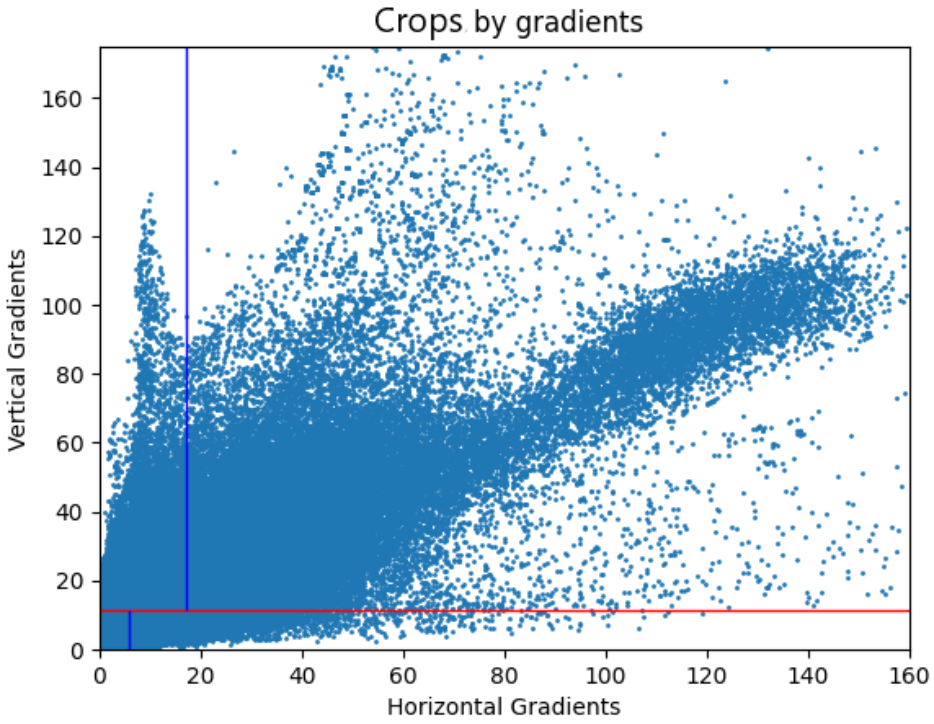

Figure 33: Distributions of the crops by gradients.

As before, 4 subsets with the same number of crops. Firstly, the dataset is halved into two groups according to their vertical gradient (red line). Then, using each of these groups is further split in two based on their horizontal gradients (blue lines). Thus, 4 subsets were created with the same number of crops, for different ranges of horizontal and vertical gradients. The first subset includes crops with low gradients, the horizontal gradients between 0.0 and 5.95 and vertical gradients between 0.0 and 11.65. The second subset have crops with the horizontal gradient between 0.0 and 13.62 and vertical gradients between 11.65 and 252.39. The third subset comprise crops with the horizontal gradient between 5.95 and 119.19 and the vertical gradients between 0.0 and 11.65. The last subset group crops with high gradients, the horizontal gradients between 13.615 and 206.403 and the vertical gradients between 11.65 and 252.39. The total number of crops in the four subsets is 109745. Some examples of crops representing different values of horizontal and vertical gradients are shown in Figure 62.

4.2.3 *Principal component analysis (PCA) of features*

A strategy different from the two previous ones, which were based on image metrics, is to create the 4 image subsets based on the visual context (features) of the crops. As shown in Figure 34, first a pre-trained model (in this case VVG16 [51]) is used to extract features from the crop, then a PCA is applied to reduce the number of variables used to split the subsets. Finally, the K-means algorithm is used to cluster the crops, grouping them in subsets.

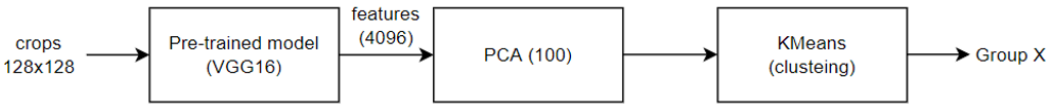

Figure 34: Proposed method to split crops using PCA.

Prior to load the VGG16 model it is necessary to remove the output layer because, in its standard configuration, the neural network is used to classify objects. In this case only the crop features are needed, so the output of this block is a vector with 4096 features per crop.

PCA is a statistical technique for reducing the dimensionality of a dataset. PCA combines the input data into a set of independent components, by a linear transformation, each representing a combination of the original inputs, in this case, the output features of VGG16. In this way is computed the principal components

and using them to perform a change of basis on the data. The most important features are the ones that best relate to the input features, which relationship can be observed using a covariance matrix. The covariance matrix presents the estimation of how every "new" feature relates to input features. In this case, the first 100 most important features are considered. The output of PCA is used to create 4 subsets, similar to what was done using the variance and gradients of the crops. To group the crops in subsets the K-means algorithm is used. The number of clusters is defined as 4 to create 4 subsets. The K-means algorithm identifies 4 centroids, and allocates every crop to the nearest cluster while keeping them as small as possible. For the first iteration, it selects the centroids randomly, and for the following iterations, the centroids' positions are optimized. When the positions of the centroids are stable the optimal centroids were found. In this case all subsets have a different number of crops, that is 116517, 99079, 6886 and 216499 crops. Some examples of crops representing different clustered crops using the PCA are shown in in Figure 63.

4.3 EXPERIMENTAL RESULTS

The performance of the proposed approach was tested by using UHD and 360° images. The datasets were described in the previous section and, as in Section 3.3, it was used the autoencoder architecture presented in Section 3.1.

For the performance evaluation of this approach three versions of the learning network were used: (i) the pre-trained, (ii) the re-trained one, obtained through transfer learning over the pre-trained models, updated all weights, and (iii) the model trained from the scratch. As in Section 3.3, the parameters of the pre-trained models were primarly loaded and then fine-tuned, by further learning the specific dataset.

The number of crops per subsets for transfer learning and the model trained from the scratch are presented in Table 3.

All models were trained with a batch size 8 and an initial learning rate of 1e-4. The learning rate is divided by 10 whenever the evaluation loss reaches a plateau (patience of 10). The *transfer-learn* models were trained with 1.3M steps using the subsets divided by variance, gradients, and using PCA group_0 and group_1. The subsets using PCA group_3 and group_4 were trained with 0.6M and 1.7M steps, respectively. The training of *transfer-learn* models are stopped if the loss do not decrease in 40 following epochs.

Table 3: Number of crops per subsets, for the different methods used to split the data

Method	Train/Test	Subset	Number of crops
Variance	Train	very low	104260
		low	
		high	
		very high	
	Test	very low	5485
		low	
		high	
		very high	
Gradient	Train	low horizontal and vertical	104260
		low horizontal and high vertical	
		high horizontal and low vertical	
		high horizontal and vertical	
	Test	low horizontal and vertical	5485
		low horizontal and high vertical	
		high horizontal and low vertical	
		high horizontal and vertical	
PCA	Train	group_0	110691
		group_1	94125
		group_2	6542
		group_3	205674
	Test	group_0	5826
		group_1	4954
		group_2	344
		group_3	10825

The models trained from the scratch were trained only with the subsets created using the variance and the gradient, as the metric to divide the crops. These models were trained for 5.6M steps and the training were stopped if the loss do not decrease in 50 following epochs. The loss function used in the training is formulated as 8.

The values of λ and N are the same as those referred in Section 3.3

The three learning-based compression algorithms used in the experiments are identified in Table 4, including the notation used in the figures ahead. For comparison, the same visual objects were also compressed as intra-coded images with four standard traditional encoders: JPEG, HEVC (HM) and VVC (VTM).

Table 4: Learning-based image compression models used in the UHD and 360° images

Model	Description
generic_autoencoder_compressAI	pretrained model: pretrained model available in compressAI
transfer-learn_<subset>	transfer learning: retrained model using "subset" for training and testing (CTUs_v1-subsets splited using variance, CTUs_v2-subsets splited using gradients and CTUs_v3-subsets splited using cluster PCA)
model_by_zero_<subset>	model by the zero : model trained by the scratch using "subset" for training and testing (CTUs_v1-subsets splited using variance, CTUs_v2-subsets splited using gradients and CTUs_v3-subsets splited using cluster PCA)

4.3.1 *Dataset Separated by Variance*

The PSNR results using the subsets separated using different intervals of variance are shown in Figures 35, 36, 37 and 38, representing the group with variance very low, low, high and very high, respectively. The blue line represents the model trained from the scratch, the orange line the model using *transfer-learn* over a specific subset, and the green line the pretrained model available in compressAI.

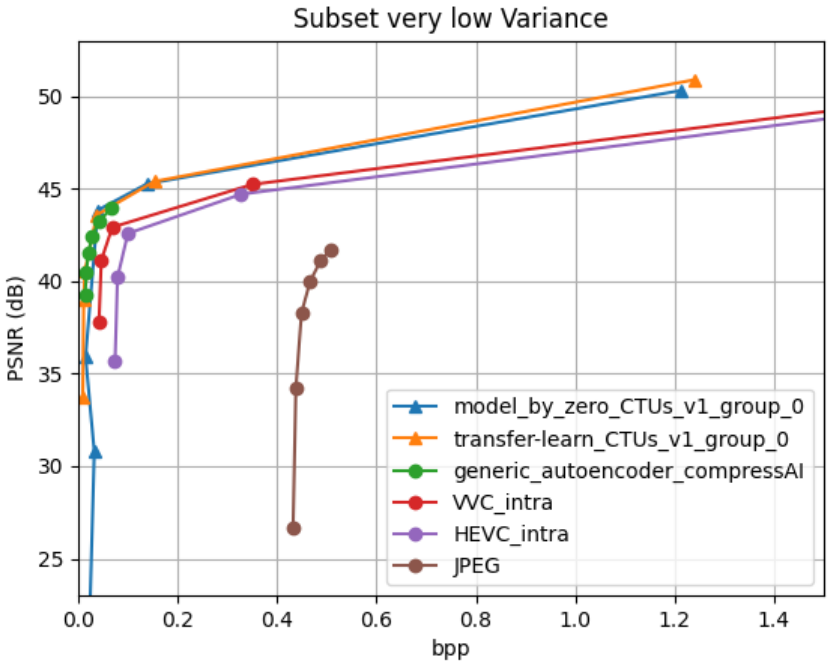

Figure 35: PSNR - Subset split variance very low.

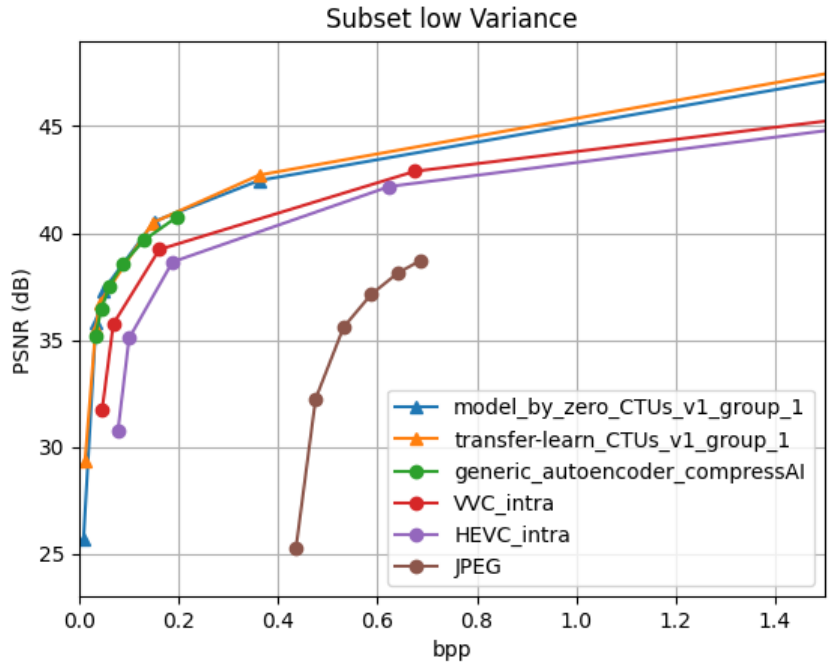

Figure 36: PSNR - Subset split variance low.

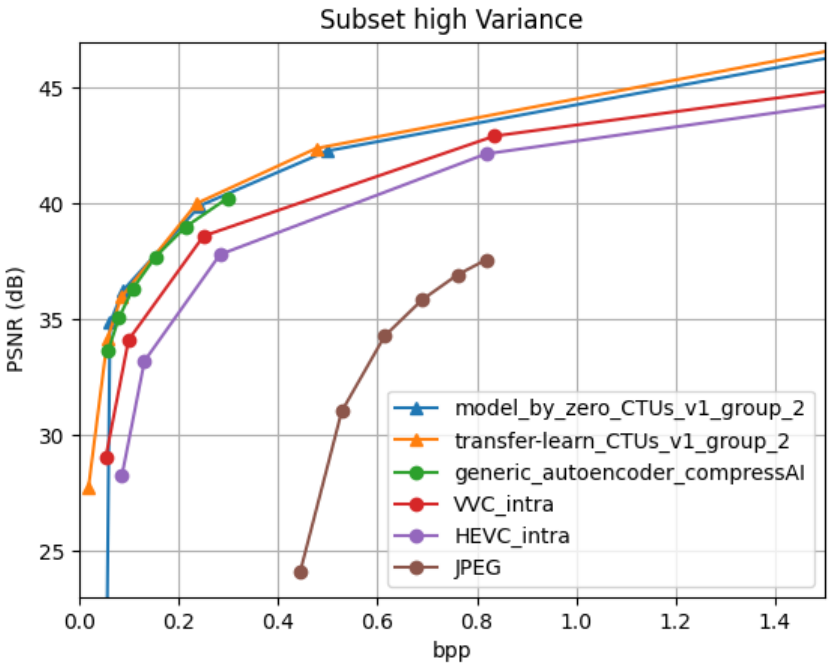

Figure 37: PSNR - Subset split variance high.

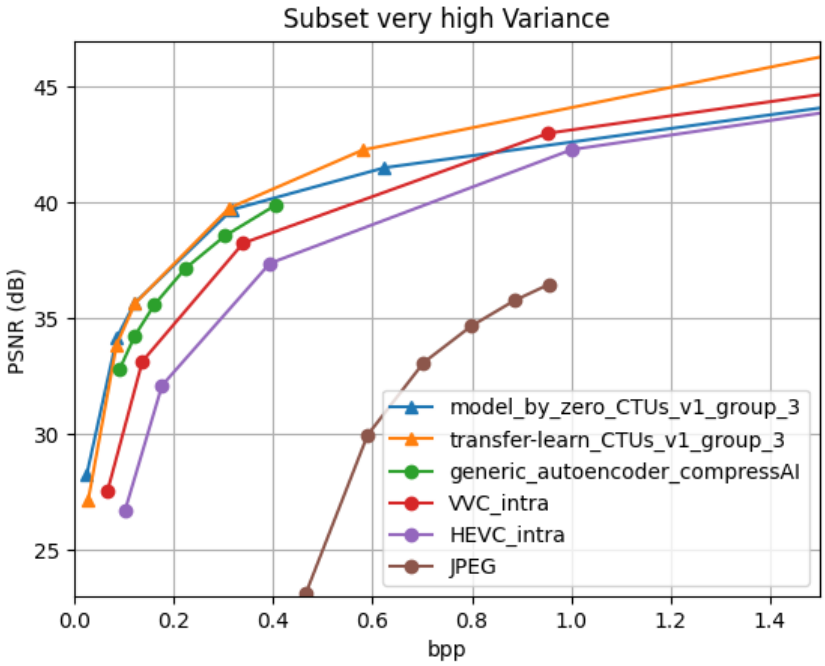

Figure 38: PSNR - Subset split variance very high.

Analysing the curves in these figures, on whole a better performance is obtained when using the models retrained rather than the generic autoencoder. This is more obvious for the subsets with a higher variance. The performance of the model trained from the scratch and the model using the *transfer-learn* is quite similar for all subsets, excluding the graph considering crops with a very high variance where it is noticeable a worse performance of the autoencoder from scratch possibly caused by overfitting. In regard to the learning based approaches, for all subsets using the variance, the fine-tuned autoencoder (*transfer-learn*) outperforms the VVC and all other encoders Only for subsets with very high variance the PSNR quality of the proposed encoder is slightly below than that of the VVC, considering a bpp higher than 0.9.

4.3.2 *Dataset Separated by Gradients*

The PSNR results using the subsets grouped for different intervals of horizontal and vertical gradients are shown in Figures 39, 41, 40 and 42, representing the group with low horizontal and vertical gradient, low horizontal and high vertical gradient, high horizontal and low vertical gradient, and high horizontal and vertical gradient,

respectively. The graph labels maintain a consistent representation as in the results presented in the previous section.

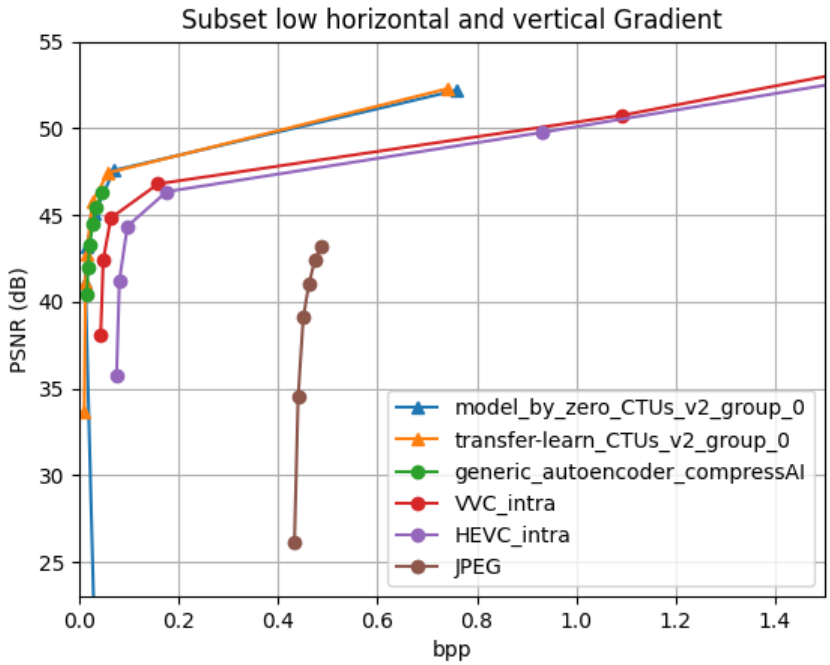

Figure 39: PSNR - Subset low horizontal and vertical Gradient.

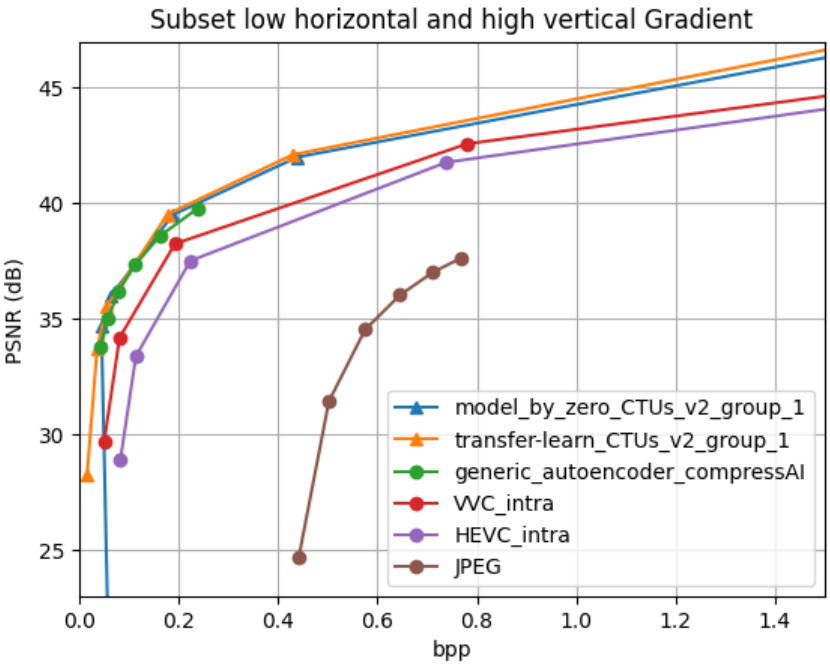

Figure 40: PSNR - Subset low horizontal and high vertical Gradient.

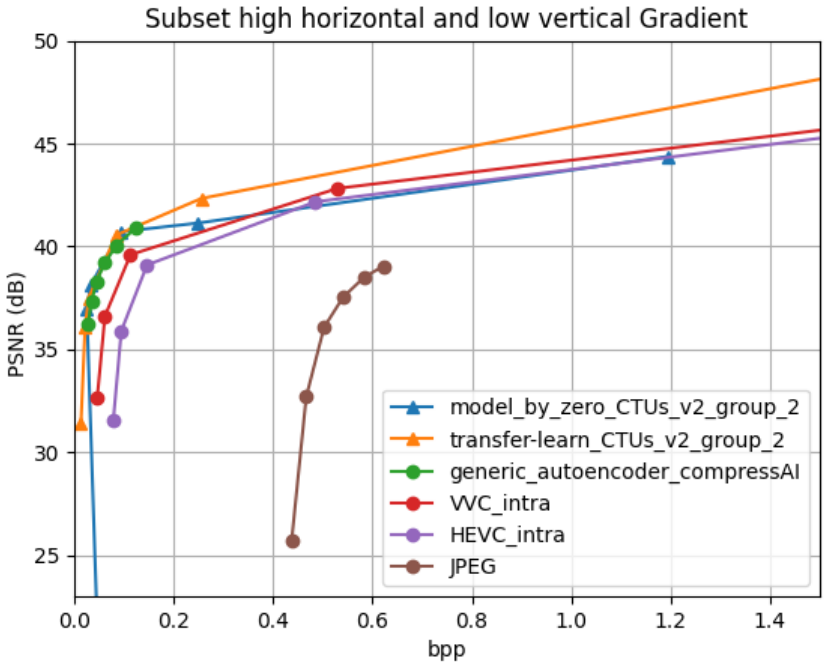

Figure 41: PSNR - Subset high horizontal and low vertical Gradient.

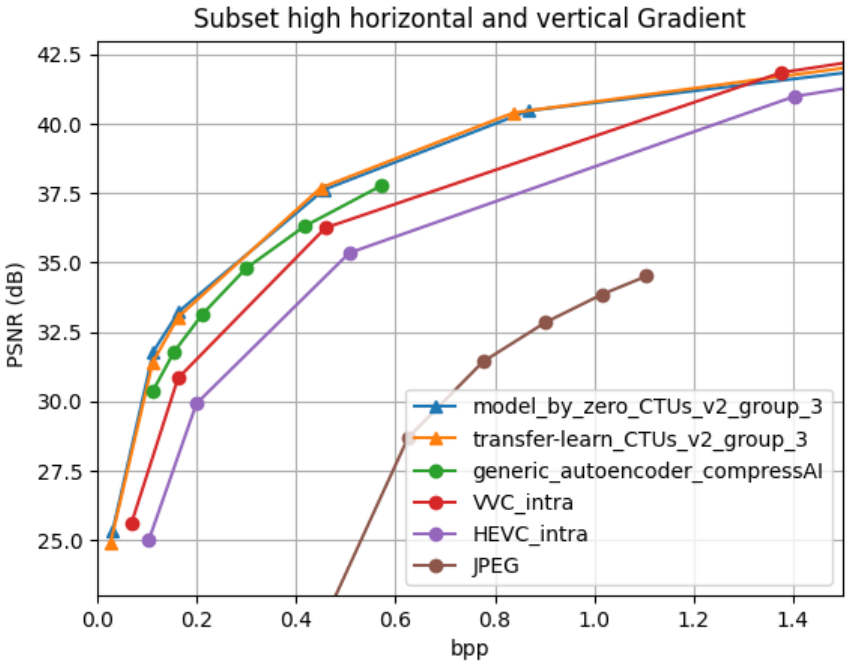

Figure 42: PSNR - Subset high horizontal and vertical Gradient.

Similar to what was observed in the case of subsets separated by the variance, it is possible to observe a better performance using the models trained from the

scratch, and the model using *transfer-learn* instead of a generic autoencoder. This is more relevant for the subsets with higher horizontal and vertical gradients. The performance of the model trained from the scratch and the model using the *transfer-learn* is quite similar for all subsets. Given this behaviour, in the following experimental studies the model trained from the scratch was not considered. Regarding the learning based approaches, for all subsets separated by horizontal and vertical gradients, the fine-tuned autoencoder (*transfer-learn*) outperforms the VVC and all other encoders, except for subsets with very high variance, where the PSNR quality is slightly below than VVC for a bpp higher than 1.3.

4.3.3 *Dataset clustering with PCA*

The PSNR results achieved by using the subsets of PCA clustering are shown in Figures 43, 44, 45 and 46, representing groups 0, 1, 2 and 3, respectively. The graph labels maintain a consistent representation as in the results presented in the previous section 4.3.1 and 4.3.2.

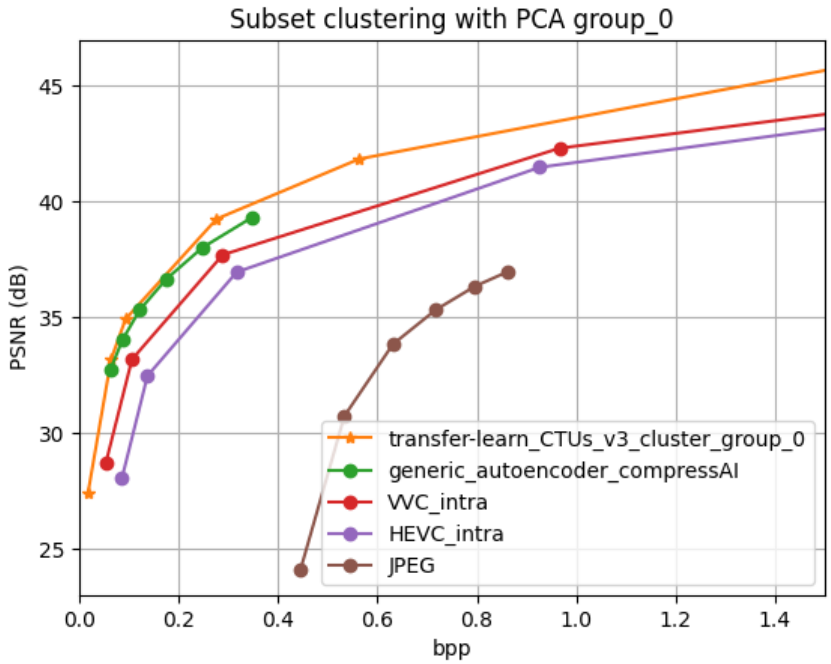

Figure 43: PSNR - Subset clustering with PCA group_0.

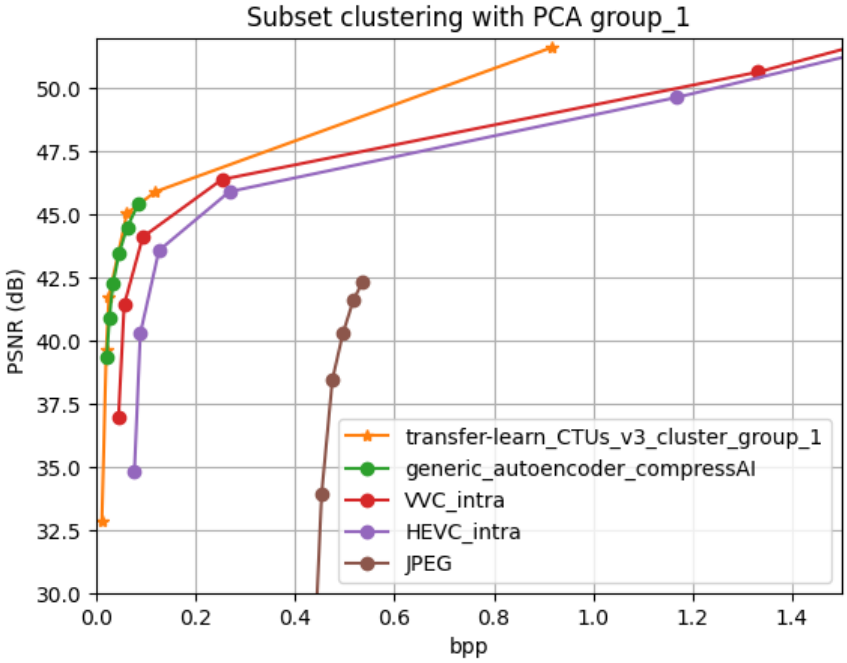

Figure 44: PSNR - Subset clustering with PCA group_1.

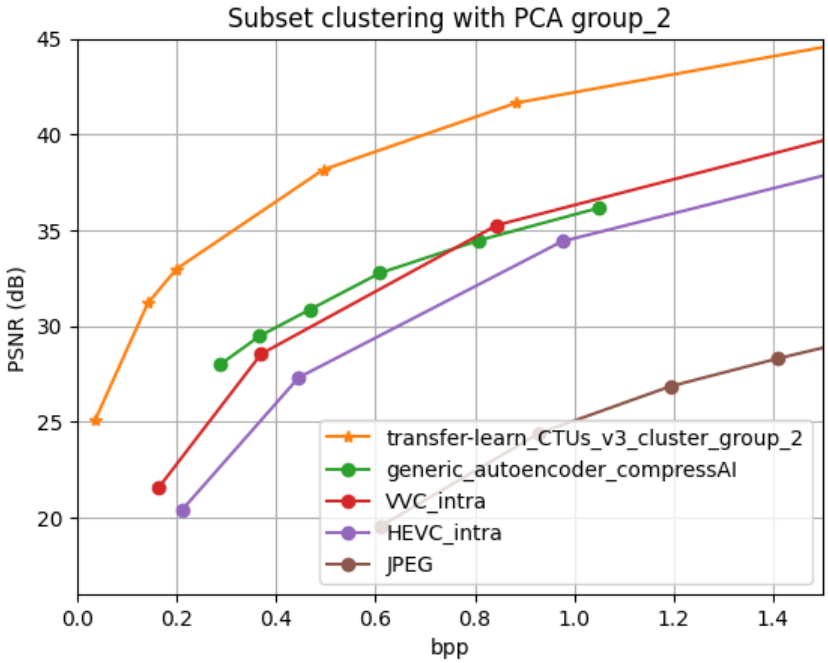

Figure 45: PSNR - Subset clustering with PCA group_2.

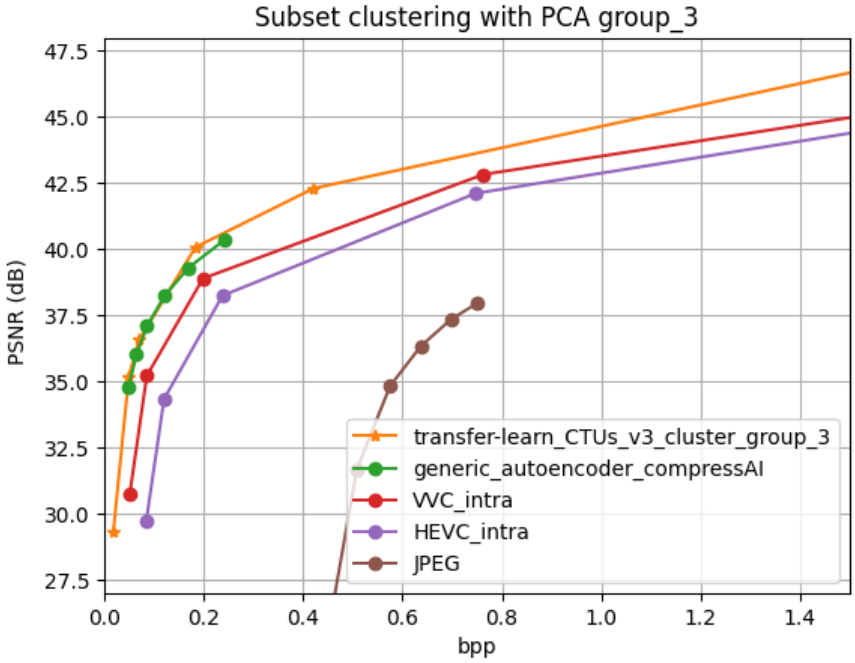

Figure 46: PSNR - Subset clustering with PCA group_3.

For the subset of the group 1, 3 and 4, the performance of *transfer-learn* model and the generic autoencoder are similar considering the low values of bpp, although there is an improvement when using the *transfer-learn* model, as in the previous sections. When using the subset of group 2 there is a significant improvement for the *transfer-learn* model, when compared to the generic autoencoder. This occurs because this subset is composed with more specif crops, such as letters as can be seen in 63(C), and doing fine tuning allows these models to learn the most efficient way to encode these crops, as has been shown in Chapter 3. In regard to the learning based approaches, for all subsets separated with cluster over the PCA, the fine-tuned autoencoder (*transfer-learn*) outperforms the VVC and all other encoders.

4.3.4 *UHD Images*

After autoencoders are trained and the performance is measured on the individual subsets, the next step is to use them to compress the UHD images. As autoencoders were trained with 128x128 pixel crops, the valid input is to use crops of the same size. Thus, it is necessary to crop the entire UHD image into 128x128 pixels, as shown in Figure 47. Then, depending on the chosen metric (variance, gradients or cluster by PCA), each crop is assigned to a group that is compressed by the

respective autoencoder. This process is done for all crops belonging to the entire UHD image.

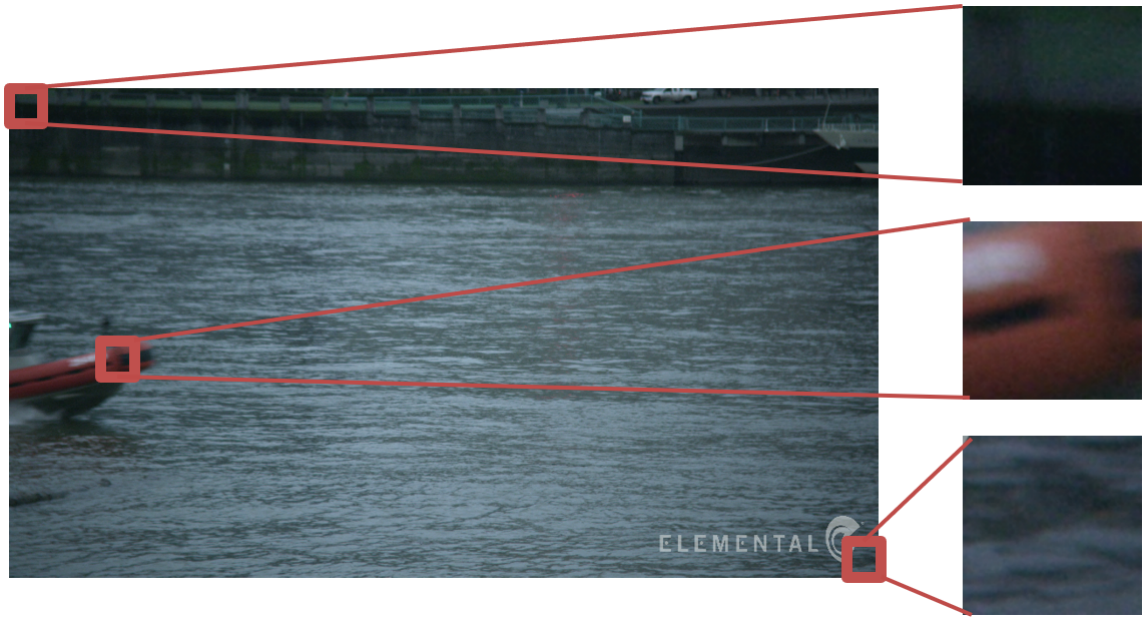

Figure 47: Example of some crops from UHD image.

The UHD test set were created by selecting 9 UHD images from the "HDR-10 calibration and test patterns set" [52] and frames of "videvo" [53]. Figure 72 show the UHD images used to evaluate the proposed method. The PSNR is measured by comparing the original UHD image to the reconstructed one, and the bpp is calculated by averaging all bpp of the crops that compose the UHD image. This is possible because all crops have the same resolution.

The PSNR results using a group of autoencoders trained using different subsets are shown in Figure 48. The blue line represents the performance using autoencoders trained with crops selected by the variance, the orange line shows the performance using autoencoders trained with crops selected by the gradient, the red line shows the performance using autoencoders trained with crops clustering by PCA, and the green line represents the performance of the generic autoencoder. Using different metrics to separate subsets for training autoencoders leads to different compression performances. The best results are achieved by the autoencoders trained with subsets clustered based on PCA, except for low bpp where they are slightly outperformed by the generic autoencoder.

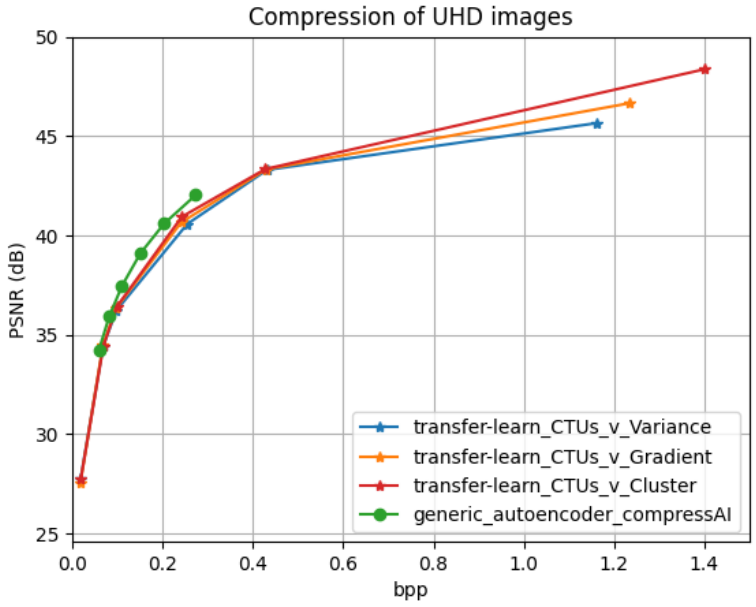

Figure 48: PSNR - UHD images tests set compression with autoencoders using different subsets.

Figure 49 compares the PSNR of reconstructed UHD images using the autoencoders trained by the subsets cluster by PCA and the traditional encoders. Analysing the curves it is possible to observe that the performance of the VVC is better than the autoencoders, except for the generic encoder that presents a better quality for low bpp.

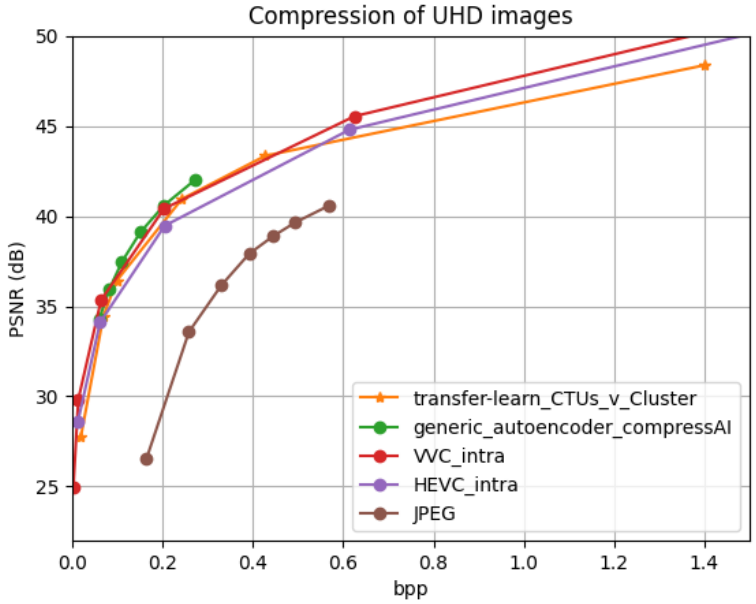

Figure 49: PSNR - UHD images tests set.

4.3.5 *360° Images*

The compression of 360° images with autoencoders follows the same procedure that was presented in the previous subsection for UHD images. The entire image is divided into crops of 128x128 pixels, which are given by input to the autoencoders, and the image is reconstructed crop by crop. To create the 360° test set 8 360° images were selected. Figure 73 show the 360° images used to evaluate the proposed method.

As mentioned before, the PSNR is measured by comparing the original 360° image with the reconstructed one, and the bpp is calculated by averaging all bpp per crops that compose the 360° image, which is possible because the tested images have a width and height multiples of 128. The PSNR results using a group of autoencoders trained by different metric is shown in Figure 50. The graph labels maintain a consistent representation as in the results presented in the previous section. A difference between compression performances can be noticed using autoencoders trained with subsets separated using different metrics. The autoencoders trained with subsets selected by cluster the crops considering PCA have a better performance compared with the other groups of autoencoders, but at low bpp the PSNR quality is slightly lower than the generic autoencoder, showing similar performance with autoencoders trained with subsets selected by gradients.

Figure 51 shows the PSNR of reconstructed 360° images using the autoencoders trained by the subsets cluster by PCA and the traditional encoders. Analysing this figure it is possible to see the fine-tuned autoencoders (*transfer-learn*) trained using the subsets divided by cluster crops using PCA to outperform VVC and all other encoders. This performance improvement is more significant for bpp bigger than 0.4.

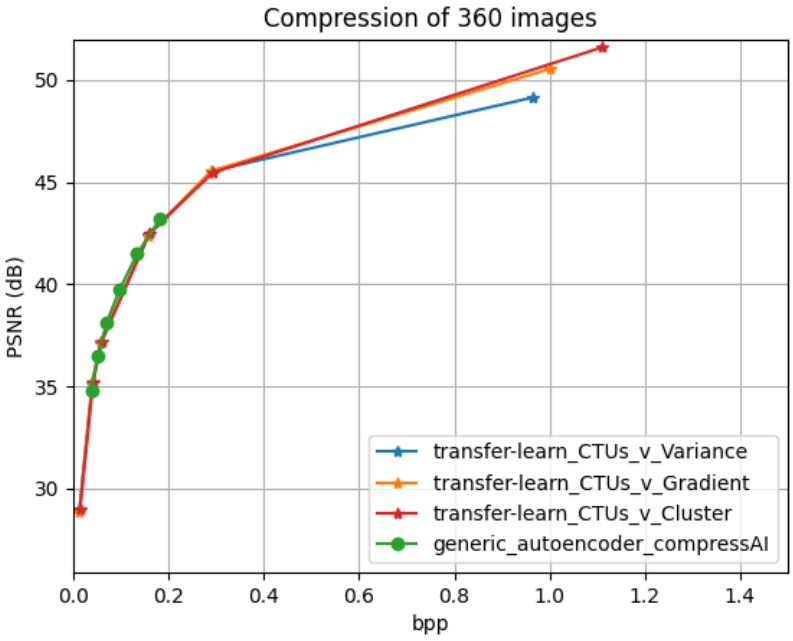

Figure 50: PSNR - 360° images tests set compression with autoencoders using different metrics.

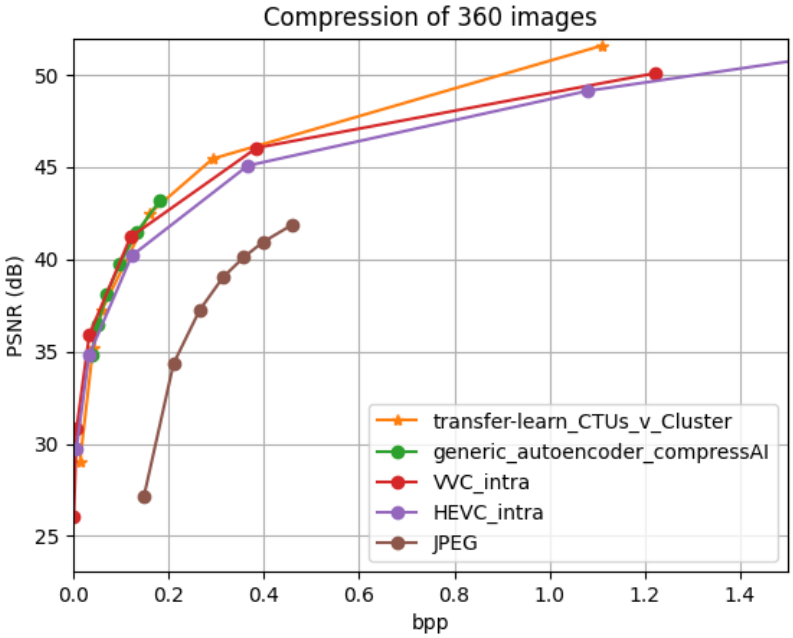

Figure 51: PSNR - 360° images tests set.

In order to evaluate the performance of the encoder in different regions of the 360° images, it was decided test the encoders by zones (coined as “north”, “equator”

and “south”), as shown in Figure 52. The equator zone occupies half the image while the poles represent a quarter of the image.

Figure 52: Example of 360° images divided by zones.

The PSNR of the north, equator and south zones of 360° images are shown in Figures 53, 54, and 55, respectively. The autoencoders trained with subsets selected by using variance achieved a similar compression performance to VVC. The autoencoders trained with subsets divided by the gradient and cluster using PCA have better performance than VVC, which is more significant for higher bpp. Comparing the difference in performance between zones it is possible to observe that autoencoders trained with subsets selected define based on gradients and clustering using PCA achieve higher compression gains than VVC in all zones of images. This difference between the performance of VVC and these autoencoders on the poles of the images (north and south) was more significant than of the equator. This fact is probably justified because the equator presents less distortion and more detail in comparison to the poles. This fact helps to support the higher performance of VVC in UHD images in comparison to the autoencoders because the distortion noticed in 360° images does not exist in UHD images.

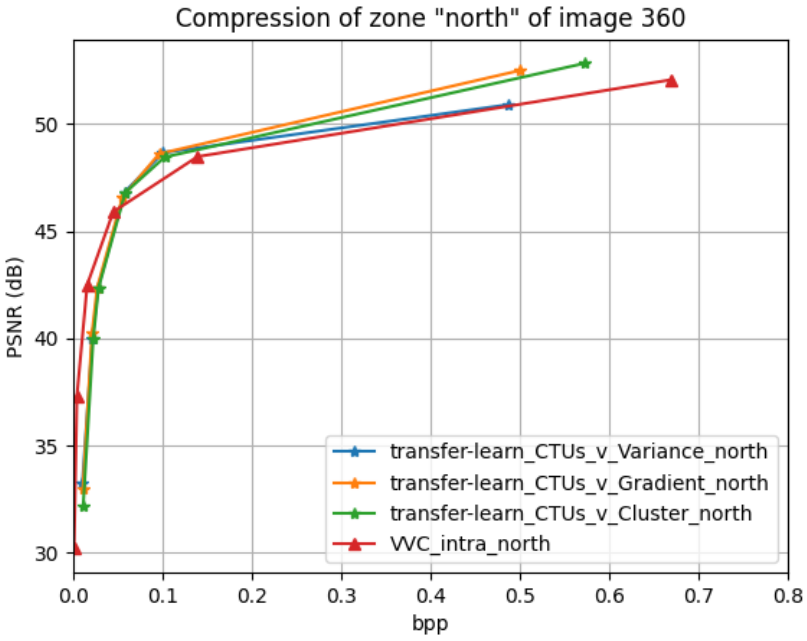

Figure 53: PSNR - 360° images tests set zone “North”.

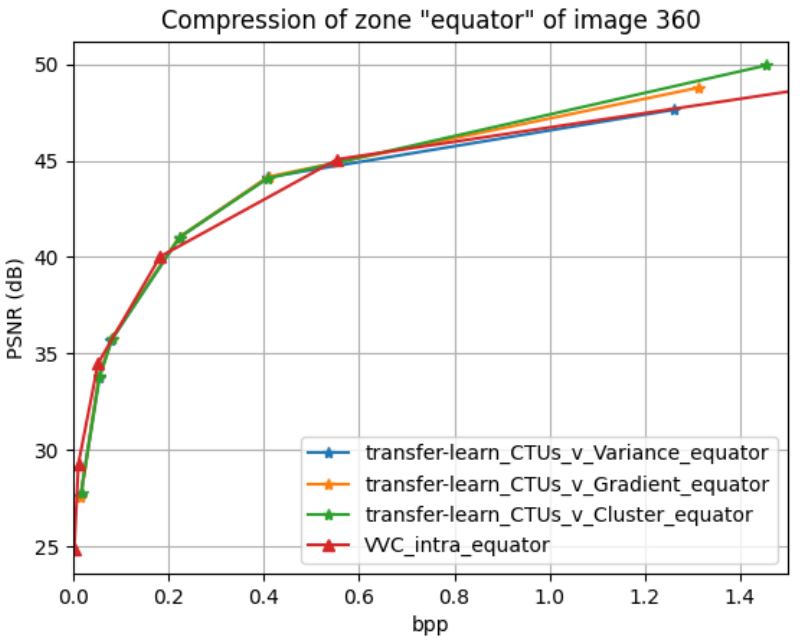

Figure 54: PSNR - 360° images tests set zone “Equator”.

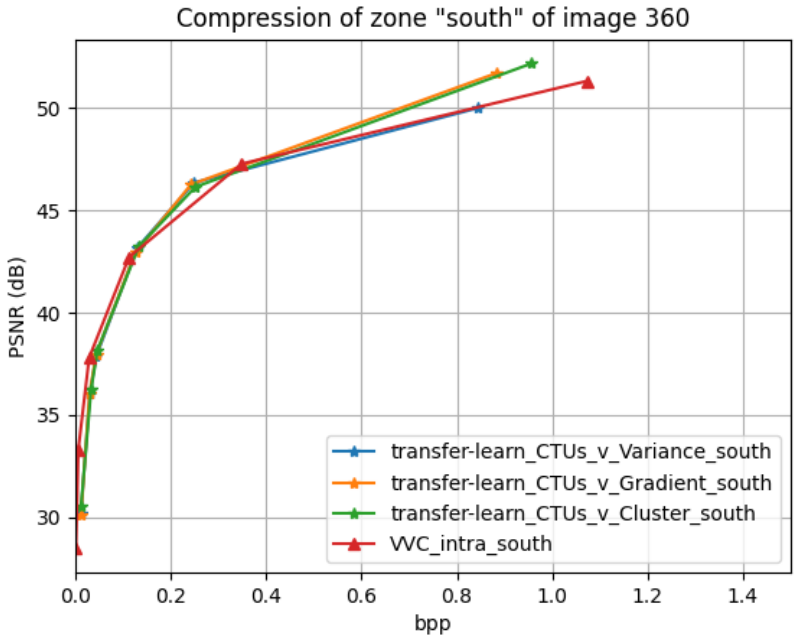

Figure 55: PSNR - 360° images tests set zone “South”.

4.4 SUMMARY

In this chapter, a novel coding approach was described based on multiple autoencoders to compress UHD and 360° images by regions, considering the image properties of each region. Different metrics were used to create the subsets such as the variance, horizontal and vertical gradients and the PCA

The performance of autoencoders and the standard encoders for each subset were presented, showing that the autoencoders consistently achieve better results than VVC.

Analysing the performance of the compression for UHD and 360° images using different groups of autoencoders, trained with subsets separated with different metrics, it is possible to observe that the criteria chosen to create the subsets is relevant. For both test sets, the autoencoders trained using the subsets created by the cluster using PCA achieved the best performance.

For the UHD images, VVC achieves a better performance than the proposed approach.In contrast, the proposed encoding scheme achieves better results than VVC for the compression of 360° images, mainly because the autoencoders achieved a significantly better performance encoding the poles of 360° images.

5

CONCLUSIONS AND FUTURE WORK

5.1 CONCLUSIONS

The main objective of this work was to explore and develop an efficient solution for compression using a machine learning approach. Image compression is an important research field that aims to achieve more compact representation of data while keeping the quality of the image. With the increase of computational resources the end-to-end learned-based methods have grown to achieve an efficient way of compression. In this dissertation, two approaches were proposed using multiple autoencoders.

The first learning-based compression proposed method had as main objective the compression of visual objects, targeting not only better performances in terms of image quality performance, but also higher accuracy on some machine driven tasks. To validate this approach a specific case was addressed: a surveillance scenario with 5 different object classes (people, cars, motorbikes, bikes and faces). It was created a dataset for each considered class and different autoencoders were trained, each one specifically optimised for one object class. Using object detection and classification, the target class of objects present in the image were compressed using the autoencoder trained to that class. As a result, the proposed encoding scheme consistently achieves better results than other traditional encoders, including the state of the art VVC. The accuracy of machine driven tasks is also measured consider object classification and face recognition, and the proposed encoding scheme achieves better results compared to VVC. It was demonstrated that fine-tuning of generic optimised autoencoders through transfer learning, yields improved compression efficiency and the performance of task-driven by machines. Smart surveillance is the envisaged application field for this work, which also opens a promising research direction towards learning-based compression of visual objects.

A second learning-based compression proposed for the efficient compression of UHD and 360° images. It was created a generic dataset consisting of crops taken from UHD and 360° images. Additionally, considering different image properties, various subsets were created, each one of them used to training a specific autoencoder. For

each considered metric four different autoencoders were trained, each targeting a particular range of values of that metric. Which is to say that the different autoencoders are optimized to particular characteristics of the image.

The proposed encoding scheme achieves better results than VVC for 360° images, which does not occur for UHD images. Also, it was observed that the performance of autoencoders depends on the metric used to separate the generic dataset, indicating its importance in the strategy used to create the subsets. In fact, it was shown that for 360° and UHD images different compression performances are obtained, depending on the image subset used to training the autoencoders: clustering with PCA based presenting the best performance and variance based on the worst.

Overall, the proposed compression approach using several autoencoders achieves better performance than the VVC, except for UHD image compression. Such achievement can be beneficial in the storage and transmission of compressed images, as well as to improve the accuracy of the systems with tasks driven by machines.

5.2 FUTURE WORK

As future work, it would be interesting to study some additional modules or even a different architecture of autoencoders, such as local multi-level feature fusion (LMLF) or concatenated residual modules (CRM), aiming to improve the performance of the compression performance.

Concerning the visual compression of visual objects firstly addressed, given the limited performance observed on images with objects like bikes and motorbikes other segmentation schemes should be considered. Indeed, "better" datasets can positively impact on the ability of autoencoders to apprehend objects' features. It also can be assessed the performance of this approach when compressing the entire image, and in this can it will be necessary to implement an efficient way to compress the background.

Also, as the performance of the approach proposed to compress UHD images is still below VVC, improvements should be considered, for example, by using the information of the neighbouring crops similar to the algorithm of predictions between CTUs used in VVC. Given the dependence of autoencoders' performance on the scheme used to create the subsets, there is room to study other metrics or to combine multiple metrics.

5.3 CONTRIBUTIONS

Resulted contributions from the research work done during this dissertation.

- R. Antonio, S. Faria, L. M.N.Tavora, A. Navarro and P. Assuncao, "Learning-based compression of visual objects for smart surveillance", 2022 Eleventh International Conference on Image Processing Theory, Tools and Applications (IPTA), Salzburg, Austria, April 2022
- P. Assuncao, S. Faria, R. Antonio, L.M.N. Tavora, A. Navarro, "Image data compression method and device using segmentation and classification", INPI, nº20221000001678/0198, patent request submitted on 19 April 2022

BIBLIOGRAPHY

[1] J. Ballé, V. Laparra, and E. P. Simoncelli. "End-to-End Optimized Image Compression". In: *5th Int. Conf. on Learning Representations* (Mar. 2017). DOI: 10.48550/arXiv.1611.01704.

[2] J. Balle et al. "Variational image compression with a scale hyperprior". In: *the 6th Int. Conf. on Learning Representations* (May 2018). DOI: 10.48550/arXiv.1802.01436.

[3] Zhengxue Cheng et al. "Deep Convolutional AutoEncoder-based Lossy Image Compression". In: *2018 Picture Coding Symposium (PCS)*. 2018, pp. 253–257. DOI: 10.1109/PCS.2018.8456308.

[4] Z. Cheng et al. "Learned Image Compression with Discretized Gaussian Mixture Likelihoods and Attention Modules". In: *Proc. IEEE/CVF Conference on Computer Vision Pattern Recognition*. Mar. 2020, pp. 7936–7945. DOI: 10.1109/CVPR42600.2020.00796.

[5] G. Hinton and F. Cambridge. "Shape RFPRFSFNTATTON in parallel systems". In: (1981).

[6] Touradj Ebrahimi. *JPEG AI Use Cases and Requirements*. https://ds.jpeg.org/documents/jpegai/wg1n90021-REQ-JPEG_AI_Use_Cases_and_Requirements.pdf.

[7] A.M. Clappis. *Uma introdução as redes neurais convolucionais utilizando o Keras*. https://medium.com/data-hackers/uma-introduÃğÃčo-as-redes-neurais-convolucionais-utilizando-o-keras-41ee8dcc033e.

[8] L. Yin. *A Summary of Neural Network Layers*. https://medium.com/machine-learning-for-li/different-convolutional-layers-43dc146f4d0e.

[9] Pytorch. *CONV2D*. https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html.

[10] Luis. *What is the sigmoid function, and what is its use in machine learning's neural networks?* https://www.luisotsm.com/single-post/what-is-the-sigmoid-function-and-what-is-its-use-in-machine-learning-s-neural-networks.

[11] admin. *Understand tanh(x) Activation Function: Why You Use it in Neural Networks*. https://www.tutorialexample.com/understand-tanhx-activation-function-why-you-use-it-in-neural-networks/.

[12] dvdbisong. *Deep Learning*. https://ekababisong.org/ieee-ompi-workshop/deep_learning/.

[13] Pytorch. *MAXPOOL2D*. https://pytorch.org/docs/stable/generated/torch.nn.MaxPool2d.html.

[14] Indian Tech Warrior. *Fully Connected Layers in Convolutional Neural Networks*. https://indiantechwarrior.com/fully-connected-layers-in-convolutional-neural-networks/.

[15] J. Despois. *Autoencoders — Deep Learning bits*. https://hackernoon.com/autoencoders-deep-learning-bits-1-11731e200694.

[16] M. Tripathi. "Facial image denoising using AutoEncoder and UNET". In: *Heritage and Sustainable Development* 2.2 (July 2021), pp. 89–96. DOI: 10.37868/hsd.v2i2.71.

[17] M. Schreyer et al. "Detection of anomalies in large scale accounting data using deep autoencoder networks". In: *CoRR* (Aug. 2017), pp. 1–19. DOI: 10.48550/arXiv.1709.05254.

[18] L. Gondara. "Medical image denoising using convolutional denoising autoencoders". In: *2016 IEEE 16th international conference on data mining workshops (ICDMW). IEEE* (2016), pp. 241–246. DOI: 10.1109/ICDMW.2016.0041.

[19] A. Ng. "Sparse autoencoder". In: *CS294A Lecture notes*. 2011, pp. 1–19.

[20] J. Masci et al. "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction". In: *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks* (2011), pp. 52–59. DOI: 10.1007/978-3-642-21735-7_7.

[21] D. P. Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: 2013. DOI: 10.48550/arXiv.1312.6114.

[22] Joao Ascenso. *JPEG AI Dataset*. https://jpeg.org/jpegai/dataset.html.

[23] A. Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* (2017), pp. 5998–6008. DOI: 10.48550/arXiv.1706.03762.

[24] M. Lu et al. "Transformer-based Image Compression". In: *Proc. DCC'22* (Nov. 2021), pp. 469–469. DOI: 10.48550/arXiv.2111.06707.

[25] N. Carion et al. “End-to-end object detection with transformers”. In: *European Conference on Computer Vision* (2020), pp. 213–229. DOI: 10.48550/arXiv.2005.12872.

[26] D. Zhou et al. “Deepvit: Towards deeper vision transformer”. In: (2021). DOI: 10.48550/arXiv.2103.11886.

[27] Y. Bai et al. “Towards End-to-End Image Compression and Analysis with Transformers”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36 (2022), pp. 104–112. DOI: 10.48550/arXiv.2112.09300.

[28] V. Goyal. “Theoretical foundations of transform coding”. In: *IEEE Signal Processing Magazine 18.5* (Sept. 2001), pp. 9–21. DOI: 10.1109/79.952802.

[29] D. Minnen, J. Ballé, and G. Toderici. “Joint Autoregressive and Hierarchical Priors for Learned Image Compression”. In: *Proc. Adv. Neural Inf. Process. Syst.* (Sept. 2018), pp. 10771–10780. DOI: 10.48550/arXiv.1809.02736.

[30] L. Theis et al. “Lossy Image Compression with Compressive Autoencoders”. In: *5th Int. Conf. on Learning Representations* (Mar. 2017). DOI: 10.48550/arXiv.1703.00395.

[31] H. Fu et al. “Learned Image Compression with Discretized Gaussian-Laplacian-Logistic Mixture Model and Concatenated Residual Modules”. In: *IEEE Transactions on Image Processing* (July 2021). DOI: 10.48550/arXiv.2107.06463.

[32] E. Agustsson et al. “Soft-to-hard vector quantization for end-to-end learning compressible representations”. In: *Proc. 31st Int. Conf Neural Inf. Process. Syst.* (June 2017), pp. 1141–1151. DOI: 10.48550/arXiv.1704.00648.

[33] G. Toderici et al. “Variable rate image compression with recurrent neural networks”. In: *The International Conference on Learning Representations* (Mar. 2016). DOI: 10.48550/arXiv.1511.06085.

[34] D. Marpe, H. Schwarz, and T. Wiegand. “Context-based adaptive binary arithmetic coding in the h. 264/avc video compression standard”. In: *IEEE Transactions on circuits and systems for video technology* (June 2003), pp. 620–636. DOI: 10.1109/TCSVT.2003.815173.

[35] J. Lee, S. Cho, and K. Beack. “Context-Adaptive Entropy Model For End-to-End Optimized Image Compression”. In: *Int. Conf. on Learning Representations (ICLR)* (May 2019). DOI: 10.48550/arXiv.1809.10452.

[36] G. Flamich, M. Havasi, and J. M. Hernández-Lobato. "Compressing Images by Encoding Their Latent Representations with Relative Entropy Coding". In: *Neural Info. Process. Sys. 34* (Apr. 2020), pp. 16131–16141. DOI: 10.48550/arXiv.2010.01185.

[37] D. Minnen, J. Ballé, and G.D. Toderici. "Joint autoregressive and hierarchical priors for learned image compression". In: (2018), pp. 10794–10803. DOI: 10.48550/arXiv.1809.02736.

[38] G. B. Huang et al. "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments". In: *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition* (Oct. 2008).

[39] T. Karras, S. Laine, and T. Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (Mar. 2019), pp. 4401–4410. DOI: 10.48550/arXiv.1812.04948.

[40] Yuxin Wu et al. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019.

[41] J. Bégaint et al. "CompressAI: a PyTorch library and evaluation platform for end-to-end compression research". In: *CompressAI: a PyTorch library and evaluation platform for end-to-end compression research*. 2020. URL: https://arxiv.org/abs/2011.03029.

[42] T. Xue et al. "Video Enhancement with Task-Oriented Flow". In: *International Journal of Computer Vision (IJCV)* 127.8 (2019), pp. 1106–1125.

[43] Zhi Li et al. *Toward A Practical Perceptual Video Quality Metric*. https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652.

[44] Sefik Ilkin Serengil and Alper Ozpinar. "LightFace: A Hybrid Deep Face Recognition Framework". In: *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*. IEEE. 2020, pp. 23–27. DOI: 10.1109/ASYU50717.2020.9259802. URL: https://doi.org/10.1109/ASYU50717.2020.9259802.

[45] A. Mercat, M. Viitanen, and J. Vanne. "UVG Dataset: 50/120fps 4K Sequences for Video Codec Analysis and Development". In: *Proceedings of the 11th ACM Multimedia Systems Conference* (June 2020), pp. 297–302.

[46] L. Song et al. "The SJTU 4K video sequence dataset". In: *Fifth International Workshop on Quality of Multimedia Experience (QoMEX)* (July 2013), pp. 34–35. DOI: 10.1109/QoMEX.2013.6603201.

[47] M. Cheon and J. S. Lee. "Subjective and objective quality assessment of compressed 4K UHD videos for immersive experience". In: *IEEE Transactions on Circuits and Systems for Video Technology* 28.7 (July 2018), pp. 1467–1480. DOI: 10.1109/TCSVT.2017.2683504.

[48] L. Katsavounidis. "NETFLIX – "Chimera" video sequence details and scenes". In: (Nov. 2015).

[49] Xiph.org Foundation. "Xiph.org Test Media". In: *Xiph.org Test Media*. 2016. URL: https://media.xiph.org/.

[50] J. Boyce et al. "JVET common test conditions and software reference configurations". In: *7th Meeting, Torino, IT* (July 2017).

[51] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations* (2014). DOI: 10.48550/arXiv.1409.1556.

[52] AVS FORUM. *HDR-10 calibration and test patterns set*. https://www.avsforum.com/threads/hdr10-test-patterns-set.2943380/.

[53] Videvo. *All Stock Video Footage 4K*. https://www.videvo.net/stock-video-footage/freeclips/yes/resolution/4k/?page=0.

A

APPENDIX A

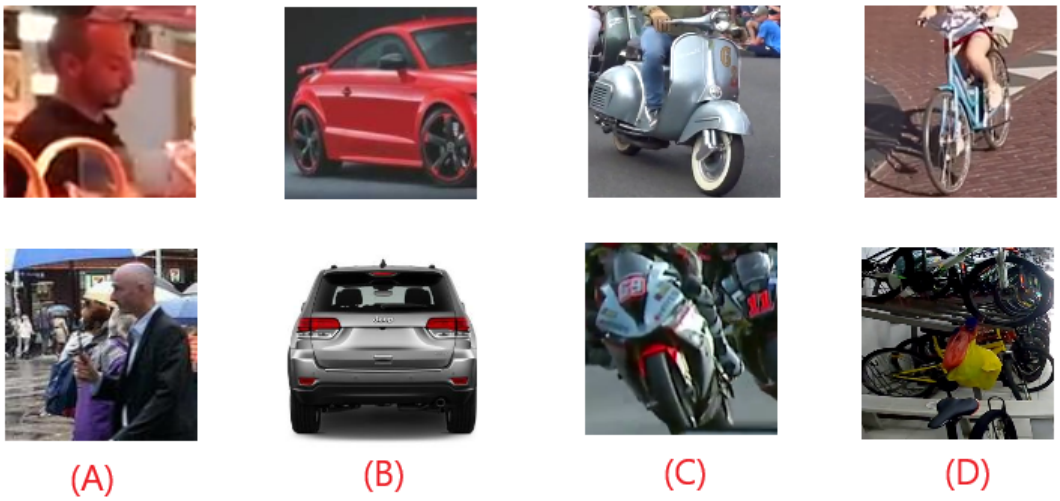

Figure 56: Example of original images datasets. *A*:people, *B*:cars, *C*:motorbikes, *D*:bikes.

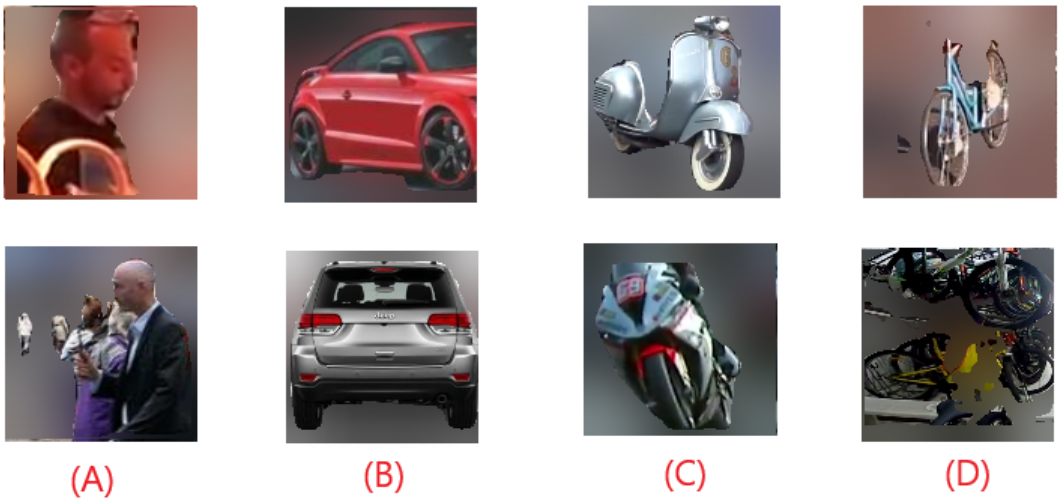

Figure 57: Example of blurred background images datasets. *A*:people, *B*:cars, *C*:motorbikes, *D*:bikes.

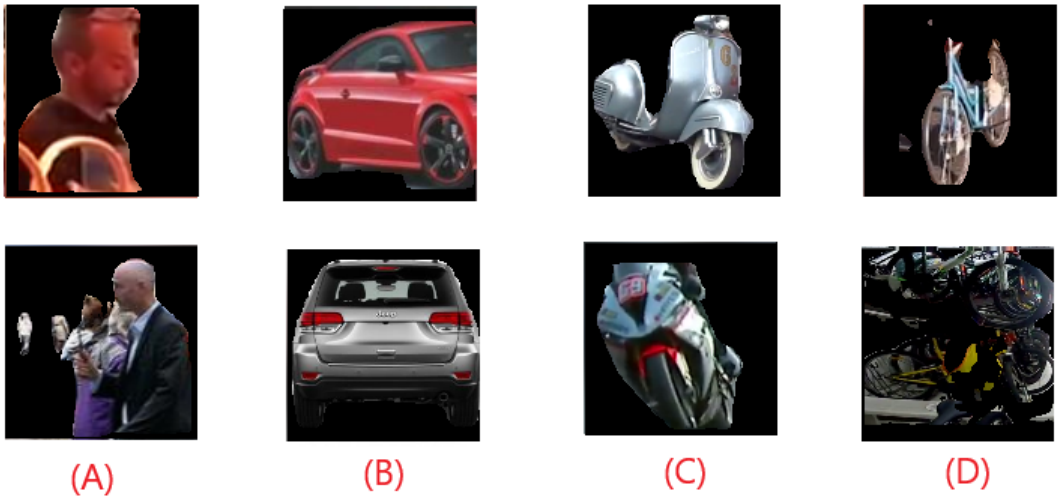

Figure 58: Example of images datasets without background. A:people, B:cars, C:motorbikes, D:bikes.

Figure 59: Example of images datasets one object in each image center of the image without background. A:people, B:cars, C:motorbikes, D:bikes.

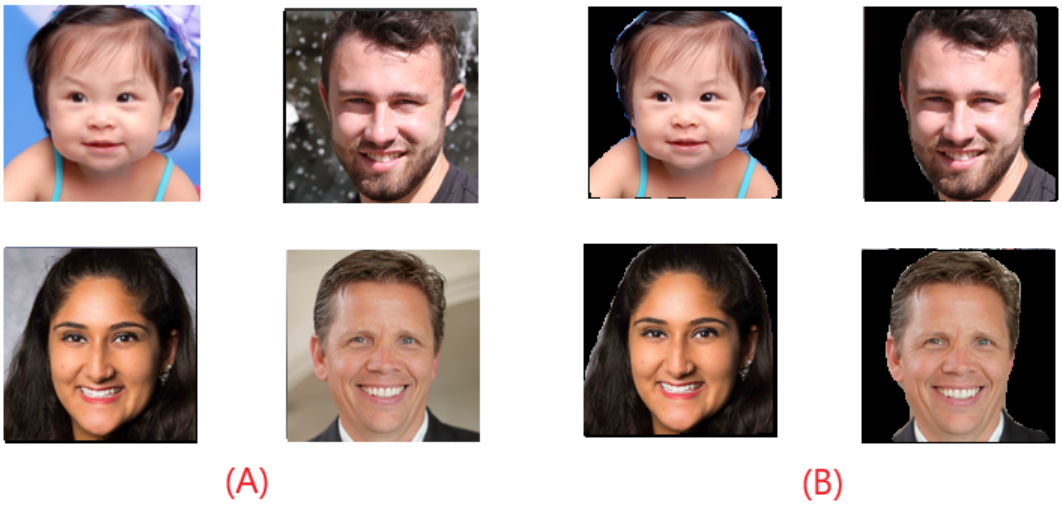

Figure 60: Example of faces images datasets. *A*:Original images, *B*:Images without background center.

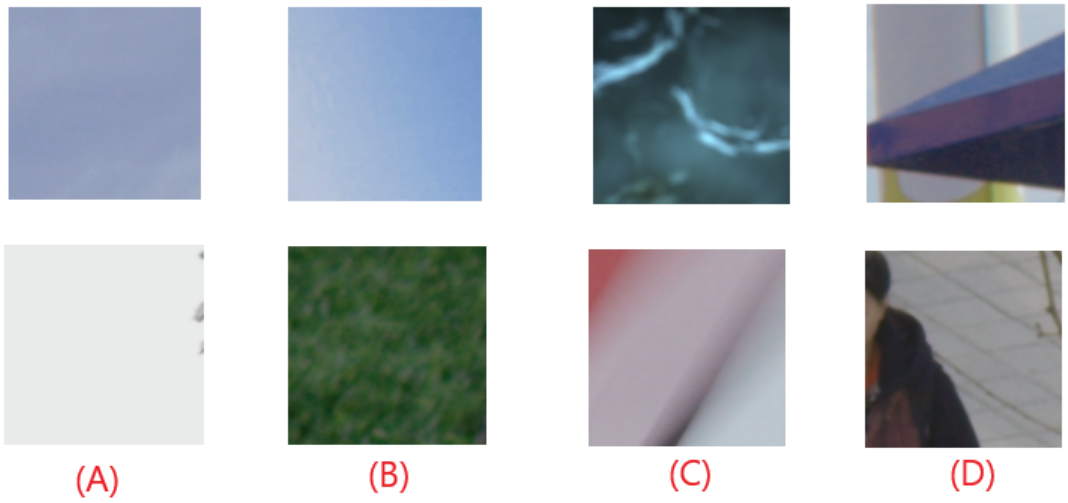

Figure 61: Example of subsets divided using variance. *A*:Very low variance, *B*:Low variance, *C*:High variance, *D*:Very high variance.

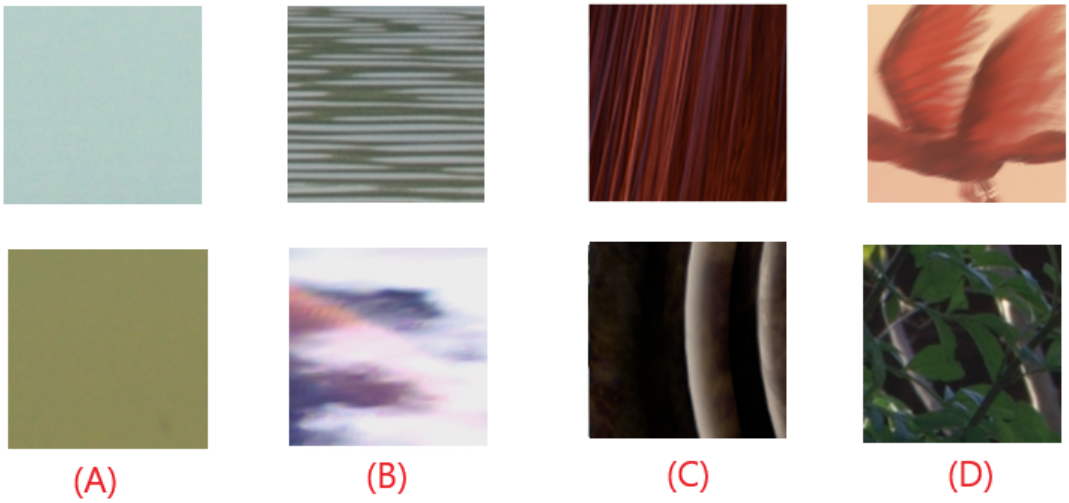

Figure 62: Example of subsets divided using horizontal and vertical gradient. *A*:Low horizontal and vertical gradient, *B*:Low horizontal and high vertical gradient, *C*:High horizontal and low vertical gradient, *D*:High horizontal and vertical gradient.

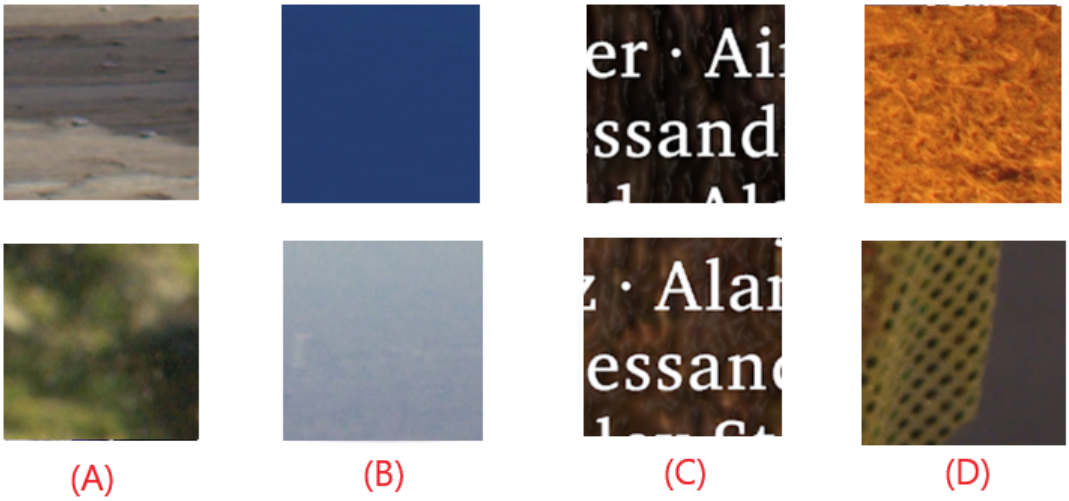

Figure 63: Example of subsets divided clustering the images using PCA. *A*:Group 0, *B*:Group 1, *C*:Group 2, *D*:Group 3.

B

APPENDIX B

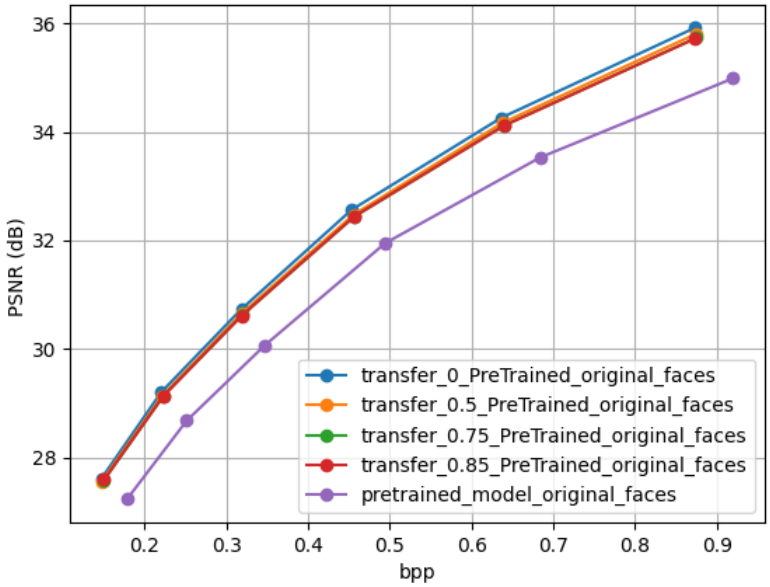

Figure 64: PSNR - faces dataset with original background using different transfer learning approaches.

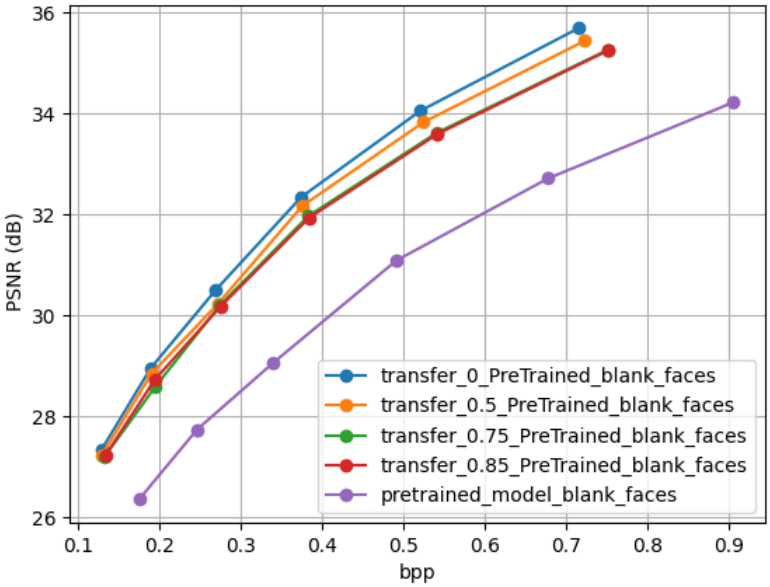

Figure 65: PSNR - faces dataset without background using different transfer learning approaches.

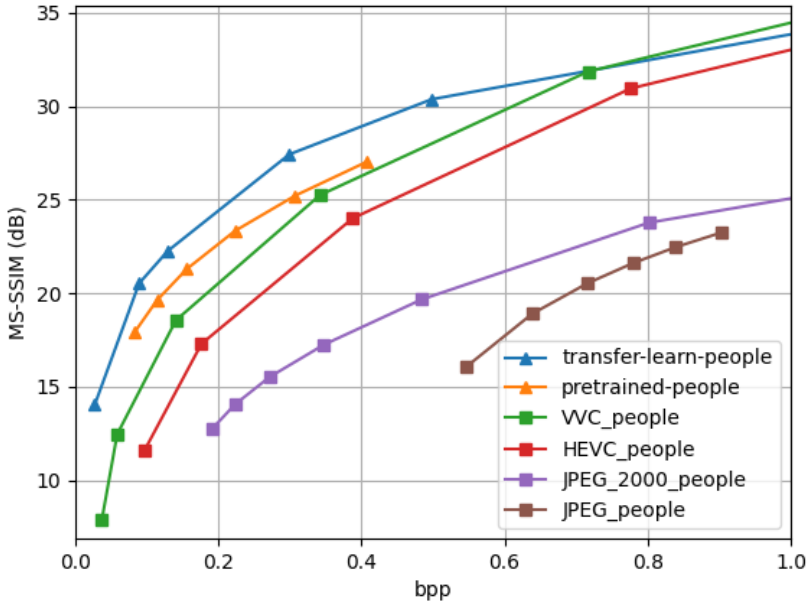

Figure 66: MS-SSIM[dB] - people dataset.

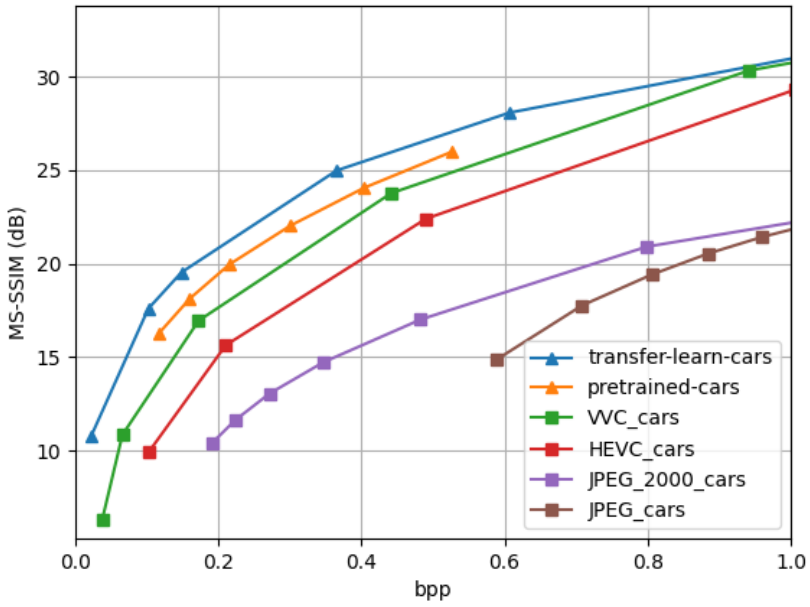

Figure 67: MS-SSIM[dB] - cars dataset.

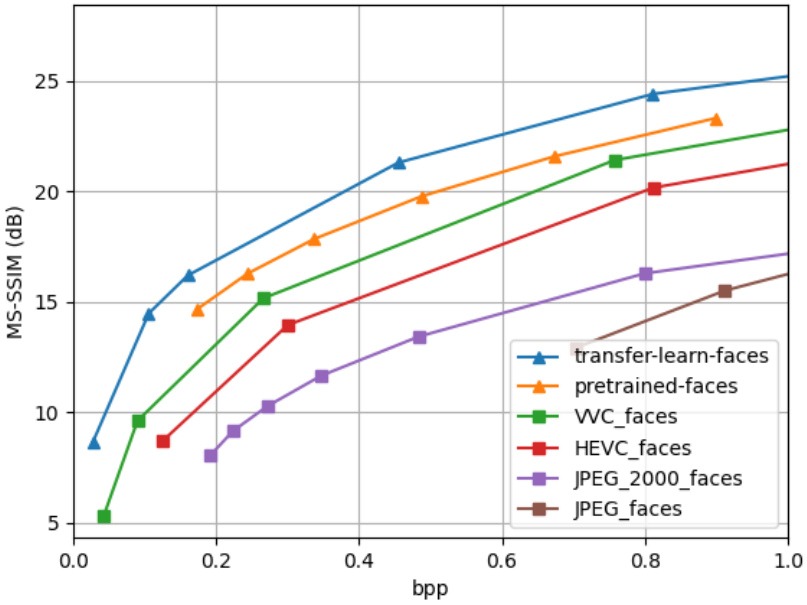

Figure 68: MS-SSIM[dB] - faces dataset.

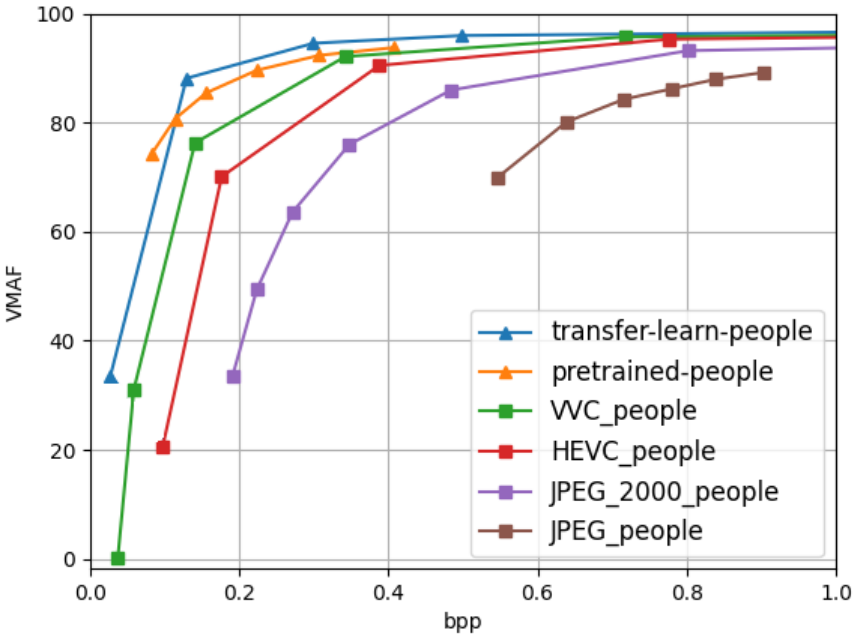

Figure 69: VMAF - people dataset.

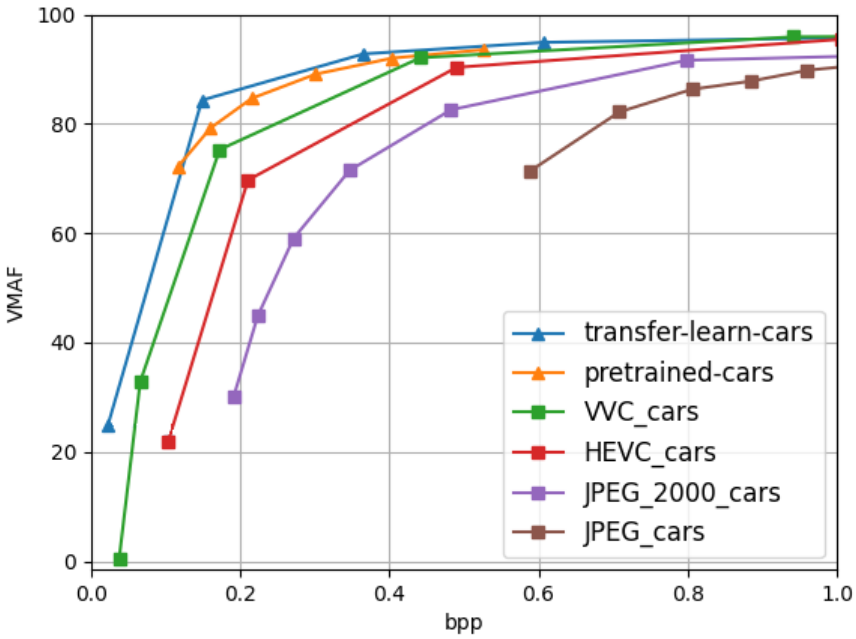

Figure 70: VMAF - cars dataset.

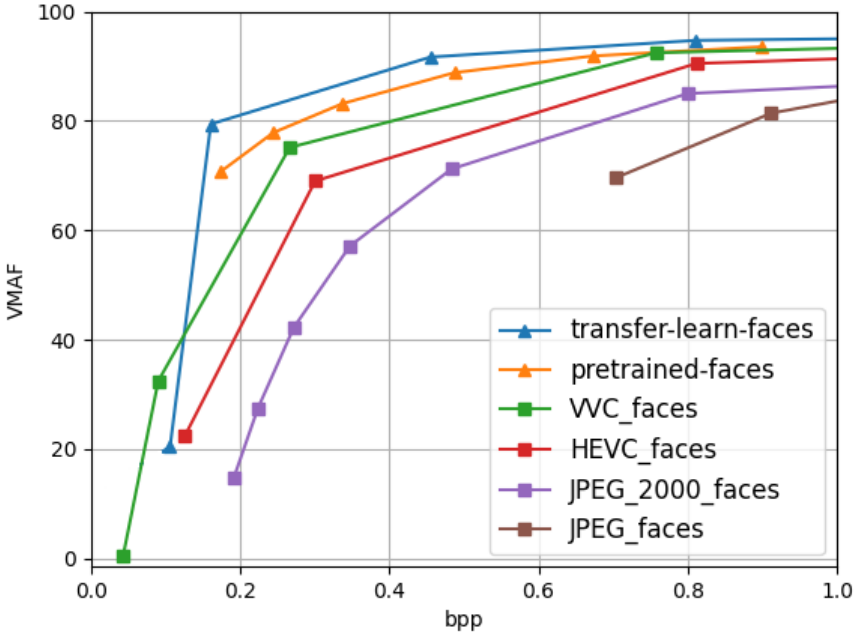

Figure 71: VMAF - faces dataset.

C

APPENDIX C

Figure 72: Subset test of UHD images.

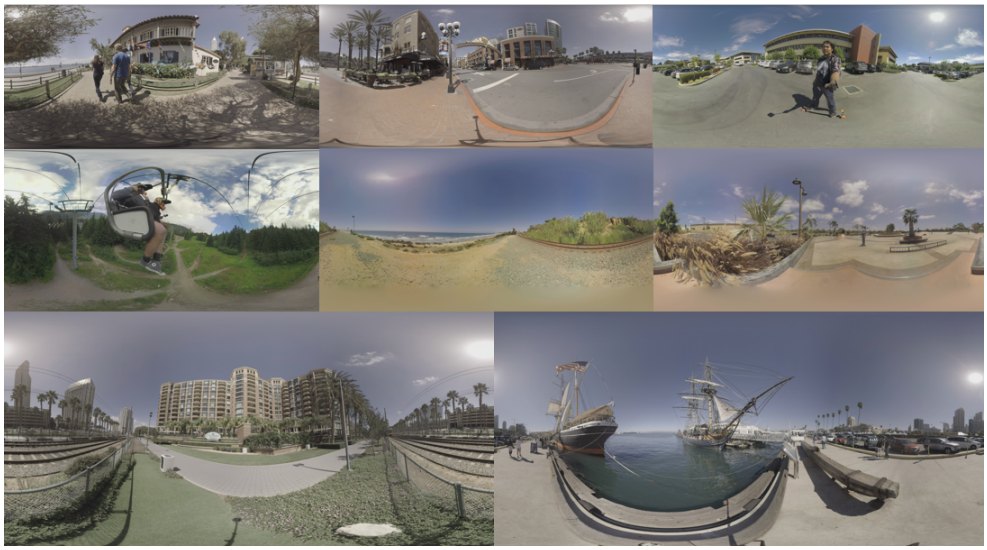

Figure 73: Subset test of 360° images.

D E C L A R A Ç Ã O

Declaro, sob compromisso de honra, que o trabalho apresentado nesta dissertação, com o título *"Learning-based image compression using multiple autoencoders"*, é original e foi realizado por Rúben Duarte António (2202288) sob orientação do Professor Pedro Antonio Amado de Assunção, Professor Luís Miguel de Oliveira Pegado de Noronha e Távora, e Professor Sérgio Manuel Maciel de Faria.

Rúben Duarte António