How fake is my image?
Evaluation of Generative Adversarial Networks

Marta Filipa de Pinto Marques

Thesis to obtain the Master of Science Degree in:

Electrical and Computer Engineering

Supervisor(s): Prof. João Miguel Duarte Ascenso

Prof. Catarina Isabel Carvalheiro Brites

Examination Committee

Chairperson: Prof. José Eduardo Charters Ribeiro da Cunha Sanguino

Supervisor**:** Prof. João Miguel Duarte Ascenso

Member of the Committee: Prof. Luís Eduardo de Pinho Ducla Soares

September 2021

Declaration

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

Acknowledgements

First and foremost, I want to thank my parents, Fernanda, and Nuno, for being my biggest cheerleaders since day one. And even though they put me on this earth without my consent, they always promised that the journey would be worth the breach of contract. I guess they were right, like always.

Then I would like to thank my supervisors, Prof. Dr. João Ascenso and Prof. Dr. Catarina Brites, for supporting me with kindness and patience throughout this project. I'm truthfully thankful to have had supervisors that are genuinely passionate when it comes to sharing knowledge. I also want to thank Instituto de Telecomunicações for making this project possible.

I would also like to thank BAC for giving me the opportunity to keep playing basketball competitively even in this tiring times. It certainly was not an easy season, but the effort put in by every single one involved gave me an additional sense of purpose when I needed it the most. A special word of appreciation to my younger teammates that make me want to be a good role model, and God knows they need it.

A big thank you to my friends that stayed close even when life didn't make it easy to do so. The resilience you show when it comes to scheduling group activities inspires me to apply that same mindset to my work, even if I probably didn't make it that day.

To my family that loves me unconditional and made sure that I was well fed and sleeping enough. I feel like I should be dead, but they made a collective effort to avoid that.

Lastly, I want to thank the amazing people that took this journey with me for every minute we spent discussing, worrying, working on some class, but also for every minute spent dreaming about becoming engineers and applying that knowledge to better the world we live in.

Abstract

Nowadays, visual information and communication systems are essential for humans everyday life. There are many services and applications that rely on visual data, from medical imaging to entertainment, many in widespread use in smartphones and social media platforms. In the past decade, Machine Learning (ML) algorithms, more precisely Deep Neural Networks (DNN) have been able to solve complex and difficult image processing and computer vision problems, such as image compression, image super-resolution, image denoising, etc. and able to extract semantic information from images, with much success.

Generative models are rather recent but have been very successful in many multimedia applications. It all started with the creation of fake images which can challenge the human perceived notion of reality. Nowadays, it is possible to create faces of persons that never existed and even entire visual scenes which look realistic and plausible. Generative Adversarial Networks (GANS) have risen in popularity among the generative models that can learn high-dimensional distributions of data, i.e. the manifold of the entire set of natural images, which are then sampled to create new never seen images. More importantly, GANs are now able to create realistic-looking images and have been applied for several image processing problems such as super-resolution or image compression.

In this context, it is rather important to assess the perceptual quality of the generative images produced by GAN-based solutions. This M.Sc. thesis aims to study and analyze the quality of the images generated with GAN based framework, for three different fields: super-resolution, denoising and compression. More importantly, other quality factors besides fidelity that play an important role in this context, such as the *naturalness/fakeness* of the image, will be addressed in such analysis. To achieve this objective, a crowdsourcing-based subjective evaluation of the GAN based solutions was performed and the performance of objective quality metrics was evaluated.

Keywords: Image Quality Assessment, Generative Adversarial Networks, Machine Learning, Deep Neural Networks

Resumo

Nos dias que correm, informação visual e os sistemas de comunicação são essenciais e dominam o nosso quotidiano. Há imensos serviços e aplicações que estão dependentes de dados visuais como é caso da imagiologia medica ou do entretenimento, muitos em larga escala usados nos nossos telemóveis e nas redes sociais. Na última década, algoritmos de aprendizagem automática, nomeadamente *Deep Neural Networks* (DNN), começaram a ser usados para resolver problemas difíceis e complexos no ramo da visão computacional e do processamento de imagem, como por exemplo, compressão, super-resolução, *denoising* (redução de ruído), etc. e alcançaram grande sucesso nestas tarefas por serem capazes de extrair informações semânticas contidas nas imagens.

Modelos generativos são relativamente recentes e têm vindo a provar ser muito bem-sucedido em diversas aplicações de multimédia. Tendo começo na criação de imagens ou de vídeos *falsos* que conseguem desafiar a perceção humana da realidade. Hoje em dia é possível criar caras de pessoas que nunca existiram ou até mesmo cenários visuais que parecem reais e plausíveis, não o sendo. *Generative Adversarial Networks* (GAN) têm vindo a crescer em popularidade entre os modelos generativos capazes de aprender distribuições de dados de alta dimensionalidade (isto é, a variedade de todo o conjunto de imagens naturais, que são utilizadas para gear novas images). Principalmente, as GANS são agora capazes de criar imagens realistas e têm sido aplicadas com sucesso a várias tarefas como a compressão de imagens como por exemplo compressão ou super resolução.

Neste contexto, é fundamental avaliar a qualidade percetual das imagens geradas por soluções que utilizem GANs. Esta tese tem como objetivo fazer uma análise da qualidade de imagens geradas por soluções baseadas em GANs em três ramos diferentes: super resolução, remoção de ruido e compressão. Visto que neste contexto outros fatores, para além da fidelidade, influenciam grandemente a qualidade de imagem, como é caso da *naturalidade/falsidade* da imagem, irá ser feita essa análise. Para isso é feito um estudo *crowdsourcing* subjetivo da qualidade de imagem das soluções que utilizam GANs onde se avalia a qualidade das imagens, assim como, o desempenho de métricas objetivas na avaliação da qualidade de imagem.

Palavras-Chave: Avaliação de qualidade de imagem, Generative Adversarial Networks, Aprendizagem Automática, Redes Neuronais Profundas

Table of Contents

List of Figures

List of Tables

List of Acronyms

AE	Autoencoder
AI	Artificial Intelligence
BN	Batch Normalization
BPG	Better Portable Graphics
Bpp	Bits per pixel
cGAN	Conditional Generative Adversarial Network
CNN	Convolutional Neural Network
Conv	Convolutional
DAE	Denoising Autoencoder
DL	Deep Learning
DNN	Deep Neural Network
FC	Fully Connected
GAN	Generative Adversarial Network
GC	Generative Compression
HR	High Resolution
IQA	Image Quality Assessment
KL	Kullback-Leibler
LR	Low Resolution
ML	Machine Learning
MOS	Mean Opinion Score
MSE	Mean Squared Error
MS-SSIM	Multi-Scale Structural Similarity
NN	Neural Network
PSNR	Peak signal-to-noise ratio
QF	Quality Factor
RB	Random Box
ReLU	Rectified Linear Unit
ResNet	Residual Network
RI	Random Instance
RNN	Recurrent Neural Network
SAE	Sparse Autoencoder
SC	Selective Generative Compression
SR	Super Resolution
SRGAN	Super-Resolution Generative Adversarial Network
SSIM	Structural Similarity
SVM	Support Vector Machine
VAE	Variational Autoencoder

Chapter 1

1. Introduction

This chapter aims to introduce this thesis. First, by setting the context and motivations behind this work. And then, by defining the objective and structure that the thesis will follow.

1.1 Context and Motivation

The need to develop algorithms capable of processing, analysing and understanding images have appeared with the first digital image back in the 1950s. At the time, this was a purely academic subject, and the focus was mainly to compress the images due to limitations in storage. By the end of the century, the advances in image compression were very significant with the introduction of well-known image compression algorithms, such as JPEG [1], PNG [2] and JPEG 2000 [3]. This was also a turning point for the field of computer vision and machine learning (ML), with techniques such as SVM (Support Vector Machine) [4] and the first neural networks, which have started to gain momentum to address more complex tasks such as image segmentation and classification. These were the first steps in creating algorithms that not only process images but also have some visual analysis capabilities.

The first wave of machine learning algorithms applied to image processing problems encountered some limitations due to the lack of training datasets and overall poor generalization capabilities. This was mainly due to the high dimensionality of visual data and the need to develop models that could represent it more efficiently. In the early 2010s, another wave in machine learning occurred, especially when a deep convolutional network (AlexNet) [5] won the ImageNet Large Scale Visual Recognition Challenge [6]. This neural network combined a somewhat high number of convolutional layers (deep learning), to extract progressively higher-level features from the dataset and fully connected layers to perform image classification using the extracted features.

In 2014, Generative Adversarial Networks proposed by Ian Goodfellow [7] introduced the concept of adversarial training, where two neural networks (Generator and Discriminator) are trained simultaneously and in competition with each other. The generator goal is to create new images while the discriminator is trained to distinguish between the images in the training dataset and the images created by the generator. This training method allows the generator to implicitly learn the underlying distribution of the images of the training dataset. This type of NN is discussed in more detail in Section 2.3.3 of this thesis.

Since the introduction of GANs, several developments were made in the generation of high-quality images. The initial promising results of adversarial networks caught the attention of several researchers, creating an explosion of studies on GANs in the following years, as illustrated by Figure 1 (left). These studies were focused mainly on improving the generated image quality and the convergence of the training process. Nowadays, many variants of the original GAN have been proposed which can generate high quality attracting images, as illustrated in Figure 1 (right). These GANs have been applied to several image processing problems such as text-to-image or image-to-image translation, image compression, super-resolution, denoising, and other natural image generation applications.

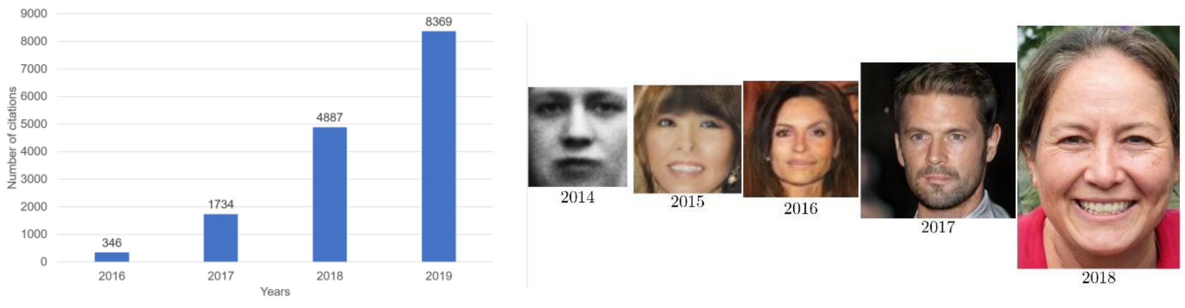


Figure 1 - Number of citations of the paper where GANs were introduced [7] between 2016 and 2019 (left). Evolution of the fake images generated by GANs between 2014 and 2018 [7, 8, 9, 10, 11] (right).

Typically, generative algorithms create images with very different characteristics of natural images and thus existing subjective assessment methodologies and objective image quality models are not suitable. For example, the images generated by GANs can still have some artefacts that show evidence that the image was synthetically generated (*fake)*; usually, the artifacts are very different from the usual artefacts created by a super-resolution or image compression solution. This motivates the need for new image quality subjective assessment experiments (also requiring some changes in the methodology) to faithfull analyse the performance of such solutions.

Moreover, the conventional metrics used to quantify image quality are quite ineffective when applied to the content generated by GANs, mainly because generative models can produce images that appear realistic and attractive but do not match when pixel-based comparisons are made. This behaviour occurs often in image compression, super-resolution or denoising GAN based systems which have shown superior perceived quality but low objective quality when popular full-reference quality metrics such as PSNR and SSIM are used. On the other hand, no-reference quality metrics do not consider the original information and thus are missing valuable (and important) information. This implies that the performance of available objective quality metrics must be revisited in this context, to understand if they are suitable to assess the performance of GAN based solutions and which quality metrics have the best performance.

1.2 Objective and Structure

The main objective of this Thesis is to perform a subjective quality assessment study to evaluate images obtained from generative adversarial network solutions. This means the use of an appropriate subjective test methodology, which was designed to measure the *naturalness/fakeness* of the images. The level of how much an image is fake is intrinsically connected to the perception of quality by humans and thus a well-known subjective methodology should provide a s olid measure. Thus, the main contribution is a generative image dataset which contains: 1) distorted images obtained from several GAN based solutions, in the fields of image super-resolution, artifact removal and compression; 2) results of a subjective test using a pairwise comparison methodology, providing a perceptual score for every image of the dataset. Moreover, recognizing the importance of objective metrics to accurately translate the human perception of quality, the correlation performance of well-known objective quality metrics are studied. To report the Thesis achievements, this document is divided into six chapters, which are described next:

- Chapter 1 presents the context and objectives.
- Chapter 2 provides an overview of the basic concepts of Neural Networks (NN), as well as a review of some selected types of NNs. This chapter culminates in a review of GANs.
- Chapter 3 presents the review three selected GAN-based solutions, namely in the following contexts: image compression, single image super-resolution, and artifact removal. These solutions were selected having into account: the importance of the problems that tackle, its popularity, and overall performance.
- Chapter 4 presents the crowdsourcing based subjective assessment test, namely the methodological approach, data processing, the solutions under evaluation and the experimental results obtained.
- Chapter 5 provides an analysis of different image quality metrics applied to the selected GAN-based solutions.
- Chapter 6 presents a summary and the future work plan.

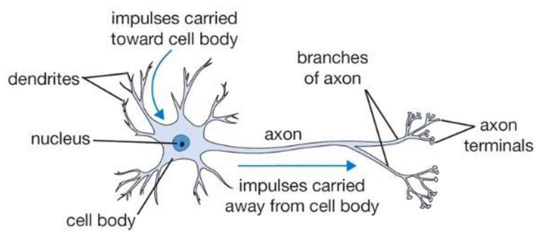
Chapter 2

2. Neural Networks: Foundations and Architectures

The idea of having a human-like intelligence in machines is the ground base for **Artificial Intelligence** (AI). AI is a scientific field whose focus is the development of computing systems with cognitive abilities. For example, a commonly used word, spoken clearly by any person, is entirely recognizable for any speaker of that language who might hear it. On the other hand, to a machine, the same word said by different orators corresponds to distinct audio signals with no clear correlation. Due to the idiosyncrasies of problems like this one, scientists begun to lean on the idea of making the computer learn from experience. This simple notion gave birth to the AI sub-field of **Machine Learning**, a computer science field that has recently received increasing attention by the research community. In ML, the idea of "learning from experience" may corresponds to a computational model that adjusts itself when presented with a large amount of data to later be able to make accurate output predictions when exposed to a different set of inputs. This chapter will focus on a specific set of ML models called **Artificial Neural Networks**, or **Neural Networks** for short.

2.1. Neural Networks Foundations Basics

The idea of having a machine mimicking the way a human brain solves some real-life tasks motivated the creation of biological brain-inspired ML models, such as **Neural Networks**. In a very simplified way, a biological brain, more specifically the nervous system, encloses a very large set of basic processing units, the so-called neurons, highly interconnected with each other through the so-called synapses. In the same way, an NN is a computing system that may also enclose a high number of **artificial neurons** connected with each other. However, the artificial neuron mathematical representation, illustrated in Figure 2 (right), is rather simpler when compared to representation of the biological neuron. Figure 2 provides a side by side visual comparison of biological and artificial neurons.


impulses carried toward cell body
dendrites
branches of axon
nucleus
axon
axon terminals
impulses carried away from cell body
cell body

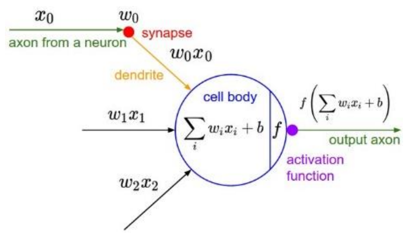


Figure 2 - Representation of a biological neuron (left) and artificial neuron (right) [12].

As shown in Figure 2, each artificial neuron receives a set of **inputs** (x_i) and associates to each of them a **weight** (w_i) that indicates the influence of said input to the neuron's **output** (y). These weights are learnable and can be positive or negative, exhibiting an excitatory or inhibitory effect of one neuron to another, respectively. The product $x_i.w_i$ is a simplified representation of a synapse and the weight w_i corresponds to the synaptic strength. Another important concept in NNs is the **activation function,** which defines the non-linear response of a neuron given the weighted sum of its inputs and a **bias** (b). The bias is another learnable parameter of the neuron, which shifts the activation function along the x-axis to a desired triggering value. Mathematically speaking, the overall behaviour of an artificial neuron is represented in (1).

$$y = f\left(b + \sum_i x_i\, w_i\right) \tag{1}$$

Figure 3 shows some of the activation functions typically used in NNs, such as the **Sigmoid**, the **Hyperbolic Tangent** (Tanh) and the **Rectified Linear Unit** (ReLU) functions. These non-linear functions allow simulating behaviours more complex than the simple linear one, and this way, more powerful NNs can be defined to solve a given task.

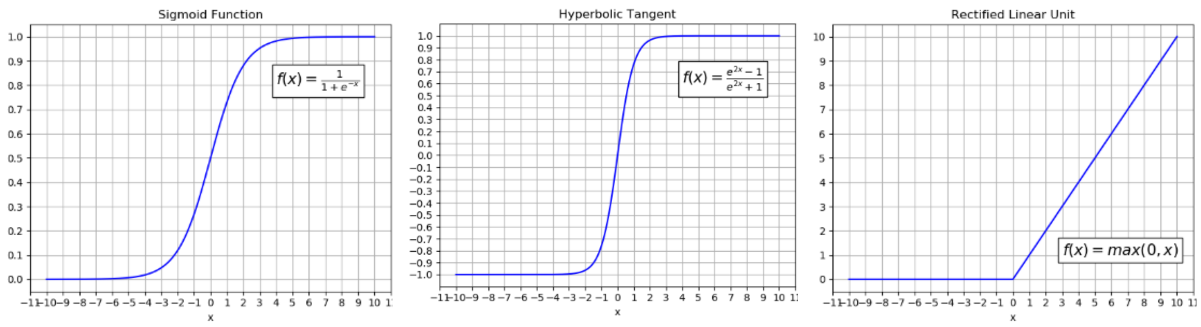


Figure 3 – Graphical representation of Sigmoid (left), Tanh (centre) and ReLU (right).

In a NN the neurons are organized into **layers**, as illustrated in Figure 4; within a layer, all neurons are connected in the same way and process the received data using the same non-linear activation function. While the first and last layers in a NN are called **input layer** and **output layer**, respectively, the layers in between, which may or may not exist, are called **hidden layers**. The role of the input layer is to feed the network with the input data, without any processing involved, thus being the starting layer in any NN architecture. Hidden layers are the heart and soul of any NN since they are responsible for the processing of information through the combination of the data received from the previous (input or hidden) layer, thus allowing to learn different representation levels from the input data. The number of hidden layers $(0, 1, \ldots, N)$ in a NN typically increases with the complexity of the problem at hand. The last layer of a NN's architecture, i.e. output layer, is, as the name suggests, responsible to output the solution to the problem the NN was designed for. The output layer is typically a **fully connected** (FC) layer, which means that every neuron on that layer is connected to all the neurons in the previous layer (see Figure 4); FC layer is, in fact, the most common layer type used in regular NNs. The number of neurons in a layer is typically called layer **size** or layer **height** and the number of layers in a NN is called **depth**; the higher the number of layers in a NN, the deeper the NN is and, consequently, the higher is the power of the NN to solve the problem at hand.

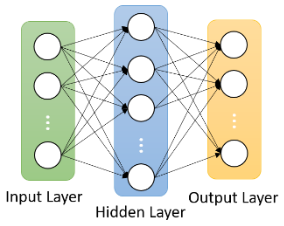


Figure 4 – Example of a fully connected neural network architecture with two layers.

Interpreting a NN as a graph, where the nodes correspond to neurons, the NN can be classified as cyclic or acyclic depending on whether the connections between nodes (i.e. neurons) form loops or not, respectively. The popular **Feedforward Neural Networks** are an example of acyclic NNs, while the **Recurrent Neural Networks** are an example of cyclic NNs. Recurrent neural networks are, due to its architecture, very good at finding patterns in sequences of inputs and have many applications (grammar learning, time series prediction, etc.). On the other hand, Feedforward Neural Networks do not have this notion of sequence but are excellent at handling image processing and other similar problems.

The architecture of a NN depends on the target application and is characterized by a set of **hyperparameters** that, unlike weighs and bias, are not learnable but rather set to a fixed value before training. Some examples of hyperparameters are the depth and height of the network and the learning rate; while the former two hyperparameters define the NN structure, the latter one establishes how the network is trained. In the context of a NN, the hyperparameters strongly influence its performance, and therefore they should be carefully set.

2.2. Model Learning

The **model learning** process is a crucial part of the creation of a functional NN. This process, also known as NN model training, is responsible for determining the weights and biases of all neuros in the network using some training data (training samples). The training samples are extracted from the so-called **training dataset** and are used to train the NN in order to perform the task is intended for.

Learning methods can be divided into two main categories: unsupervised and supervised. In

supervised learning, the training data is labelled, this means that the training samples are pairs of inputs-outputs. The labels (i.e. ground truth) provide a guideline to the learning process. On the other hand, in **unsupervised learning** no target output values, i.e. ground truth, are given, which means that only the training samples are used to learn and model the training dataset's underlying structure.

Although unsupervised learning can be applied in the context of NN, the most common approach to train the NN model is supervised learning. In supervised learning algorithms, the goal is to determine the network model from the training dataset by learning to minimize the error between the predictions made by the network and the ground truth (target output associated to the training samples). In this context, a **loss function** is used to measure the prediction error, which may vary depending on the problem at hand. Some examples of frequently used loss functions are the mean square error, the mean absolute error, and the hinge loss.

The **gradient descent** is a widely used supervised learning iterative algorithm that, in its simplest form, can be described by the flowchart depicted in Figure 5. Briefly, the gradient descent algorithm finds, in an iterative way, the NN model parameters (i.e. weights and biases) that minimize the loss function.

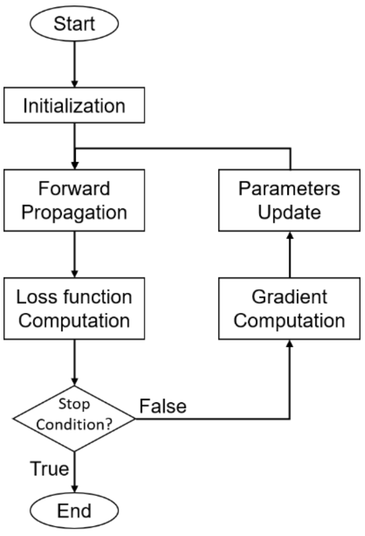


Figure 5 - Simplified gradient descent algorithm flowchart.

In more detail, the steps of this algorithm can be described as:

- **Initialization:** The algorithm attributes values to all the weights and biases of the NN. Typically, these initial values are randomly picked to create some asymmetry in the NN, and consequentially make each neuron have a different starting point in the learning process.
- **Forward propagation:** The NN makes predictions with the available input training data by propagating it forward through all the network layers. The outputs generated by the NN in the first iterations of the gradient descent algorithm are most likely far from the target outputs. However, it will provide valuable information that can guide the algorithm in future steps.
- **Loss function computation:** Computes the loss function between the predicted and the target

values. This loss function assesses the performance of the network for its current weights and biases.

- **Stop Condition:** The value obtained for the loss function is evaluated at each iteration and can be used as a stop condition for the algorithm. However, other factors can be used as stop conditions, such as the maximum number of iterations. When one of the stopping criteria is met the algorithm terminates, and the current weights and biases are the result of the learning process, defining the network model.
- **Gradient Computation:** In this step starts the refinement of the parameters. This is possible with the computation of the gradient of the loss function with respect to the model parameters. In general, the gradient is an operator that given a certain function as input, outputs the corresponding direction and rate in which the function's increase is larger. Mathematically speaking, the gradient is a vector of partial derivatives of the function, as define in (2). Since the main goal of the gradient descent algorithm is to learn the network model parameters that minimize the loss function, the computation of the gradient indicates in which direction and with which speed/rate the model parameters should be changed such that the prediction error is minimized.

$$\nabla\mathcal{L} = \begin{bmatrix} \frac{\partial}{\partial w_1}\mathcal{L} \\ \vdots \\ \frac{\partial}{\partial w_j}\mathcal{L} \end{bmatrix} \tag{2}$$

In order to compute the gradient of the loss function with respect to every model parameter, the **backpropagation** algorithm is used. This algorithm recursively applies the chain rule making the loss gradient with respect to a given weight or bias only dependent on the output value of the neuron associated with said parameter and the derivative of the inputs with respect to the output value. Naturally, for this to happen, the derivatives of the activation functions used in the NN have to be known. Since the objective in this step is to calculate the gradient of the loss function, the loss gradient with respect to the NN output is first computed and this result is then propagated backwards through all the network layers.

- **Parameters update:** This operation depends on the previous value of the weight or bias , $w_j^{(i)}$, the partial derivative of the loss function with respect to the parameter to be updated, $\frac{\partial}{\partial w_j^{(i)}}\mathcal{L}^{(i)}$, and the learning rate, η. The learning rate is a hyperparameter that scales the size of the gradient step given in the weights updating for the loss function to decrease and is typically set manually after some experimentation. The weights updating is performed as defined in (3), where $w_j^{(i+1)}$ is the updated weight in iteration $(i+1)$.

$$w_j^{(i+1)} = w_j^{(i)} - \eta\ \frac{\partial}{\partial w_j^{(i)}}\mathcal{L}^{(i)} \tag{3}$$

When all the weights of the NN are updated, the iteration is completed and the algorithm initiates a new iteration, starting with the forward propagation with the new parameters.

The convergence of the gradient descent is not guaranteed, as illustrated in Figure 6, and depends heavily on the learning rate. When a poor choice of the learning rate is made, the algorithm might never achieve de expected results, diverging. The divergence risk can be reduced by using techniques, such as, the momentum term or the adaptive step size [13]

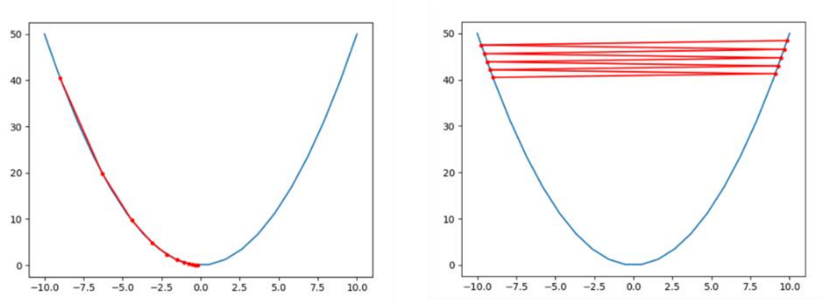


Figure 6 - Example of gradient descent converging (left) and diverging (right).

The most straightforward variant of the gradient descent algorithm, the so-called **Batch Gradient Descent** or **Vanilla Gradient Descent**, uses all the training data in each iteration to update the model parameters. However, depending on the number of samples in the training dataset, it might be useful to update the model parameters for a set of training samples or even for each training sample, as the amount of training data used, in each iteration, to update the model parameters has a direct impact on the computational resources consumption (e.g. memory). While the former variant is commonly known as **Mini-Batch Gradient Descent**, the latter is known as **Stochastic Gradient Descent** and presents a huge performance improvement when compared with the **Batch Gradient Descent**, especially when the training dataset is rather redundant.

After the model learning process is concluded, the NN is ready to be used in realistic conditions, i.e. using input samples that the NN has never seen before (notably during the training stage), the so-called **test samples**, thus allowing to evaluate the real NN's performance in the accomplishment of the task it has been designed for. Depending on how the network model is learned, the NN might become too specialized in the training dataset resulting in **overfitting**; when this happens, the model's generalization capability for unseen (test) data tends to be low, which results in poor performance. In order to guarantee that the learning process does not lead to an overfitted model, **regularization** techniques must be employed. Some examples of regularization techniques are the early stopping [14] and dropout [15].

2.3. Main Neural Network Types

Nowadays, there is a large variety of Neural Networks with different types of layers, topologies, and purposes. Considering the thesis focus on the application of neural networks (more specifically GANs) to some computer vision problems, this section is dedicated to reviewing the most relevant types of NN in the field. The main types identified were Convolutional Neural Networks (Section 2.3.1), Autoencoder

(Section 2.3.2), and GANs (Section 2.3.3).

2.3.1. Convolutional Neural Networks

Using neural networks composed only of fully connected layers for image processing tasks is far from ideal since the model would have a large number of parameters, making the training process slow and the network prone to overfitting. This problem motivated the creation of convolutional neural networks (CNN). CNN are feedforward neural networks that, contrarily to regular neural networks, take advantage of the spatial dependence of its inputs leading to more efficient feature extraction and fewer parameters. The architecture of a CNN may vary greatly depending on the type of problem at hand, but typically there are two key building blocks: **convolutional layers** and **pooling layers.**

Due to its properties CNN are commonly applied in problems of image classification, natural language processing [16], computer vision [17], among others. In Figure 7 is illustrated the architecture of a CNN designed for image classification and is divided into two functional parts: feature learning and classification. The convolutional and pooling layers are used in the feature extraction portion of the network and exhibit a hierarchical organization. This means that layers closer to the input will identify lower-level features, for example, edges or dark spots, and layers further away from the input will identify higher-level features, with more semantic meaning to that specific problem. In other words, deeper NN will have higher feature extraction capabilities. [18]

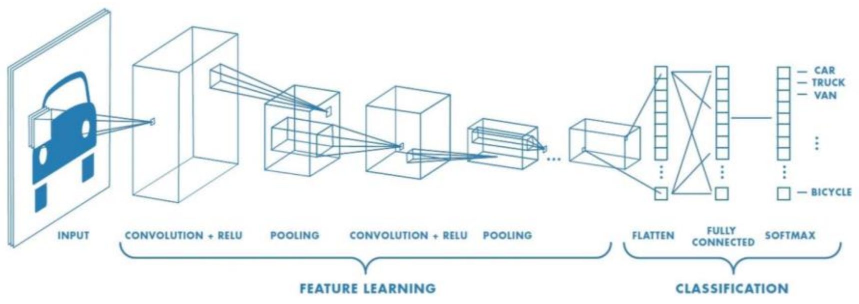


Figure 7 - Architecture of a CNN used for image classification [19].

The classification part of the CNN has a set of fully connected layers since the desired output, in this case, is a probability distribution. In other applications, this functional part of the network varies depending on the desired output.

2.3.1.1. Convolutional Layers

Convolutional layers, as its name suggests, are the main elements of this type of NN. These layers are composed of a set of **filters** or **kernels**. A kernel is an array of weights, learned through the typical training process of a NN, that represent some characteristic of the data relevant to the NN. These filters can be 1,2 or 3-dimensional arrays depending on the input's dimension. In this type of layer, the output is computed by sliding the kernels over the input data and performing a dot product. Which makes the model insensitive to translations in the data in the feature extraction process. This operation is also known as a convolution and is represented with a very simple example in Figure 8. The result of the convolution is summed a bias, and to that sum is applied some activation function (typically the ReLU).

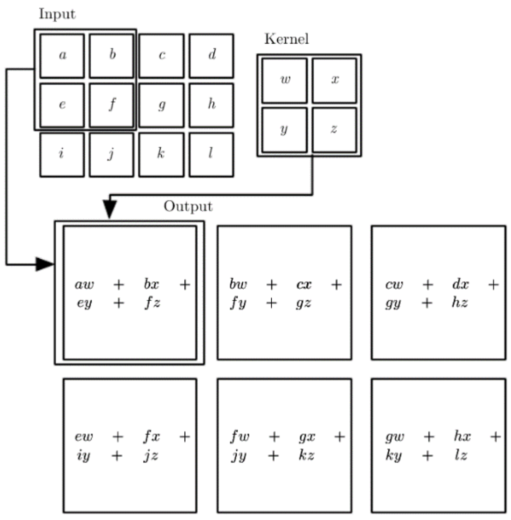


Figure 8 - Example of a 2-D convolution with stride equal to 1 and without padding [20].

The number of filters used by a certain layer is a hyperparameter and is adjusted having into account the complexity of the tasks delegated to that layer. This hyperparameter will influence the size of the layer's output, more specifically, will affect the depth of the output volume.

Other hyperparameters that affect the output size, and the number of neurons in the layer, is the **stride** (S), the **size of the filters** (F) and the amount of **padding** (P). The stride controls the sliding of the filter that the stride is to equal the number of pixels jumped. Sometimes the desired stride and kernel size don't fit the input data size nicely, as shown in Figure 8, thus the need to use padding. Padding consists of adding to the input an extra border of pixels. There are several types of padding, like zero-padding that consists of adding a border of zeros to the data or the reflection padding, where the data added is the result of reflecting the input image about the border, among others. This technique is used to control the output size without changing the behavior of the layer.

The different paradigm of operation of these NN leads to a 3-dimensional arrangement of neurons, as shown in Figure 9.

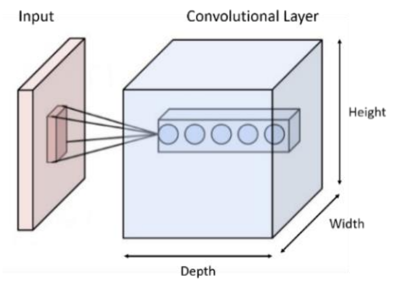


Figure 9 - Simplified representation of the neurons' organization in a convolutional layer.

Each neuron is associated with a given feature (filter) and a given set of inputs. With the filter sliding along the input, the same weights are shared between different neurons, promoting the reduction of parameters. That is to say that neurons in the same vertical cut of the layer (**depth slice**) share the same kernel. Likewise, neurons in the same horizontal cut (**depth column**) share the same input area (receptive field**)**, as highlighted in Figure 9.

2.3.1.2. Pooling Layers

In a CNN, the chain of convolutional layers, responsible for feature extraction, is usually intertwined with pooling layers. The pooling layers follow the convolutional layers and serve the purpose of downsampling the output of the previous convolutional layer, as illustrate in Figure 10. The need to downsample the output of the convolutional layers comes from the fact that chaining convolutional layers greatly increases the number of parameters. These pooling layers work with sliding filters, like the convolutional layers. These filters have a function associated with them in order to transform the input of the filter into a single output. The most common pooling layer is the max-pooling layer, that as the name suggests, returns the maximum value of the inputs.

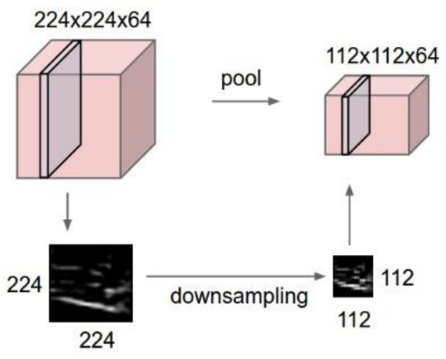


Figure 10 - Illustration of the input-output relationship in a pooling layer with filter size 2x2 and stride 2 [12].

The pooling layer makes the model lose information that may or may not be relevant for the NN. This motivates some architectures that don’t use pooling layers, relying on other techniques to oppose the increase in parameters [21].

2.3.2. Autoencoders

Autoencoders (AE) are a type of unsupervised learning algorithm. More specifically are designed to have the output approximate the input with some constrains or restrictions, most often requiring the input to be represented also with lower dimensionality. These restrictions built into the neural network allow to represent the input data in some compressed way (as the name auto-encoder suggests) and in this process learn the most important features of the input data. This compressed representation (also called **code**) lies on some **latent space** [20].

As shown in Figure 11, an AE is composed of an encoder that maps the input to its latent space, followed by a decoder that attempts to revert the data to its original form using the code provided by the encoder.

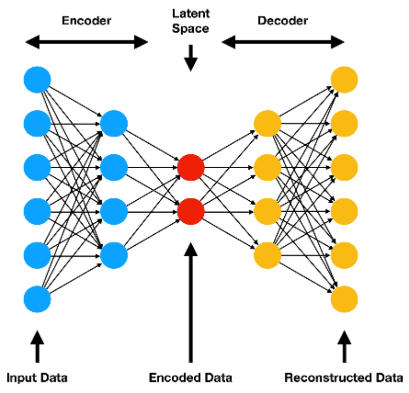


Figure 11 - Architecture of an autoencoder [22] with encoder.

The encoding performed in an AE is lossy, meaning that output of the network will never be exactly the same as the input. During the training of the auto-encoder, the encoder and decoder work together trying to recreate as closely as possible the input but with certain restrictions (typically in the architecture design) to prevent a simple copy of the data along the network.

This learning process is performed by minimizing a loss function as described in Section 2.2. The AE's performance heavily dependent on the type of data that is trained on. That is to say that the AE is a data specific solution.

The layer in which the data is represented in a compressed way, i.e. the latent space representation is usually called the bottleneck layer. The number of neurons in the bottleneck can be used to categorize different types of AE, namely, an AE is said **undercomplete** if the dimension (usually the number of neurons) of the bottleneck is smaller than the number of neurons in the input layer and **overcomplete** otherwise. In an undercomplete autoencoder, the restriction that prevents the NN to output a copy of its input is the reduced bottleneck layer size. In an overcomplete AE, the restriction is different, most often a sparsity regularization on the responses of bottleneck layer (e.g. forcing the output of some neurons to zero).

2.3.2.1. Sparse Autoencoders

The Sparse Autoencoder (SAE) is an overcomplete AE that relies on sparsity penalties to enforce the good generalization of the model. The structure of an SAE allows for a large number of neurons in the bottleneck layer, but the computation of the loss function includes some penalties on the output values of this layer. For example, the target may correspond to have very small activations at the output of the bottleneck layer. Thus, a term is added to the cost function which increases its cost if the average activation value (over all the training samples) is not close to zero. This is called L1 regularization and the loss function penalizes the absolute value of the activations.

Other possibility is to use a sparsity regularization that can be imposed by adding a Kullback-Leibler

(KL) divergence term to the loss function computation. In general, the KL divergence is a function that measures the similarities between two probability distributions. In the context of the AE sparsity regularization, the KL divergence is used to infer if the number of active neurons in the bottleneck layer is close to what is expected. This guides the learning process to obtain a compact code in the bottleneck layer.

2.3.2.2. Denoising Autoencoders

In the Denoising Autoencoder (DAE) the training data is slightly corrupted before being fed to the neural network but still maintain the original (uncorrupted data) as the target output. This makes the overcomplete autoencoder learn a good data representation, i.e. a model is obtained that is generalizable, since input (with noise) and target output are no longer the same. Naturally, it is important to guarantee that by adding noise the data is not degraded to the point of making the DAE uncapable of recovering the input. In Figure 12 is shown the typical architecture of this type of models.

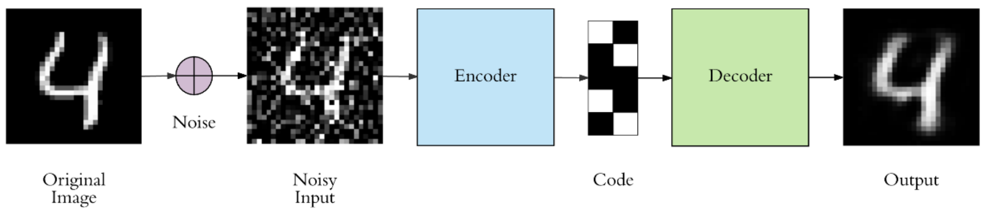


Figure 12 - Block diagram of the general architecture of a Denoising Autoencoder [23].

2.3.2.3. Variational Autoencoders

The variational autoencoders (VAE) [24, 25] encode and decode as the previously described AEs, but the code (in the latent space) elements do not correspond to a single value but to a probability distribution (in general Gaussian Distributions). The structure of an VAE is depicted in Figure 13. As shown, the latent space representation of the data is composed of two vectors: the mean value (μ) and the standard deviation (σ). This is a change in paradigm from the previously mentioned autoencoders since the variational autoencoder (VAE) describes the input in the latent space in a probabilistic way and the others don’t. Where the DAE and the SAE are considered discriminative models, the VAE is considered a generative model. The VAE creates a powerful representation of the data in the latent space which can be sampled, and the decoder network is capable of create new data similar to what was observed during training.

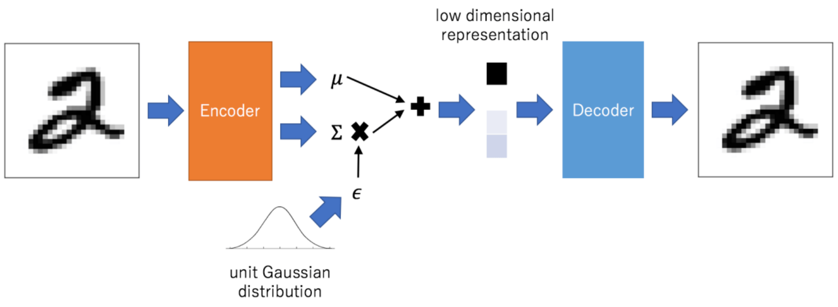


Figure 13 - Representation of the typical architecture of a Variational Autoencoder [26].

The training of these neural networks is similar to other AE in the sense that the minimization of the loss function is the main goal. However, in variational auto-encoders it is also used the KL divergence on the bottleneck layer to ensure that the distributions at the bottleneck layer are close to a normal distribution. This additional constraint in the loss function, forces the encoder to generate a vector of means and a vector of standard deviations.

2.3.3. Generative Adversarial Network

Generative Adversarial Autoencoders are a type of generative model proposed in 2014 by Ian Goodfellow *et al.* [7]. GANs have several applications, such as generate photographs [27], text-to-image synthesis [28], image-to-image translation [29], compression [30], video prediction [31], among others.

This model, unlike other generative models (e.g. the VAE described in the previous section), does not estimates explicitly a probability distribution but learns it implicitly during training from examples. GANs are composed of two separable neural networks, the **Generator** ($G(x)$) and the **Discriminator** ($D(x)$) and exploit a game-theory approach where the two NNs compete against each other as adversaries. Moreover, the goal of the Generator is to create synthetic (fake) data that resembles as closely as possible to real data and thus, fool the Discriminator, whose goal is to distinguish between fake data and real data.

The GAN architecture is shown in Figure 14. In practice, the Generator receives samples from a simple distribution (e.g. random noise) which is processed by a sequence of neural network layers, and the output corresponds to data that resembles the data from the training set at least in a semantic way. However, the Generator is blind to the training dataset, being dependent on the Discriminator to guide its learning process. The Discriminator receives data samples and outputs probabilities that represents if the sample belongs to the training dataset or not. This means that the discriminator evaluates the authenticity of images or how close the images produced by the generator are close the images of the training dataset. When the generator reaches a high level of performance (this means produces realistic looking images), the Discriminator is unable to distinguish between real and fake data, which means $D(x) = 0.5$ for every sample. Thus, Generator goal is to produce images that are considered realistic (i.e. act as a forger without being caught) and the discriminator goal is to assess if these images are fake or not (i.e. act as the police to detect forgeries).

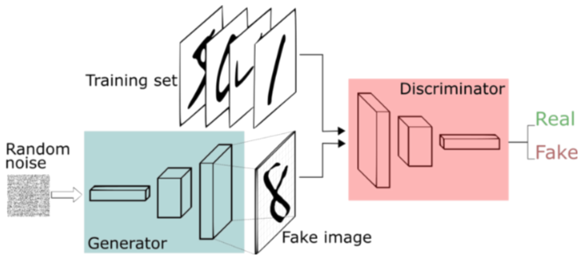


Figure 14 – Architecture of a Generative Adversarial Network.

The random noise is the input of the generator that drives the creation of fake image. This fake image and the training set (real) images are fed to the Discriminator, that classifies each image as being either fake or real [32]. This design of a GAN imposes that the cost function (4) to be a minimax game, where the Discriminator try to maximize the cost and the generator to minimize it. In (4), $D(x)$ is the discriminator output for real data of the training set (i.e. probability that some real data sample is real) and $D(1 - G(z))$ is the discriminator output for fake data (i.e. the probability that a fake instance is real). In (4), $p_{data}(x)$ corresponds to the distribution of probabilities of the training dataset and $p_z(z)$ is the random noise distribution. Furthermore, $\mathbb{E}_{x \sim p_{data}(x)}[\log D(x)]$ is the expectation over all the samples belonging to the training set and $\mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z))]$ corresponds to the expectation over all the samples produced by the Generator. This means that a discriminator model needs to be found that maximizes (this means recognizes) real data and fake data, while a generator model also needs to be found to generate realistic fake data.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z))] \tag{4}$$

During the learning process this cost function will be used and, ideally, the Generator will reach $p_g = p_{data}$, where p_g is the implicitly learned probability distribution, i.e. the generator can fool the discriminator. The learning process, as depicted in Figure 15, can be described by the following steps:

1. Obtain a minibatch of size m from both the training dataset $(x^{(1)} \dots x^{(m)})$ and the random noise source $(z^{(1)} \dots z^{(m)})$.
2. Compute the gradient ascent, as shown in (5), since the goal of the Discriminator is to maximize the cost function.

$$\Delta_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log\left(D\left(x^{(i)}\right)\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right] \tag{5}$$

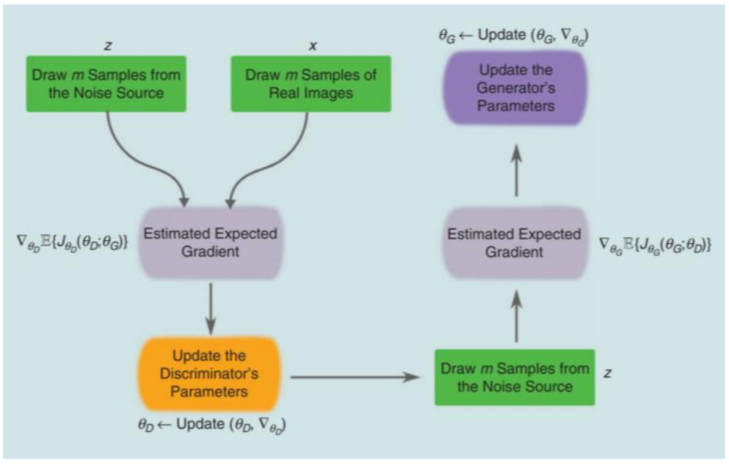
3. Update the discriminator's parameters with the computed gradient.
4. Sample another minibatch of size m of random noise samples $(z^{(1)} \dots z^{(m)})$.
5. Compute the gradient descent, as shown in (6). As aforementioned, the generator is not directly exposed to the training dataset samples, and as such, the gradient computation is done

independently of $(x^{(1)} \dots x^{(m)})$.

$$\Delta_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \left[\log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right] \tag{6}$$

6. Update the generator's parameters with the computed gradient.

These steps (1 through 6) correspond to one iteration of training. The number of training iterations done in the learning process depends on the stop condition imposed, as mentioned in Section 2.2. In each training iteration, steps 1 through 3 are repeated k times (hyperparameter).



z
Draw m Samples from the Noise Source
x
Draw m Samples of Real Images
$\theta_G \leftarrow$ Update $(\theta_G, \nabla_{\theta_G})$
Update the Generator's Parameters
$\nabla_{\theta_D}\mathbb{E}\{J_{\theta_D}(\theta_D;\theta_G)\}$
Estimated Expected Gradient
Estimated Expected Gradient
$\nabla_{\theta_G}\mathbb{E}\{J_{\theta_G}(\theta_G;\theta_D)\}$
Update the Discriminator's Parameters
$\theta_D \leftarrow$ Update $(\theta_D, \nabla_{\theta_D})$
Draw m Samples from the Noise Source
z

Figure 15 – Main loop of the training process of a GAN [33].

At the beginning of the training process, the generator produces images that don't resemble the real data. Consequently, the discriminator is not fooled, which means $D(G(z))$ is close to 0. With the formulation presented in (4), this leads to a problem of vanishing gradients when computing the gradient descent needed to update the generator's parameters. Usually, alternative formulations are used to prevent this. A common practice is maximizing to the probability of the Discriminator being wrong $\left(-\log\left(D(G(z)\right)\right)$ instead of minimizing the probability of the Discriminator being right for generator data $\left(\log\left(1 - D(G(z)\right)\right)$. This formulation leads to the same solution as the previous, but, as depicted in Figure 16, produces higher gradient signals in early training [7, 33].

It's also important to note that even though this minimax game has a theoretical solution in practice the convergence of this model might not be achieved if the training is not properly made [34].

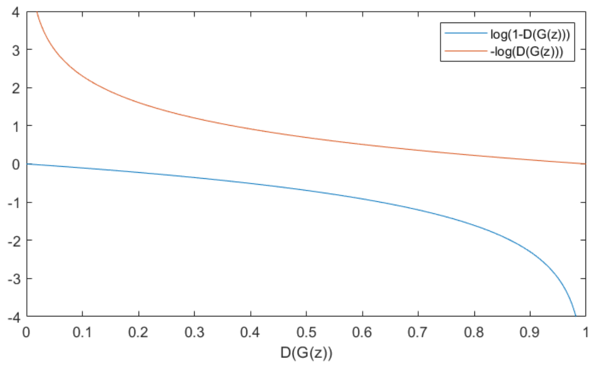


Figure 16 – Comparison between $\log{(1 - D\big(G(z)\big)}$ and $-\log{(D\big(G(z)\big)}$, illustrating the saturation of the $\log{(1 - D\big(G(z)\big)}$ curve for small values of $D\big(G(z)\big)$.

Chapter 3

3. Relevant GAN-Based Image Processing Solutions

In this chapter, it is described the application of GAN-based solutions to some selected relevant image processing problems, such as image compression, super-resolution, and artifact removal. The selected solutions were selected due to its performance and overall popularity and will serve as a starting point to achieving this Thesis goals.

For each solution, a brief contextualization of the problem itself is made followed by the detailed description of the entire solution including reported experimental results. The review of each solution is divided into the following 4 parts:

A. Identification of the objectives and technical approach.
B. Overview of the architecture and walkthrough.
C. Identification and review and the key tools used.
D. Performance reporting and analysis, comparing the solution to other relevant state-of-art.

3.1. Generative Adversarial Networks for Extreme Learned Image Compression

Nowadays, deep neural networks have become a feasible and promising solution for the image compression problem [35]. Image codecs based on neural networks offer competitive results, when compared to conventional image compression solutions, such as WebP [36], JPEG2000 [1] or BPG [37]. The most distinctive aspect in neural network image compression is that the transform (and sometimes the entropy coding model) is end-to-end learned with a large amount of data and is not hand-crafted as traditional image codecs.

This solution targets extremely low bitrates, where is hard to preserve many visual elements of an image with high fidelity. The solution described in this section proposes to generate artificial elements and can be classified as extreme compression, where the pixel-wise preservation becomes less important when compared to the global structure and the semantic meaning of the image.

A. Objective and Technical Approach

Agustsson *et al.* proposed two GAN based frameworks for image compression [30] which target bitrates lower than 0.1 bpp. In this context, the use of GANs tries to deliver more appealing images by implicitly learning a suitable latent space representation (i.e. the natural image manifold). The proposed solutions optimize the image coding process beyond the usual conventional image quality metrics used in the training process, with a loss function that includes an adversarial loss term. This framework has two operation modes, namely:

- **Generative Compression** (GC): This mode of operation exploits the generative capabilities of GANs to aid in image compression, preserving the content as much as possible. This solution is similar to the vanilla GAN described in Section 2.3.3 but introduces a new loss function and instead of sampling from a random noise distribution, the generator (decoder) uses the encoded image as its input.
- **Selective Generative Compression** (SC): This mode of operation preserves some elements of the original image with high fidelity and other elements are only generated based on semantic information about these elements. The SC uses some additional information about the image, namely, its semantic label map, which can be seen as a Conditional GAN (cGAN) [38]. This solution is particularly suitable in scenarios where a part of the image is perceptually very relevant, while other parts are less relevant.

Regarding these two modes, the SC mode requires additional information, which is not available in the GC mode, namely the input image segmented with regions with a semantic meaning and some additional indication which regions should be preserved and which can be artificially generated (a binary heatmap).

B. Architecture and Walkthrough

The two modes of operation in this framework are based on GANs and obtain decoded images where texture and other elements of the image are fully synthesized without receiving any information from the encoder about these elements. In this context, distortion (or quality) metrics such as PSNR and MS-SSIM cannot account how real the image is, since local changes in structure of the image are severe penalized and thus, cannot evaluate how the global structure of the image is preserved.

<u>**Generative Compression**</u>

Figure 17 shows the block diagram architecture of the proposed solution GC mode. The input (x), with dimensions $W \times H$, is feed to the **Encoder** (E) which convolutionally processes the input into feature maps with lower spatial resolution (16 down-sampling factor). The down-sampling is performed with strided convolutions. This results in a feature map of dimension $\frac{W}{16} \times \frac{H}{16} \times C$, where C is the number of activation maps. The output of the encoder (w) passes through a **Quantizer** (q) which outputs a discretization of the feature map $(\hat{w})$, with a pre-defined number of levels.

The following blocks, **Generator** (G) and **Discriminator** (D) behave as a classic GAN. This means that the generator tries to reconstruct the image (x) using the $(\hat{w})$ feature map and the discriminator attempts to distinguish the fake images created by the generator $(\hat{x})$ and the real images. However, the loss function guiding the training process is not the typical GAN loss presented in Section 2.3.3. The GC's loss function includes the standard GAN loss and one additional term that includes the distortion between original and decoded images and the bitrate. This loss function allows to balance the artificially generated content with the preservation of the original image.

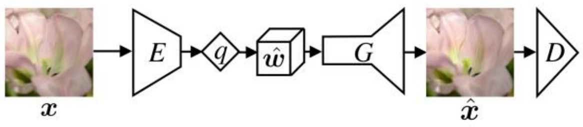


Figure 17 - Architecture of the proposed GC network [30].

Selective Generative Compression

The architecture of the SC mode is presented in Figure 18 and shares some functional blocks with the GC architecture. In this sense, the behaviour of the encoder and quantizer is the same. However, unlike the GC, this solution requires:

- **Semantic label map** (s): provides additional semantic information with a segmentation of the images into regions and the classification of each one (label map). This information is very important for the generator to distinguish the regions in which it must perform full generation and the regions where it should preserve the original content.
- **Heatmap** (m): is a binary map with equal dimensions as the feature map $(\hat{w})$ that indicates which elements should be preserved with higher fidelity and which elements can be fully generated. This heatmap is typically defined by the user.

As shown in Figure 18, the semantic label map is passed to the **feature extractor** (F) that process and feeds it to the generator. In a real scenario, the semantic map s is lossless coded using some suitable solution and thus is available at both encoder and decoder. The heatmap (m) is used to mask the feature map $(\hat{w})$ before being transmitted to the generator (at the decoder side). This step defines which regions should be preserved and which regions should be fully synthesized using only the semantic map information.

The SC solution depends on the semantic label map which is required not only during training but also during compression. The semantic label map can be obtained through an image semantic segmentation network, such as PSPNet [39] or Mask R-CNN [40], for example.

Regarding training, two modes are considered: Random Instance (RI) and Random Box (RB). In RI some instances (regions) of the semantic label map are picked randomly to be preserved. In RB a square region of the image is selected in a random way, which is also preserved. This last one is more challenging since it is difficult to integrate synthetic and preserved content in a seamless way using a square region.

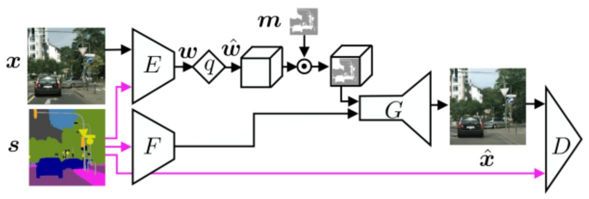


Figure 18 - Proposed SC network architecture [41].

C. Main Tools

Some of the key tools introduced in this image coding framework are:

- **New training objective:** The introduction of a new loss function in the GC and SC models makes for more balanced models, in the sense that there is a trade-off between preserving the overall structure of the image and its pixel-wise similarities with the original. Moreover, the GC's loss function (7) is composed of the typical GAN loss function $\mathcal{L}_{GAN}$, presented in Section 2.3.3, and a distortion plus rate term. This term is $(\lambda\mathbb{E}[d(x, G(z))] + \beta H(\hat{w}))$ and allows to obtain better reconstruction quality with a small amount of bitrate. The function d measures the reconstruction error of the generator's output. This function can be, for example, a simple MSE (Mean Squared Error) or a more complex MS-SSIM [42]. The λ $(\lambda > 0)$ parameter is used to balance the relative importance of the pixel-wise distortion. The parameter β ($\beta \in [0, 1]$) is used to limit the bitrate, however in this application the bitrate can be limited by design (quantization) to avoid having a loss function dependent on an entropy term ($H(\hat{w})$).

$$\min_{E,G} \mathcal{L}_{GAN} + \lambda\mathbb{E}[d(x, G(z))] + \beta H(\hat{w}) \tag{7}$$

- **Improved discriminator with segmentation data:** An extension of the SC mode is also proposed, that consists of feeding the discriminator with a semantic label map (s) ($GD(D^+)$). This dependence with s only exists during training, i.e., the actual $GC(D^+)$ image compression operation (inference) does not require the semantic label map. Naturally, it is also used in the SC mode. The use of semantic label maps at the discriminator side allows to improve the discriminator performance.

D. Performance Assessment

To assess the proposed solution performance, the GC model was trained without semantic label maps. The training dataset is composed of several natural images, namely 188k images from the Open Images

[43] dataset. After training, the model was evaluated on the Kodak [44], RAISE-1k [45] and Cityscapes [46] datasets.

Figure 19 shows an example of the proposed GC mode output for an image in the Cityscape dataset. In this example, GC is compared to the BPG codec (an implementation of HEVC) and to the MSE baseline. The MSE baseline corresponds to a solution similar to GC but instead of having the adversarial loss guiding the learning process it has a simple MSE loss. This means, the discriminator is not present in the MSE baseline and that MSE baseline was trained on a subset of the Cityscapes dataset which gives an advantage in specialization when compared to the GC model.

To evaluate the performance for an objective quality metric, the PSNR was still used despite the limitations. The GC model is outperformed by its benchmarks with respect to PSNR. This was expected since PSNR favors pixel-wise similarities and the GC model was not trained with that goal, unlike the other BPG and MSE baseline solutions. However, when it comes to the subjective quality of the image, the GC model is superior, obtaining a sharper and more detailed overall image as shown. This highlights the inadequacy of metrics such as PSNR to evaluate image quality.

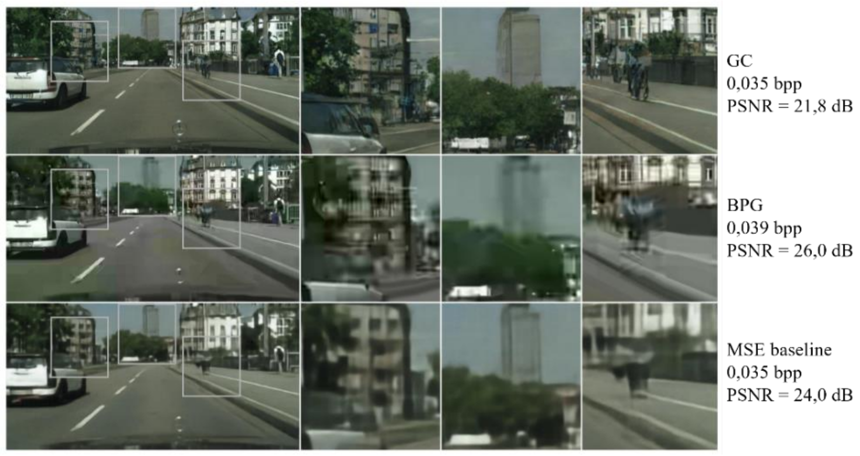


Figure 19 - Evaluation between GC, BPG and MSE baseline using an image from the Cityscapes dataset (Adapted from [30]).

As aforementioned, the GC model was also tested on the Kodak and RAISE-1k datasets. Figure 20 shows a comparison between an original image, the GC model and BPG codec, for each of these datasets. Regarding the subjective quality of the compressed images, the BPG's images are overall blurrier, lacking texture and detail due to the low bitrate and the limitations of this conventional solution. On the other hand, the GC model can generate more realistic textures and maintain sharper details. However, the GC solution has some visible shortcomings, e.g., in one of the Kodak images (Figure 20 - Top), some windows have annoying artefacts that contribute to an unnatural appearance.

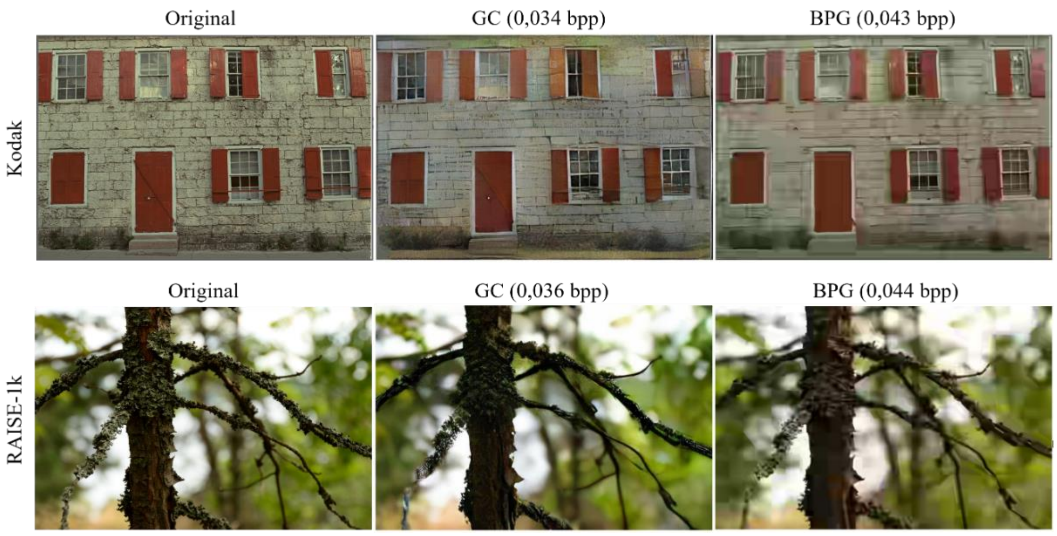


Figure 20 - Visual comparison between original image (left), GC (middle) and BPG (right) using images from different datasets: Kodak (top) and RAISE 1k (bottom) (Adapted from [41, 44, 45]).

The SC solution was trained in the Cityscapes dataset using the RI and RB training modes. For the RB training, the model was able to combine the preserved and generated (synthetic) contents even in cases where an object is a combination of preserved and generated content (i.e. the box limits cut the object), as depicted in Figure 21. In this figure, it is presented two different base images, and for each image is showcased two different RB scenarios (box location changes). Moreover, the lower left corner of the images is the heatmap of each scenario, where the white region corresponds to the RB.

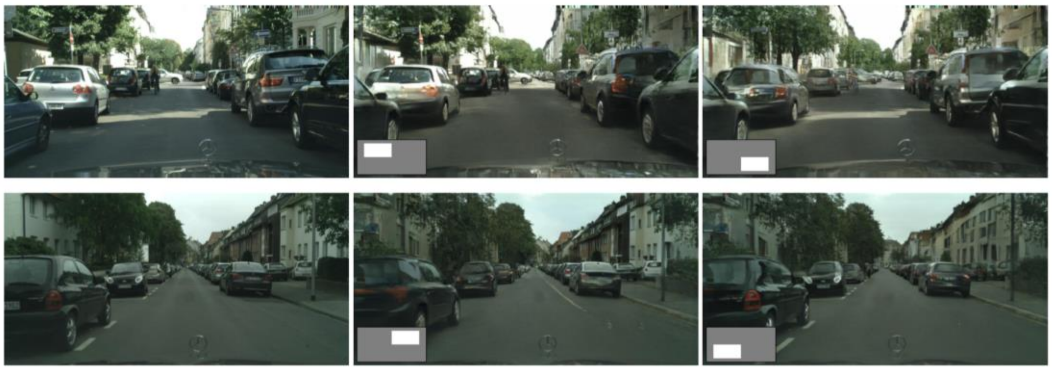


Figure 21 - Example of visual comparison for different random boxes using the SC model with RB training [41].

Figure 22 shows the visual quality (and bitrate savings) when the SC model is used with RI training and different elements to preserve are selected. In the lower left corner of the images is the heatmap of each scenario, where the white colored regions corresponds to the RI. It is also shown the bpp for each image and the relative savings due to the selective generation when compared to the no synth case.

The SC model obtains images without major degradations of the visual quality of the image, especially if a suitable decision on which elements to preserve is made. This is particularly evident in scenarios where the synthesized elements have a repetitive pattern or texture, such as trees, streets or the sky. Moreover, very low bitrates (bitrate savings) can be obtained if compared to the no synth case for which no image element was synthesized (i.e. information of all image regions was transmitted).

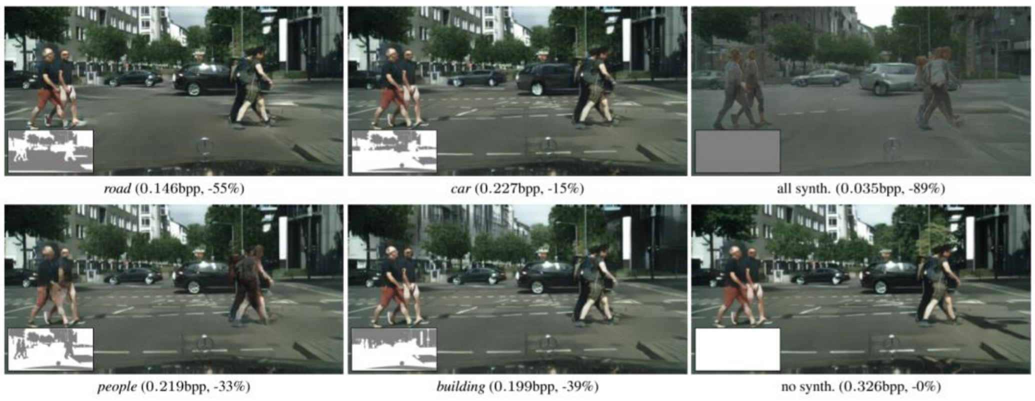

road (0.146bpp, -55%)
car (0.227bpp, -15%)
all synth. (0.035bpp, -89%)
people (0.219bpp, -33%)
building (0.199bpp, -39%)
no synth. (0.326bpp, -0%)

Figure 22 – Example of visual comparison for different synthesised elements of the image using the SC model with RI training [30].

3.2. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network

In Super Resolution (SR) a higher resolution image (HR) is obtained from one or more lower resolutions (LR) images [47, 48]. Super resolution algorithms have several applications, such as enhancement of satellite and aerial images [49], medical image processing [50], among others. DL-based solutions have often proven to be more efficient than traditional SR solutions, such as those based on hand-crafted interpolation filters [51]. This is mainly due to the ability to extract relevant higher-level features when finding a mapping between the LR and HR images [52].

Deep convolutional networks designed for this problem and trained using the MSE as loss function result in high peak signal-to-noise (PSNR) ratios, but usually lack the fine texture details that are characteristic of an HR image. To meet the fidelity expected from an HR image, this section presents a GAN-based solution that attempts to minimize this shortcoming by introducing a new objective function.

A. Objective and Technical Approach

In 2017, Ledig *et al.* [52] proposed a generative adversarial network for image super-resolution (SRGAN) that can up-sample images up to 4x using a single low-resolution image. The introduction of the GAN adversarial loss makes the HR images richer in high-frequency details, when compared to other SR algorithms that use MSE in the loss function. When the MSE loss function is used, pixel-wise similarities between the ground-truth and the SR image are accounted and thus, high PSNR quality scores are obtained; however, this often results in perceptually unsatisfying HR images.

Also, this work introduces a perceptual loss function that balances the generation of content with the preservation of original LR image content. This new training objective coupled with a ResNet [53] inspired architecture creates a solution that outperforms previous state-of-the-art models.

B. *Architecture and Walkthrough*

In this work, two different models for the super-resolution problem are proposed:

- **SRResNet model**: an application of a ResNet [53] trained with the MSE as its loss function. This network is composed of 16 residual blocks that are briefly described in Section 3.2 - C. The fact that the model is optimized for MSE makes it more sensible to pixel changes and invariant to perceptual changes. This model sets a new state of the art in image SR in PSNR and SSIM measures, as showcased in Section 3.2 - D and, as such, serves as reference when testing the other model proposed in this paper.
- **SRGAN model**: GAN based solution composed of the usual Generator-Discriminator pair. The generator creates reconstructed images very similar to the real images and thus difficult to classify by the discriminator, which attempts to distinguish between the real images and the reconstructed high-resolution images created by the generator.

The proposed generator network is shown in Figure 23 and reuses the residual blocks and skip connections as defined in ResNet (see Section 3.2 - C). A block composed of two convolutional (Conv) layers with 3×3 kernels, 64 feature maps and stride 1 are followed by a batch-normalization (BN) layer [54] and a parametric ReLU (PreLU) [55] as the activation function. These six layers form a residual block since the input of the block is added by an element-wise sum to the block output. Moreover, a skip connection is also present that adds the input of the B residual blocks to their output after processing by a Conv+BN layers.

After the last element-wise sum, the next network layers are responsible to perform the actual up-sampling operation. This upscaling is achieved using sub-pixel convolutions as proposed by Shi *et al.* in [56]. These layers are composed of a convolutional layer with 3×3 kernels, 256 feature maps and stride 1, followed by a PixelShuffler layer with an upscale factor of 2 and end with a PReLU activation layer. The last layer is a convolutional layer with a 9×9 kernel, 3 feature maps and stride 1 which produces the high-resolution image. The training of this generator is the same as described in Section 2.3.3. However, the learning process is guided by a perceptual loss function, described in Section 3.2 - C, that weights the contributions of the adversarial loss with a content loss.

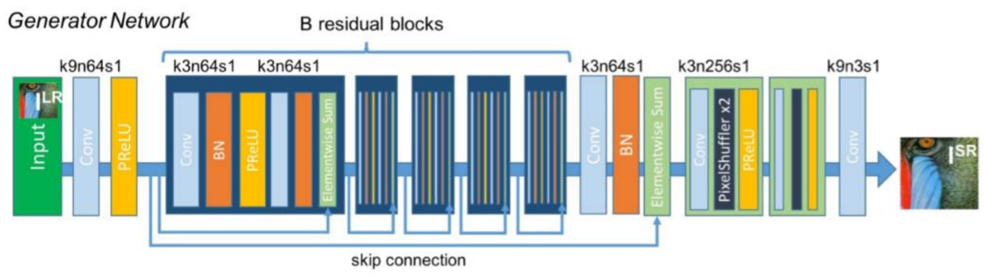


Figure 23 - Architecture of the Generator Network with corresponding kernel size (k), number of feature maps (n) and stride (s) shown for each convolutional block [52].

The discriminator network is shown in Figure 24 and is also a convolutional network with 8 convolutional layers. These layers have 3×3 kernels, and the number of kernels doubles every 2 convolutional layers, starting with 64 and ending with 512 kernels. The image resolution is reduced using stride 2 in the convolutional layers, each time the number of features is doubled. The convolutional layers are followed by a chain of dense layer (fully connected layer) with height 1024, LeakyReLU activation and a dense layer with height 1. The last layer is a sigmoid activation function that makes the output range from 0 to 1, representing a probability of the image belonging to the training dataset.

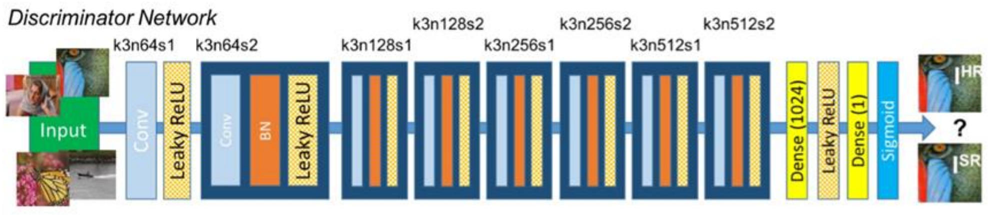


Figure 24 - Architecture of the Discriminator Network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer [52].

C. Main Tools

The most relevant processing tools defined or used in this solution are:

- **Perceptual loss function:** this work defines an objective loss function that attempts to follow more closely the human perception of image quality compared to previous MSE optimized SR deep neural network solutions. The proposed perceptual loss function is a weighted sum of two terms: 1) content loss l_X^{SR}, which measures image similarity, and 2) adversarial loss l_{Gen}^{SR}. The content loss function can be the any function that measures the similarities between two images, however in this paper it is either the MSE or the VGG loss that is define next. Regarding the adversarial loss, it is computed as described in Section 2.3.3, since is the typical adversarial loss.

$$l^{SR} = l_X^{SR} + 10^{-3}\, l_{Gen}^{SR} \tag{8}$$

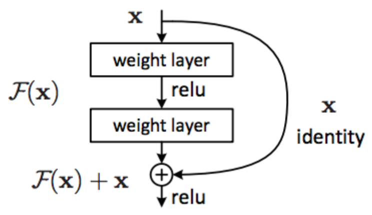
With this loss function it is possible to obtain a balance between the generated textures (fine details) with the preservation/fidelity of the original image. The use of a GAN framework allows to learn an efficient mapping between the LR and the HR (desired) images and the adversarial loss term pushes the model to output HR images with superior perceptual quality, i.e. the generator is able to create realistic-looking images. However, in the context of the SR problem, fidelity is also important. Thus, the content loss term is responsible for guaranteeing that not only is the HR realistic, but also maintains a close relationship with its LR counterpart.

- **VGG loss:** the VGG loss tries to mitigate the blurry effect (lack of detail and smooth textures) that the MSE produces. This loss is computed on the feature maps of the VGG19 network proposed by Simonyan and Zisserman [57] composed of 19 convolutional layers. Thus, the ground truth (I^{HR}) and the prediction of the generator ($G_{\theta_G}(I^{LR})$) are passed through the VGG network. Then the VGG loss is computed as the Euclidean distance between the feature map in the j-th convolutional layer and before the i-th maxpooling layer of the NN ($\phi_{i,j}$), as depicted in (9).

$$l^{SR}_{VGG/i.j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} \left(\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}\left(G_{\theta_G}(I^{LR})_{x,y}\right)\right)^2 \quad (9)$$

  The introduction of this loss allows to make the comparison between the ground truth and the generator predictions in a more perceptual space compared to the usual MSE loss.

- **ResNet:** Kaiming *et al.* [53] proposed a deep residual network for image recognition (ResNet) where residual blocks, convolutional blocks with non-linearities that employ skip connections, are proposed. Deeper NN tends to be harder to train, as such, the introduction of skip connections (or residuals blocks) in the architecture of the generator, facilitates the training process. The skip connection is a connection between the output of one layer with the input of a previous layer. The residual blocks guarantee that the performance of deeper networks is not degraded (i.e. deals with the vanishing gradient problem). This type of blocks was used in this SR solution since it is easier to optimize and can produce more accurate results than the typical convolutional blocks.



x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 25 - Architecture of a generic residual block [53].

D. *Performance Assessment*

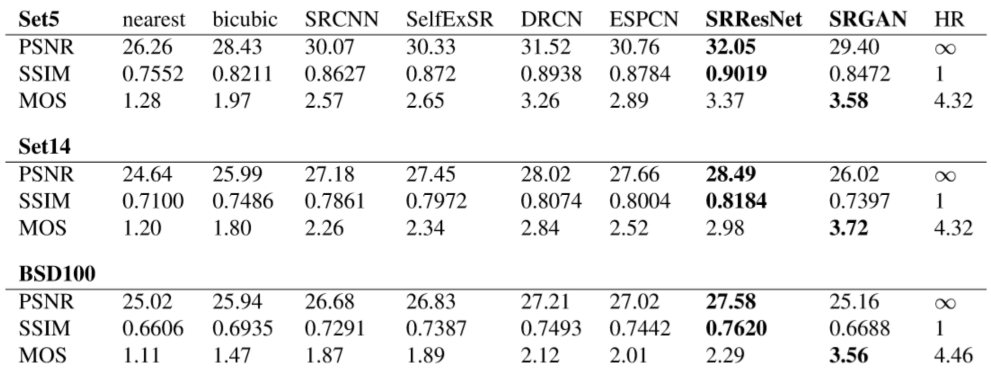
The proposed models SRResNet and SRGAN, with VGG54 as the content loss function, were evaluated on three datasets, namely Set5 [58], Set14 [59], and BSD100 [60]. Some reference methods (benchmarks) were selected, namely nearest neighbor, bicubic, SRCN [61]. SelfExSR [62], DRCN [63], and ESPCN [56]always for an 4x up-scale factor.

All solutions were trained on 350 thousand images obtained from the ImageNet dataset [64, 6]. To obtain the LR images, a bicubic kernel with a down-sampling factor of 4 is employed. The LR and HR images were scaled to the range, $[0,1]$ and $[-1,1]$, respectively. Consequently, the MSE loss computed varies from $[-1,1]$ and the VGG loss was also scaled to be on the same range of the MSE loss. The Adam optimizer [65] was used with the momentum term equal to 0.9 and the generator network was initialized with the trained MSE-based SRResNet network to guarantee that some desired local minima can be found during training. The SR solutions based on the GAN framework were trained with $k = 1$, where k corresponds is the GAN framework hyperparameter described in Section 2.3.3, and were performed two times 10^5 update iterations, first with a learning rate of 10^{-4} and then with learning rate of 10^{-5}.

To assess the objective quality of the HR images, the PSNR and SSIM objective metrics were used. Subjective experiments were performed and MOS (Mean Opinion Score) computed from the scores of the experiment subjects. This objective (PSNR and SSIM) and subjective (MOS) quality evaluation ensures a complete analysis of the performance of all models. Regarding the MOS, 26 participants classified a total of 1128 images each, from 1(bad quality) to 5 (excellent quality).

The results obtained in the aforementioned conditions are presented in Table 1. The SRResNet proposed solution outperforms its benchmarks regarding objective metrics (PSNR and SSIM). Moreover, the MOS results highlights the fact that the SRGAN is superior in perceptual quality when compared to the competition. However, the objective metrics do not show the same behaviour as the MOS, which allows to conclude that these metrics do not fully account the image quality with respect to the human visual system.

Table 1 – Experimental results of NN, bicubic, SRCNN, SelfExSR, DRCN, ESPCN, SRResNet, SRGAN and the original HR on benchmark data. Highest measures (PSNR [dB], SSIM, MOS) in bold [4×upscaling] [52].



Set5	nearest	bicubic	SRCNN	SelfExSR	DRCN	ESPCN	**SRResNet**	**SRGAN**	HR
PSNR	26.26	28.43	30.07	30.33	31.52	30.76	**32.05**	29.40	∞
SSIM	0.7552	0.8211	0.8627	0.872	0.8938	0.8784	**0.9019**	0.8472	1
MOS	1.28	1.97	2.57	2.65	3.26	2.89	3.37	**3.58**	4.32
Set14									
PSNR	24.64	25.99	27.18	27.45	28.02	27.66	**28.49**	26.02	∞
SSIM	0.7100	0.7486	0.7861	0.7972	0.8074	0.8004	**0.8184**	0.7397	1
MOS	1.20	1.80	2.26	2.34	2.84	2.52	2.98	**3.72**	4.32
BSD100									
PSNR	25.02	25.94	26.68	26.83	27.21	27.02	**27.58**	25.16	∞
SSIM	0.6606	0.6935	0.7291	0.7387	0.7493	0.7442	**0.7620**	0.6688	1
MOS	1.11	1.47	1.87	1.89	2.12	2.01	2.29	**3.56**	4.46

In Figure 26, some examples of the images produced by the Bicubic, SRResNet and SRGAN are shown along with the original image. As shown in these examples, the quality improvements of SRResNet and SRGAN are visible, especially in high frequency details from the bicubic to the SRResNet, and from the SRResNET to the SRGAN.

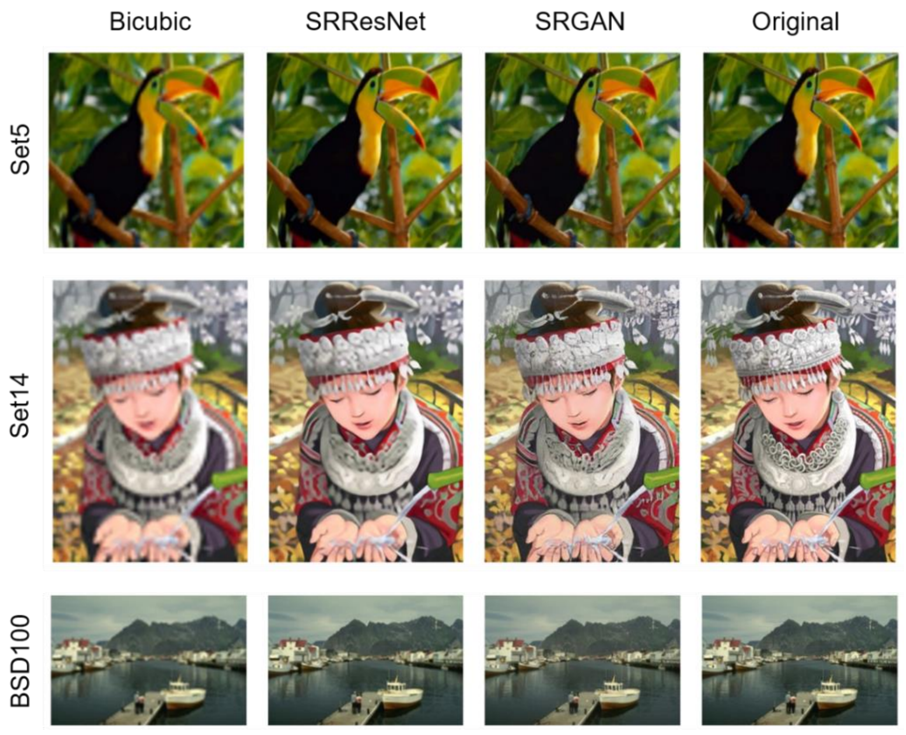


Figure 26 -Visual comparison between Bicubic, SRResNet, SRGAN and the Original image using images from different datasets: Set5 (top), Set14 (middle) and BSD100 (bottom) (Adapted from [52]).

3.3. Deep Universal Generative Adversarial Compression Artifact Removal

Compression algorithms are used in several contexts to increase the efficiency of storage and transmission of data. There are two main types of compression [66], lossless compression, and lossy compression. In lossless compression algorithms, the compressed version of the data retains all the information contained in the original, where in lossy compression some of the information is discarded, typically information that is considered less relevant, to achieve higher compression ratios.

Only a few applications, such as medical imaging [67], require lossless image compression, which means that lossy compression algorithms are more often used. Unfortunately, lossy compression introduces quality loss and produces some compression artifacts, such as blocking, posterizing, contouring, blurring, and ringing effects. These artifacts not only degrade the visual quality of the images but also impair the performance of computer vision algorithms, such as an object detector.

In the past, several conventional solutions have been proposed to remove these artifacts, mainly based on statistical modelling, without any learning from a dataset of examples. In this section an ensemble of GAN based convolutional neural networks is described with the aim to obtain more realistic reconstructions.

A. Objective and Technical Approach

Galteri *et al.* [68] proposed, in 2019, a compression artifact removal solution that is efficient even in highly degraded images and works in scenarios where the encoding parameters are unknown. This is different from previous deep learning based methods which assume that the encoding parameters are known, namely the compression factor QF. To achieve this objective, the authors introduce an ensemble of deep convolutional residual networks, trained for different compression quality factors (QF), coupled with a quality predictor model. Each model its trained for different compression qualities, i.e. from lower to higher compression rates, allowing for a tailored image reconstruction.

The core of this solution is a neural network that can be trained with direct supervision or with adversarial training. The authors explore several configurations for this model, namely using the GAN framework and balancing the adversarial loss with a content loss or using direct supervision with MSE and SSIM. Regarding the content loss several alternatives were evaluated, such as the MSE, SSIM, and VGG19 [57].

B. Architecture and Walkthrough

The architecture of the proposed compression artifact removal solution is shown in Figure 27. The proposed solution introduces an ensemble of GAN models (generators), which are convolutional networks trained with decoded images compressed with different quality factor. The input compressed image (I^c) is passed to a compression quality predictor NN, described with more detail in Section 3.3 - C, that estimates the image QF. With this estimation the QF switcher passes the I^c to the better suited GAN and the reconstructed I^R image is obtained.

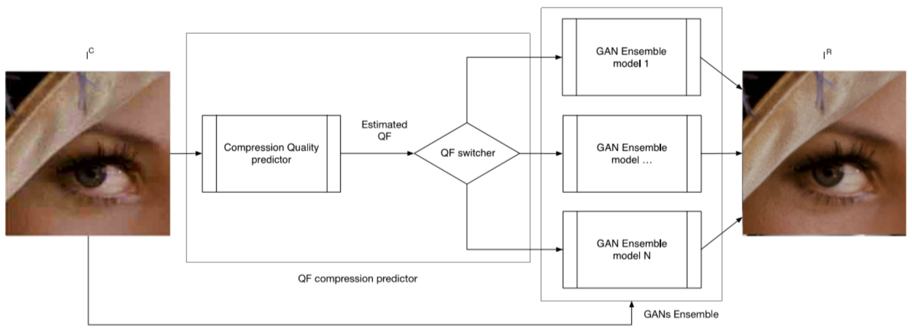


Figure 27 - Architecture of the universal compression artifact removal solution [68].

The GAN model that performs the quality enhancement includes the generator network shown in Figure 28. It follows a generative approach which given an input image is able to output an improved version of the same image. This neural network is composed of convolutional layers with 3×3 kernels and 64 feature maps with Leaky ReLU as its activation function. The convolutional layers all have stride 1, except for the 2nd layer that has stride 2 to reduce the feature map to half of its original size. This layer is followed by 15 similar residual blocks, inspired by K. He *et al.* [53] work, that have a skip connection every two convolutional layers. The output of this chain of residual blocks is up-sampled (after a convolutional layer) using nearest-neighbour [69], which allows to obtain feature maps with the same size as the original image. To mitigate the impact of some undesired up-sampling artifacts, another two convolutional layers with the same parameters as before is used. The last layer of the NN is a convolutional layer with one feature map and Tanh as its activation function, this non-linearity makes the resulting images range from $[-1, 1]$.

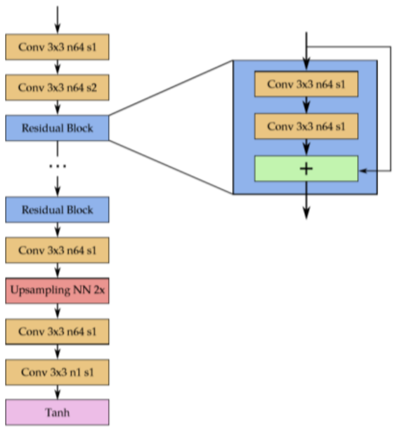


Figure 28 – Architecture of the deep convolutional residual neural network, where n is number of filters and s is the stride [70].

The architecture of the proposed Discriminator is shown in Figure 29. In a similar way as the Generator, the discriminator is composed of convolutional layers with 3×3 kernels, stride 1 and Leaky ReLU as its activation function. The number of filters doubles every 2 layers until reaching 256. The last convolutional layer has 2×2 kernels and a feature map of size 1. Finally, a non-linearity is applied, namely a sigmoid, to make the output range from 0 to 1 and represents the adversarial loss to be used during training. The discriminator divides the input images into patches (similar to blocks) of size 16×16 making the evaluation at the patch level instead of an evaluation of the entire image. This is done since compression algorithms typically decompose images into patches before compression, thus creating artifacts within patches. Which means, that the better the generator is at reconstructing each patch the more accurate the reconstructions will be. This approach allows to remove existing high frequency noise, like mosquito noise and thus improves the overall reconstruction quality.

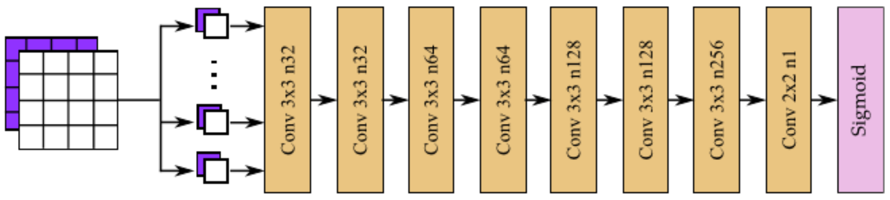


Figure 29 – Discriminator Network Architecture, where n is number of filters and s is the stride [68].

C. Main Tools

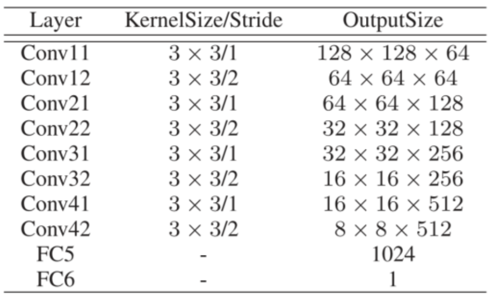
The main tools used in this solution are:

- **New training objective:** training with MSE and SSIM as the loss function typically produces images that lack some of its finer details and consequentially are less appealing to the human eyes. To mitigate this problem, adversarial training can be used to generate more photorealistic images. However, similar to the SR problem described in Section 3.2 [52], the preservation of the original content is also crucial for compression artefact removal and thus, requires to be balanced with the generation of new content. The authors introduce an adversarial patch loss (10) that weights the typical adversarial loss (l_{adv}), described in Section 2.3.3, with a perceptual loss term (l_P), that evaluates the similarities between the compressed image I^C and the reconstructed image I^R. These terms are weighted with the parameter λ.

$$l_{AR} = l_P + \lambda\, l_{adv} \tag{10}$$

- **Perceptual loss:** functions such as MSE and SSIM can be used to evaluate the similarities between I^C and I^R, but often do not represent well the human visual perception of image similarity. To improve this comparison, and consequently have more reliable l_P values, it is introduced a loss computed in the feature maps of a pre-trained VGG19 network [57]. This is a similar approach to the perceptual loss function introduced in the SR model described in Section 3.2, where the VGG19 network is explained with more detail (see Section 3.2 - C).
- **Quality predictor network:** this network is extremely important in order to have an ensemble of NN that work efficiently. This network correctly identifies the better suited GAN in the ensemble to reconstruct any given compressed image. The main purpose of this network is to produce accurate estimations of the compression quality used in the input image (I^c). The predictor network proposed in this work is described in Table 2 and is a convolutional neural network composed of 8 convolutional layers with kernels of size 3×3 and stride alternating between 1 and 2. The layers with stride 2 produce a reduction in the feature map size by half, each time. After the 8^{th} convolutional layer two fully connected layers combine the previous output maps and translate them to a single value corresponding to the quality estimation.

Table 2 – Architecture of the proposed quality predictor network [68].



Layer	KernelSize/Stride	OutputSize
Conv11	$3 \times 3/1$	$128 \times 128 \times 64$
Conv12	$3 \times 3/2$	$64 \times 64 \times 64$
Conv21	$3 \times 3/1$	$64 \times 64 \times 128$
Conv22	$3 \times 3/2$	$32 \times 32 \times 128$
Conv31	$3 \times 3/1$	$32 \times 32 \times 256$
Conv32	$3 \times 3/2$	$16 \times 16 \times 256$
Conv41	$3 \times 3/1$	$16 \times 16 \times 512$
Conv42	$3 \times 3/2$	$8 \times 8 \times 512$
FC5	-	1024
FC6	-	1

D. Performance Assessment

The evaluation of performance is presented in three different studies:

- **Evaluation A – GAN model:** objective evaluation of just the proposed generator network, i.e. without the ensemble depicted in Figure 27, using metrics such as PSNR, PSNR-B [71] and SSIM.
- **Evaluation B – Ensemble of GANs:** objective evaluation of the proposed ensemble of GANs using the same metrics of the first evaluation.
- **Evaluation C – GAN subjective evaluation:** subjective evaluation that compares the MOS results of just the generator model when trained with and without adversarial loss.

Evaluation A: GAN model

The generator network was evaluated on the BDS500 [60] and LIVE1 [72] datasets and compared with some state-of-the-art approaches, namely SA-DCT [73], AR-CNN [74] and, in some cases with the work of Svoboda *et al.* [75] and CAS-CNN [76]. Twelve different versions of the proposed generator NN were tested, for the three of the training loss functions described next, for each JPEG compression factor (QF) of 10, 20, 30 and 40:

- Perceptual loss: uses the capabilities of adversarial training and corresponds to the "our GAN" entry in the results presented in Table 3;
- MSE: corresponding to the "our MSE" entry in the results presented in Table 3;
- SSIM, corresponding to the "our SSIM" entry in the results presented in Table 3;

These models were trained on 16 random 128×128 patches from the MS-COCO [77] dataset, with random flipping and rotation for data augmentation. The images were compressed using the MATLAB JPEG compressor at the corresponding QF parameter. The training is done with the Adam optimizer with the momentum term equal to 0.9 and a learning rate equal to 10^{-4} on the first 50k iterations and the last 50k iterations have a smaller learning rate of 10^{-5}.

The quality assessment is performed on luminance only, which means that I^C and I^R images are transformed to gray-scale before evaluation. The models are evaluated on PSNR, PSNR-B and SSIM for both test sets and the results are presented in Table 3. The proposed models shows better results on objective metrics than the other deep models, except for the PSNR-B measures at QFs of 10 and 40 where the model proposed by Cavigelli *et al.* (CAS-CNN) [76] slightly outperforms it.

Table 3 - Evaluation of the proposed generator NN and selected benchmarks using the PSNR, PNSR-B and SSIM quality metrics on BDS500 and LIVE1, with the best result for each metric and respective QF group in bold [68].

QF	Method	LIVE1			BSD500		
		PSNR	PSNR-B	SSIM	PSNR	PSNR-B	SSIM
10	JPEG	27.77	25.33	0.791	27.58	24.97	0.769
	SA-DCT [12]	28.65	28.01	0.809	-	-	-
	AR-CNN [9]	29.13	28.74	0.823	28.74	28.38	0.796
	L4 [43]	29.08	28.71	0.824	28.75	28.29	0.800
	CAS-CNN, MS loss[4]	29.36	28.92	0.830	-	-	-
	CAS-CNN, w/ loss FT[4]	29.44	**29.19**	0.833	**-**	**-**	**-**
	ED [51]	29.40	29.09	0.833	28.96	28.57	0.806
	CED-EST [51]	29.40	29.08	0.832	28.95	28.56	0.805
	CED-GT [51]	26.54	26.51	0.767	26.00	25.97	0.731
	Our MSE	**29.47**	29.13	0.833	**29.05**	**28.64**	0.806
	Our SSIM	28.94	28.46	**0.841**	28.53	27.97	**0.816**
	Our GAN	27.65	27.63	0.777	27.31	27.28	0.749
20	JPEG	30.07	27.57	0.868	29.72	26.97	0.852
	SA-DCT [12]	30.81	29.82	0.878	-	-	-
	AR-CNN [9]	31.40	30.69	0.890	30.80	30.08	0.868
	L4 [43]	31.42	30.83	0.890	30.90	30.13	0.871
	L8 [43]	31.51	30.92	0.891	30.99	30.19	0.872
	CAS-CNN, MS loss[4]	31.67	30.84	0.894	-	-	-
	CAS-CNN, w/ loss FT[4]	31.70	30.88	0.895	-	-	-
	ED[51]	31.68	31.14	0.895	31.08	30.33	0.875
	CED-EST [51]	31.65	31.13	0.895	31.04	30.32	0.875
	CED-GT [51]	29.33	29.32	0.854	28.62	28.58	0.825
	Our MSE	**31.81**	**31.29**	0.897	**31.23**	**30.49**	0.877
	Our SSIM	31.51	30.84	**0.901**	30.92	30.01	**0.883**
	Our GAN	29.99	29.69	0.864	29.48	29.03	0.841
30	JPEG	31.41	28.92	0.900	30.98	28.23	0.886
	SA-DCT [12]	32.08	30.92	0.908	-	-	-
	AR-CNN [9]	32.69	32.15	0.917	**-**	**-**	**-**
	Our MSE	**33.21**	**32.51**	0.923	**32.53**	**31.57**	0.906
	Our SSIM	32.95	32.17	**0.926**	32.26	31.16	**0.911**
	Our GAN	31.65	31.38	0.900	31.04	30.57	0.881
40	JPEG	32.35	29.96	0.917	31.88	29.14	0.906
	SA-DCT [12]	32.99	31.79	0.924	**-**	**-**	**-**
	AR-CNN [9]	33.63	33.12	0.931	-	-	-
	CAS-CNN, MS loss[4]	33.98	32.83	0.935	-	-	-
	CAS-CNN, w/ loss FT[4]	34.10	**33.68**	0.937	-	-	-
	Our MSE	**34.17**	33.42	0.937	**33.45**	**32.34**	0.923
	Our SSIM	33.98	33.07	**0.939**	33.25	31.94	**0.926**
	Our GAN	31.64	31.17	0.903	30.98	30.16	0.884

Figure 30 shows the results for two images for some region highlighted in yellow in the original image. Both the JPEG 20 decoded image, the proposed GAN artifact removal algorithm along with the AR-CNN benchmark output are shown. The JPEG compression algorithm with QF 20 introduces some undesired artifacts, namely blocking, ringing and color quantization. The reconstruction performed by the AR-CNN removes the artifacts of the decoded image, but the image is somewhat blurry, lacking some textures and finer details. The GAN proposed in this paper not only eliminates the compression artifacts but is also able to produce more faithful reconstructions of the image.

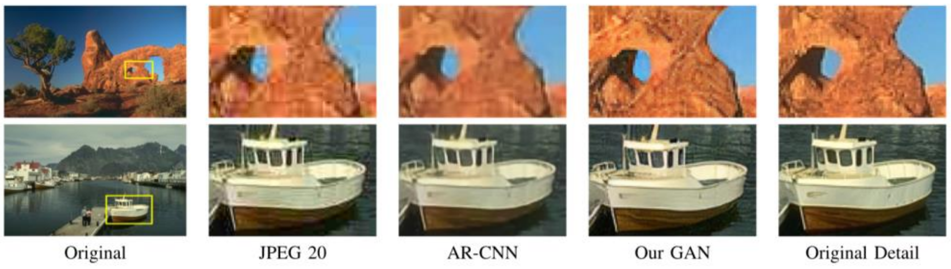


Figure 30 - Visual comparison between JPEG (QF = 20), AR-CNN, the proposed solution and the Original [68].

Evaluation B: Ensemble of GANs

The proposed ensemble of GANs is evaluated with different configurations. Each ensemble combines the generator networks trained for the same loss function but each GAN network for a different QF and is driven by a quality predictor network. The following ensembles are evaluated:

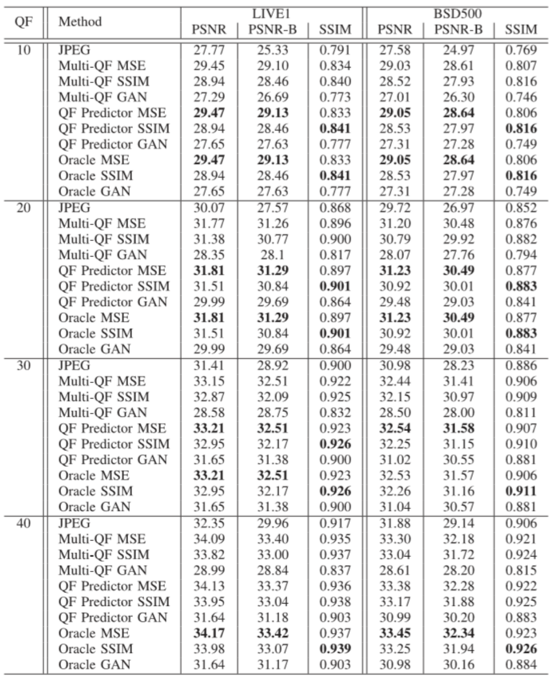
- QF predictor GAN: Ensemble of 4 generator networks trained with the proposed perceptual loss.
- QF predictor MSE: Ensemble of 4 generator networks trained with MSE loss function.
- QF predictor SSIM: Ensemble of 4 generator networks trained with SSIM loss function.

These models are evaluated against other three variants of the proposed generator network not in ensemble, for the same loss functions previously presented (perceptual loss, MSE and SSIM). However, these networks are not trained with a training data comprised of images compressed under the same condition (that is, same QF). The Multi-QF models are trained over images compressed with different QFs using the three loss functions and are referred as Multi-QF GAN, Multi-QF MSE, Multi-QF SSIM.

The ensemble models and the Multi-QF networks are compared with the generator by himself trained as described in Evaluation A. This means that for each target QF, there is a network trained with images compressed with always the same target QF. Again, different loss functions are used, and the networks obtained are referred as Oracle GAN, Oracle MSE and Oracle SSIM.

All of these models are evaluated under the same in the conditions of the previous experiments, which means that are evaluated with the same QFs, metrics and datasets. The experimental results obtained are shown in Table 4. It is clear that the ensemble performs better than Multi-QF. The ensemble shows the same results as its oracle corresponding networks, highlighting the correct behaviour of the QF predictor.

Table 4 - Comparison between the proposed ensemble of NN and the Multi-QF NN regarding the PSNR, PNSR-B and SSIM on BDS500 and LIVE1, with the best result for each metric and respective QF group in bold [68].



QF	Method	LIVE1			BSD500		
		PSNR	PSNR-B	SSIM	PSNR	PSNR-B	SSIM
10	JPEG	27.77	25.33	0.791	27.58	24.97	0.769
	Multi-QF MSE	29.45	29.10	0.834	29.03	28.61	0.807
	Multi-QF SSIM	28.94	28.46	0.840	28.52	27.93	0.816
	Multi-QF GAN	27.29	26.69	0.773	27.01	26.30	0.746
	QF Predictor MSE	**29.47**	**29.13**	0.833	**29.05**	**28.64**	0.806
	QF Predictor SSIM	28.94	28.46	**0.841**	28.53	27.97	**0.816**
	QF Predictor GAN	27.65	27.63	0.777	27.31	27.28	0.749
	Oracle MSE	**29.47**	**29.13**	0.833	**29.05**	**28.64**	0.806
	Oracle SSIM	28.94	28.46	**0.841**	28.53	27.97	**0.816**
	Oracle GAN	27.65	27.63	0.777	27.31	27.28	0.749
20	JPEG	30.07	27.57	0.868	29.72	26.97	0.852
	Multi-QF MSE	31.77	31.26	0.896	31.20	30.48	0.876
	Multi-QF SSIM	31.38	30.77	0.900	30.79	29.92	0.882
	Multi-QF GAN	28.35	28.1	0.817	28.07	27.76	0.794
	QF Predictor MSE	**31.81**	**31.29**	0.897	**31.23**	**30.49**	0.877
	QF Predictor SSIM	31.51	30.84	**0.901**	30.92	30.01	**0.883**
	QF Predictor GAN	29.99	29.69	0.864	29.48	29.03	0.841
	Oracle MSE	**31.81**	**31.29**	0.897	**31.23**	**30.49**	0.877
	Oracle SSIM	31.51	30.84	**0.901**	30.92	30.01	**0.883**
	Oracle GAN	29.99	29.69	0.864	29.48	29.03	0.841
30	JPEG	31.41	28.92	0.900	30.98	28.23	0.886
	Multi-QF MSE	33.15	32.51	0.922	32.44	31.41	0.906
	Multi-QF SSIM	32.87	32.09	0.925	32.15	30.97	0.909
	Multi-QF GAN	28.58	28.75	0.832	28.50	28.00	0.811
	QF Predictor MSE	**33.21**	**32.51**	0.923	**32.54**	**31.58**	0.907
	QF Predictor SSIM	32.95	32.17	**0.926**	32.25	31.15	0.910
	QF Predictor GAN	31.65	31.38	0.900	31.02	30.55	0.881
	Oracle MSE	**33.21**	**32.51**	0.923	32.53	31.57	0.906
	Oracle SSIM	32.95	32.17	**0.926**	32.26	31.16	**0.911**
	Oracle GAN	31.65	31.38	0.900	31.04	30.57	0.881
40	JPEG	32.35	29.96	0.917	31.88	29.14	0.906
	Multi-QF MSE	34.09	33.40	0.935	33.30	32.18	0.921
	Multi-QF SSIM	33.82	33.00	0.937	33.04	31.72	0.924
	Multi-QF GAN	28.99	28.84	0.837	28.61	28.20	0.815
	QF Predictor MSE	34.13	33.37	0.936	33.38	32.28	0.922
	QF Predictor SSIM	33.95	33.04	0.938	33.17	31.88	0.925
	QF Predictor GAN	31.64	31.18	0.903	30.99	30.20	0.883
	Oracle MSE	**34.17**	**33.42**	0.937	**33.45**	**32.34**	0.923
	Oracle SSIM	33.98	33.07	**0.939**	33.25	31.94	**0.926**
	Oracle GAN	31.64	31.17	0.903	30.98	30.16	0.884

Evaluation C – GAN subjective evaluation

Two different generator networks without ensemble, one trained with SSIM as the loss function and another with the proposed perceptual loss, are evaluated with subjective experiments (thus collecting MOS scores). Ten subjects participated in these experiments. These subjects compare the reconstruction of the images with the original giving a score of 0 to 100 on a scale of similarity between them. This experiment was done over 50 random images from the BSD500 [60] and only includes the reconstructions obtained with the model directly trained by the SSIM and the proposed GAN approach.

Figure 31 illustrates the MOS scores obtained in this subjective evaluation. The results obtained in this experiment show an overall preference to the GAN reconstruction with only a few exceptions. In general, the GAN model obtains a MOS of 68.32 with a standard deviation of 20.75, whereas the SSIM model only achieved a MOS of 49.51 with a comparable standard deviation of 22.72.

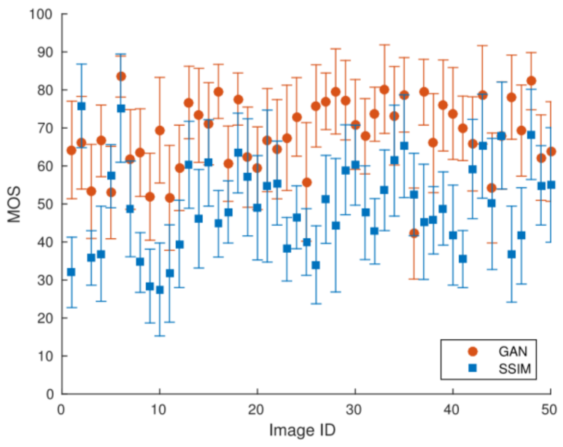


Figure 31 - MOS scores and confidence intervals for the 50 random BSD500 images used in the subjective evaluation [68].

Chapter 4

4. GAN-based Image Processing: Subjective Quality Evaluation

In recent years, GANs have become progressively more capable of generating realistic images and have reached a point where, perceptually, the images generated can be confused as real images. These advances propelled the development of a variety of GAN-based solutions for several image processing tasks such as super-resolution, compression, artifact removal, etc. This, in turn, created the need to accurately evaluate the image quality of these solutions. In this context, subjective evaluations are crucial in the quality assessment process and this chapter provides an overview of the methodologies and conditions under which some GAN-based solutions are evaluated.

The results obtained from this study are publicly available and ca be seen at https://github.com/martafilipa/gan-image-database.

4.1. Test Material and Preparation

To design a subjective assessment, it is important to define what will constitute the test material. That is to say, which images will compose the study, how will the images be shown and what preparations need to be done beforehand.

4.1.1. Dataset selection

The JPEG AI dataset [78] was used for this subjective quality evaluation seeing that it was designed to evaluate the performance of state-of-the-art learning-based image coding solutions. Due to its nature, this dataset is divided into 2 subsets. The test material used in this assessment belonged to the test subset since it provides a diverse and well-balanced set of images that allows the evaluation to be representative.

To keep the test session restrained to less than 60 minutes from the 16 total images that compose this test set only 8 are used in the subjective assessment. The selection of the 8 test images is done by visual inspection and considering the need for diversity. Moreover, to aid in the decision process the test set contents are categorized as presented in Table 5. This categorization helps when it comes to choosing a diverse set of scenes, however, the set needs to be balanced when it comes to the complexity of said scenes. That being said, the images are selected to range from less to more complex when it comes to high-frequency components, color saturation, etc.

Table 5 -Categorization of JPEG AI test set images by its contents. Highlighted in green are the selected test images and in orange are the images included in the training phase.

Landscapes	**Synthetic**	**Texture**	**Subjects**
Matterhorn (00001)	Curiosity Rover (00010)	Rotunda of Mosta (00004)	Racing car (00002)
Train (00006)		Windows (00007)	Sardinia festival (00003)
Transmission towers (00008)		Port (00009)	Las Vegas sign (00005)
Beach (00014)		Bell pepper (00011)	Woman (00012)
Harbor (00015)			Dog (00013)
			Ponytail (00016)

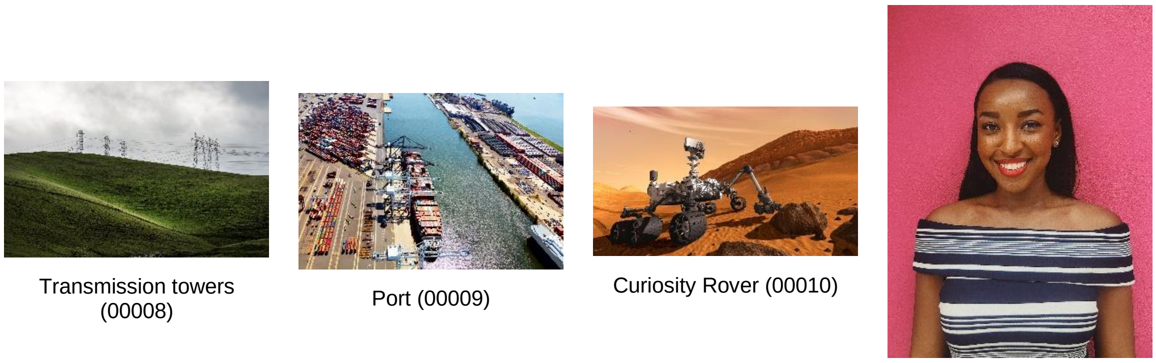
That being said, the images selected are presented in Figure 32. Besides the images used in the test it was also necessary a set of extra images to include in the training phase of the assessment. The training phase as the goal of getting the user familiar with the test environment and with the type of content that will evaluate in the test. This leads to the need of having a set of images that are not included in the test set and are representative of the type of artefacts produced by the GAN-based solutions. With this is mind the images selected were Matterhorn (00001) and Harbor (00005).

Racing car (00002)

Rotunda of Mosta (00004)

Las Vegas sign (00005)

Train (00006)



Transmission towers (00008)

Port (00009)

Curiosity Rover (00010)

Woman (00012)

Figure 32 - Images from JPEG AI test dataset selected for subjective assessment.

4.1.2. Display Conditions

Normally the subjective assessment would be done in a controlled environment where all subjects would visualize the images in the same monitor with the same configurations. However, due to the limitations imposed by the global pandemic, the approach shifted in the direction of an online crowdsourcing survey.

With the assessment done in different devices, there are consequently different viewing conditions of the test material. This creates the need to control as much as possible the conditions under which the survey is taken. In order to do so, the app developed to employ subjective assessment imposes restrictions regarding the resolution and size of the display. Particularly, it enforces a resolution of at least 1920x1080 and a display size of at least 13 inches.

4.1.3. Content Preparation

Considering that this assessment follows a Forced Choice (FC) design, meaning that two images need to be displayed side-by-side, it is important to determine the target size for the test material. With the previously mentioned target resolution of 1920x1080 and the layout presented in Figure 33 the desired resolution for the test images is 940x880.

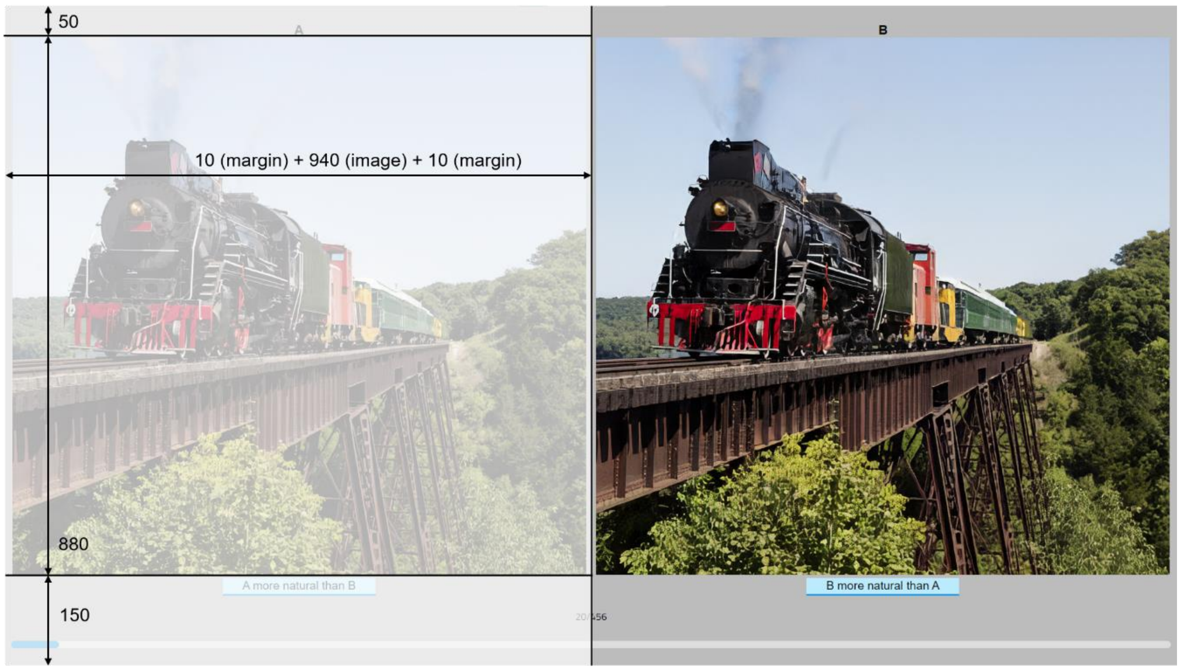


Figure 33 – Example of test layout highlighting the requirements for image sizes targeting the screen size of 1920x1080.

In order to adapt the dataset to the desired resolution are performed crops taking into consideration the need to preserve the relevant elements of each image. In Figure 34 is presented an example of a crop performed over the *Train* image. The remaining example of crops and corresponding resolution information is included in Annex A.1.

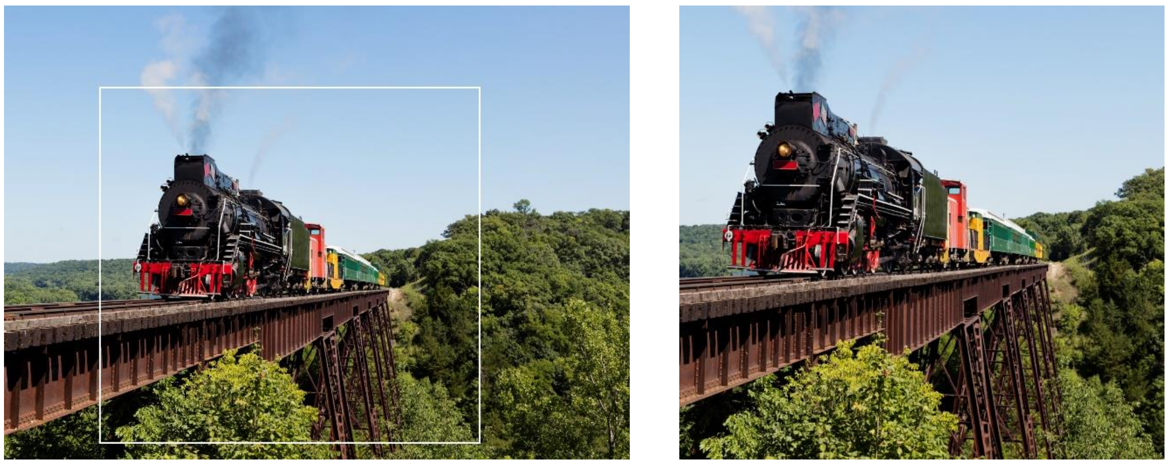


Figure 34 - Example of the original JPEG AI image (left) and respective crop (right).

4.2. GAN-based Image Processing Solutions and Benchmark

This section provides an overview of the solutions used to obtain the images that will make part of the subjective assessment. In this case, the preprocessing of the images, the training of the model, the postprocessing of the images and other related aspects. The goal is to make the pipeline that produces the test images clear to the reader.

Regarding the GAN-based solutions used in this section, they were selected having in mind the importance of the task performed, the performance, and the availability of software. For each solution the objective is to obtain 3 levels of image quality (Lo, Mi, Hi) to evaluate the perceptual impacts of the artefacts generated by GAN based solutions.

4.2.1. Enhanced Super-Resolution Generative Adversarial Networks

In section 3.2. a GAN that performs super resolution [52] is described, however we could not find an official publicly available software for this solution. Considering other choices, it was selected the ESRGAN proposed by Xintao Wang *et al.* in [79] for which an implementation was available. The ESRGAN model is an improved version of the solution reviewed in section 3.2 and introduces some new tools. Among those, a Residual-in-Residual Dense Block (RDDB) illustrated in Figure 35, the removal of batch normalization layers and an improved discriminator network based on Relativistic average GAN (RaGAN) [80].

The RDDB has a higher number of layers and connections when compared with the generic residual block. Which leads to deeper and more complex structure. These blocks also employ residual learning to different levels, *i.e.*, the skip connections are not only between the input and output, but rather connect different points of the block to each other.

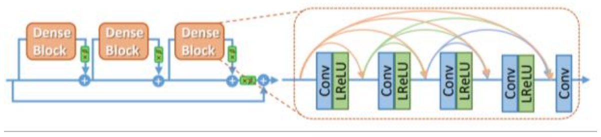


Figure 35 - Architecture of the Residual in Residual Dense Block (RRDB)

The lack of batch normalization layers in the network helped reduce the computational complexity of the training task while mitigating the artefacts produced by these layers, when coupled with a dense network trained under the GAN framework.

When it comes to the discriminator the architecture is the same as described in section 3.2 with the relativistic GAN loss. This loss, instead of estimating the probability of a given sample being *fake* or *real*, it estimates if the sample is more *real* than a randomly sampled fake data. This change leads to a more stable network that creates higher quality images.

Regarding the generator used, it follows an SRResNet [52] architecture and has 32 layers of RRDB blocks. This architecture is joined with a loss function (11) that balances a perceptual loss (L_{precep}), a content loss (L_1) and the relativistic adversarial loss (L_G^{Ra}). The perceptual loss it is computed using a VGG19 network, the content loss is a 1-norm distance between the generated image and the ground truth, and the adversarial loss is obtained using the generator/discriminator pair described previously.

The weight of these losses in the total generator loss (L_G) is controlled by λ and η, hyperparameters.

$$L_G = L_{percep} + \lambda L_G^{Ra} + \eta L_1 \tag{11}$$

Three variants of the proposed solution were used in this study. These variants are defined to obtain three distinct level of artifacts caused by the content generation capabilities of the network. This is done by changing the weight (λ) of the adversarial loss (L_G^{Ra}). Which means that the adversarial loss varies in importance, and thus may lead to images with higher apparent quality but less real, i.e., more fake. The different ESRGAN models used can be defined as follows, where perceived quality of the generated images increasing from Lo to Mi, and from Mi to Hi:

- ESRGAN Lo: $\lambda = 5 \times 10^{-2}$
- ESRGAN Mi: $\lambda = 10^{-2}$
- ESRGAN Hi: $\lambda = 5 \times 10^{-3}$ (pretrained model)

The ESRGAN Lo and ESRGAN Mi models were trained specifically for this study, while the ESRGAN Hi was a pretrained model provided by the authors. The models trained followed closely the recommendations made by the authors to maintain a fair comparison.

As recommended, the training set used was the DIV2K [81] as. More specifically, the subset of training data that refers to high resolution images, and the corresponding low-resolution images obtained with a bicubic kernel and a downscale factor of 4. And for validation is used the entirety of Set14 [59].

These datasets follow some pre-processing that is done to decrease access times and reduce training duration. This pre-processing consists of creating patches of size 480x480 for the HR images and patches of size 120x120 for the LR images.

Other relevant training parameters used are the Adam optimizer [65] with $\beta_1 = 0.9$, $\beta_2 = 0.99$ and a learning rate of 10^{-4}. The mini-batch size that was set to 4 instead of the recommended 16 due to hardware limitations. In Annex A.2 the full configuration files are available.

After the models were trained, the next step is to obtain the super resolution images for the test set defined in section 4.1. To perform the super resolution task on the test set, low-resolution (LR) images are first obtained. These LR images are obtained as recommended by the authors with a Matlab function that performs resize using a bicubic kernel and a downscale factor of 4.

These LR images were fed to the ESRGAN Lo, Mi and Hi and the result are HR images that are reconstructions of the original HR image with different levels of artefacts depending on which model originated the image. As mentioned in the previous section the test material needs to be of size 940x880 and so a crop is performed to the HR GAN-generated images.

It is important to note that the crops mentioned in the previous section are done after the super resolution image is obtained and not before. In Figure 36 it is shown an illustration of the pipeline that creates the test images for the ESRGAN solution.

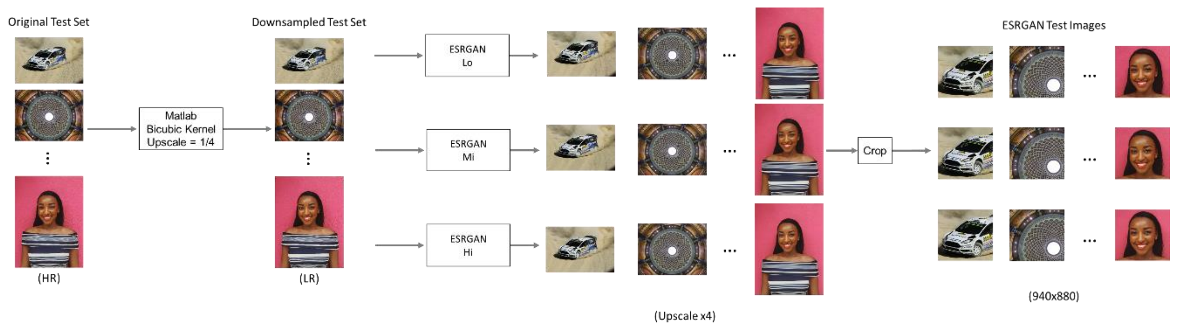


Figure 36 - Visual representation of the pipeline used to obtain ESRGAN test images

4.2.2. High-Fidelity Generative Image Compression

The compression solution presented in section 3.1. [30] also doesn't have a publicly available software. Moreover, recently a GAN-based image compression solution was proposed which is considered more efficient and is nowadays recognized as highly efficient [82]; this solution is called High-Fidelity Generative Image Compression (HiFiC), was proposed by Fabian Mentzer *et al.*, and the software is public available.

The HiFiC solution uses some of the same tools present in SC mode of the older model, such as, the use of a Conditional GAN. Moreover, the model uses the additional information, such as, class labels or a semantic map, to better control the generated content and to obtain better results in more challenging scenarios.

A new objective function is introduced in the HiFiC model (12). This objective function is built with the goal of minimizing the rate-distortion trade-off. That is to say, the network goal is to achieve the lowest possible combined bitrate ($r(y)$ where y is the latent space representation of the image x) and distortion ($d(x,x')$, where x' is the reconstructed image from the latent space representation y and x is the test image). As any other generator network, the objective function of the HiFiC's generator has a term that relates to the discriminator network ($\log(D(x',y))$).

$$\mathcal{L}_{EGP} = \mathbb{E}_{x\sim px}\left[\lambda r(y) + d(x,x') - \beta \log\big(D(x',y)\big)\right] \tag{12}$$

The hyperparameters λ and β both influence the bitrate, however in the experiments β is fixed and λ is replaced by an adaptive step that allows the model to follow more closely the desired bitrate. This adaptive step (λ') is given as a function (13) of a new hyperparameter (r_t) which serves as a threshold to decide what value of λ is adequate ($\lambda^{(a)}$ or $\lambda^{(b)}$).

$$\lambda' = \begin{cases} \lambda^{(a)}, & r(y) > r_t \\ \lambda^{(b)}, & otherwise \end{cases} \quad \text{where} \quad \lambda^{(a)} \gg \lambda^{(b)} \tag{13}$$

The hyperparameter r_t relates directly to the desired bitrate in bpp and, typically, is a descriptive way to identify a given HiFiC model. Along with the image compression solution, 3 pre-trained models, trained under the same condition and with 3 distinct target bitrates $r_t = \{0.14, 0.3, 0.45\}$ bpp, were available.

For the subjective assessment experiments, it was crucial that the 3 models presented visible differences in quality, from minor to severe degradations. However, after close inspection of the images produced by these pretrained models it was clear that the difference in perceptual quality between the models with $r_t = 0.14$ bpp and $r_t = 0.3$ bpp was almost nonexistent. Moreover, the pretrained models lead to images with very few artifacts. Therefore, a new HiFiC model was trained, which is more restrictive in terms of bitrate, in this case, the target rate was set to 0.06 bpp.

The training of this new model follows the methodology proposed by the authors of the proposed image compression solution. The training has two steps: i) training the model without the adversarial training, that is with a simple objective function and ii) training with discriminator, that is with the complete loss function (12): In both cases, the distortion function (14) used is a weighted sum of a perceptual loss ($d_P = LPIPS$) and a MSE loss.

$$d = k_M \mathrm{MSE} + \mathrm{k_P} \mathrm{LPIPS} \tag{14}$$

Where k_M and k_P are hyperparameters that balance the weight of each component. These hyperparameters are fixed in the experiments described by the authors and therefore, will also be fixed in the same way in our training.

Besides changing the target rate r_t it was also necessary change $\lambda^{(a)}$ while maintaining $\lambda^{(b)}$ fixed, as illustrated by the authors experiments. The decision of adequate $\lambda^{(a)}$ value was not a obvious one and lacking the time and insight of the model to make a better hyperparameter tunning it was chosen by making a linear regression of the values of $\lambda^{(a)}$ used for each r_t presented in the paper. This procedure resulted in $\lambda^{(a)} = 2.44$ and that was the value used while training the HiFiC model.

The training dataset used was the same as the experiments described in the paper, that is the Coco 2014 [77] dataset. It is also important to note that besides the configurations changed in order to set the desired target rate it was also changed some configurations in the `model.py` file due to hardware limitations. These changes are shown in Annex A.2. Configuration file of model HiFiC Lo and reflect a reduction in the batch size used.

With the training concluded it was clear that the model did not reach the desired target rate as it is depicted in *Table 6*. This could be improved with different $\lambda^{(a)}$, but since the results obtained with this model were already different from those resulting of models with $r_t = \{0.14, 0.3\}$ bpp there was no need to train another model with different hyperparameters. It is also important to note that for the pretrained models the target bitrate is not always achieved this is due to the model's rate-distortion trade-off. For some images the compression task might be easier and the degradation allows for lower bitrates in other images the model refuses the lower bitrates since the degradation is greater.

Table 6 – Bitrates per image in JPEGAI dataset obtained with the HiFiC models.

ID	Name	Bitrate (bpp)			
		$r_t = 0.06$	$r_t = 0.14$	$r_t = 0.3$	$r_t = 0.45$
00001	Matterhorn	0,142	0,214	0,382	0,587
00002	Racing car	0,099	0,164	0,33	0,532
00003	Sardinia festival	0,102	0,154	0,283	0,415
00004	Rotunda of Mosta	0,172	0,245	0,461	0,705
00005	Las Vegas sign	0,089	0,123	0,219	0,305
00006	Train	0,117	0,174	0,313	0,478
00007	Windows	0,133	0,184	0,343	0,482
00008	Transmission towers	0,079	0,136	0,276	0,456
00009	Port	0,2	0,292	0,535	0,849
00010	Curiosity Rover	0,075	0,129	0,256	0,38
00011	Bell pepper	0,192	0,275	0,497	0,74
00012	Woman	0,095	0,162	0,354	0,59
00013	Dog	0,135	0,205	0,369	0,532
00014	Beach	0,077	0,126	0,247	0,375
00015	Harbor	0,09	0,136	0,248	0,364
00016	Ponytail	0,052	0,099	0,232	0,384

Following the same nomenclature presented in the previous section, and going from lower to higher perceived quality the models used this subjective assessment were as follows:

- HiFiC Lo: $r_t = 0{,}06$ bpp
- HiFiC Mi: $r_t = 0{,}14$ bpp (pretrained)
- HiFiC Hi: $r_t = 0{,}3$ bpp (pretrained)

And, once again, after the compressed images are obtained the crops are performed to obtain the images that will be evaluated.

4.2.3. Image and Video Restoration and Compression Artefact Removal using a NoGAN Approach

Once again, the model described for the artefact removal task in section 3.3. [68] did not have publicly available software. After considering several models with public software available, the selected solution for artefact removal was ArNet [83], a more recent and efficient model which had a public available implementation along with a pretrained model. The main novelty of ArNet is the NoGAN [84] technique, a new methodology to train the GANs. The NoGAN training pretrains the generator and discriminator with straightforward, fast, and reliable conventional loss functions, which are then trained together in a normal GAN setting. However, the number of iterations that the generator and discriminator are trained is limited, resulting in a model that is faster to train and with higher quality (less artefacts).

To perform subjective assessment, 3 distinct levels of quality should be obtained as in the other GAN based solutions. In this case, the network used to obtain the images was the pretrained available network, and to achieve different levels of quality, the target JPEG decoded quality was varied and consequently the difficulty of the artifact removal task. It is important to note that this model is trained on a limited dataset, namely with images with similar degradations. In this case, the quality factor was changed. The quality factor (QF) values used for the experiment and corresponding nomenclature used in the experiment are as follow:

- ArNet Lo: $QF = 7$
- ArNet Mi: $QF = 14$
- ArNet Hi: $QF = 21$

The JPEG compressed images were obtained using the Python PIL module (save function). These images are then fed to the pretrained ArNet and to the resulting reconstructed images is performed the cropping to fit the images to the test environment.

4.2.4. Benchmarking Image Compression: HEVC Intra

To have a benchmark, the subjective test included images compressed with the High Efficiency Video Coding (HEVC) codec. This was included to provide a better understanding of HiFiC's performance when compared with a traditional solution. In order to have a fair comparison between solutions the HEVC images were obtain for similar bitrates by varying the quality parameter (QP) as presented in Annex A.3**.**

The software used to compress the images was the official implementation of Rec. ITU-T H.265 | ISO/IEC 23008-2 High efficiency video coding. This codec works over YUV format files and as such there is the need to convert the images from RGB (PNG format) to YUV colorspace and after compression the opposite conversion. This was done using the FFmpeg and running the following commands:

- Conversion from PNG to YUV:

```
fmpeg -hide_banner -f rawvideo -pix_fmt yuv444p10le -video_size <image size> -
                  i <input image path> -y <output image path>
```

- Conversion from YUV to PNG:

```
ffmpeg -hide_banner -i <input image path> -pix_fmt yuv444p10le -
vf scale=out_color_matrix=bt709 -color_primaries bt709 -color_trc bt709 -
colorspace bt709 -y <output image path>
```

4.3. Subjective Test Methodology

This section describes the subjective assessment methodology, namely the design of the test and user interface as other important information, such as the grading scale. The design choices are based on the ITU-R Recommendation BT.500-14 [85] and some previous work on crowdsourcing subjective assessment.

4.3.1. Subjective test design

Regarding the subjective quality test design, laboratory studies are nowadays well established due to their accuracy and repeatability. In this case, the laboratory environment can be well controlled, e.g. lighting conditions, distance to the screen, and all subjects screened and well trained. However, these tests are time consuming and often difficult to perform, especially considering the COVID pandemic situation. In contrast, crowdsourcing studies can be easily performed. Moreover, the reliability of crowdsourcing has been proven to be good enough, if outlier removal is performed [86]. Considering these reasons, a crowdsourcing quality evaluation procedure was used in this work.

A pairwise comparison methodology was selected to achieve high discriminatory power and thus high accuracy. In this case, the 2-alternative forced-choice (2AFC) setting was used, where pairs of images are shown to the observers. After, the observers must select the image in each pair with higher quality. The main reasons for this choice are: 1) the selected GAN-models introduce small changes in the perceived quality which requires a method that allows users to discriminate these differences in an efficient and fast easy; 2) a pairwise comparison methodology is also easier (an faster) for the subjects to perform comparisons, which is rather suitable for a crowdsourcing experiment that is always remote without any close interaction.

The pairwise comparison method is an indirect measurement, and thus quality scores cannot be obtained simply by averaging ratings as in other test designs, e.g. ITU-T double stimulus impairment scale test. An algorithm is required to compute the underlying quality scores values from the binary decisions of each subject, the methodology used in this experiment is described in 4.4.2.

4.3.2. Pairwise comparison test design

The test material consists of 8 different reference images (n) which were obtained using 10 different conditions for the GAN solutions under evaluation. These 10 conditions correspond to 3 levels of quality (k) for each task (compression, super resolution, and artefact removal) and the original image. Moreover, a conventional solution for image compression, in this case, HEVC Intra was included as benchmark.

To have a complete design of the subjective test, that is every image is compared with every other image, each subject would need to perform $C_2^{n \times k} = 3160$ evaluations, which would lead to a test with an unacceptable long duration. To lower the number of comparisons and thus the time needed for a subject to perform the pairwise comparison test, an image is not compared to all other images and thus the test follows an incomplete design. The pairs of images compared were carefully selected as described next:

- *Within content across test conditions*: For each content (reference and all GAN processed images), all possible pairs are considered; this means that all images produced for different test conditions are compared among themselves for the same content, which allows to achieve accurate and reliable psychovisual scores. The number of comparisons associated are $n \times C_2^k = 360$.
- *Cross-content across test conditions*: Cross-content pairs can increase the accuracy of the obtained psychovisual scores and allow to have a single homogenous scale (with the same meaning) between different content [87] with the same meaning. Therefore, cross-content pairs were included to obtain more accurate psychometric scaling results and thus compare user opinion for different contents. These cross-content pairs were selected by fixing the quality level of the GAN solution under evaluation (compression, denoising and super-resolution) and comparing images with adjacent ID's. This translates to $(k-1)(k-2) = 72$ additional pairs.
- *HEVC Intra benchmark*: The HiFiC solution was compared against a conventional compression solution, namely, HEVC Intra. The pairs used compared each HiFiC image (24 in total) with a HEVC image obtained for similar bitrates. Which translates to 24 comparisons, leading to a total of 456 comparisons.

Note that considering the incomplete design of the test, it is necessary to compute the psychovisual scores using a computational model which supports such incomplete design, i.e. that not all scores are available.

Regarding the duration of the subjective test assessment, two sessions of maximum 30 minutes each were defined. This division was done to avoid subjects getting too fatigued to make accurate judgements. To avoid fatigue, the subjects are asked to take a break of at least 15 minutes if the session duration exceeds the recommended time.

Regarding the display environment in which the subjective test is performed some restrictions were imposed. In this case, the screen resolution and size were restricted to a minimum of 1920x1080 and 13 inches, respectively.

4.3.3. Crowdsourcing platform

A crowdsourcing test environment which supports such pairwise comparisons was developed to perform the subjective test and is publicly available (https://github.com/martafilipa/subjective_test). In this case, a crowdsourcing web platform with three main components was built: HTTP server, database, and clients with a common web browser. The architecture of such system is shown in Figure 37. The web server provides the graphical environment and any information that the subject needs to perform the test in the form of HTML pages. The server also collects any information transmitted by the subject and connects to the database to store the data.

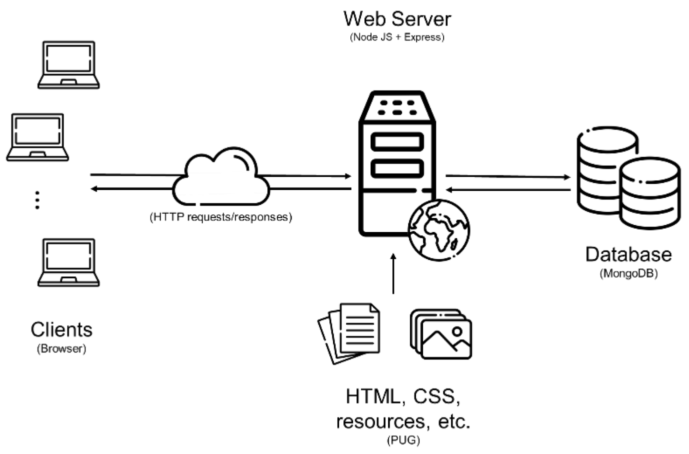
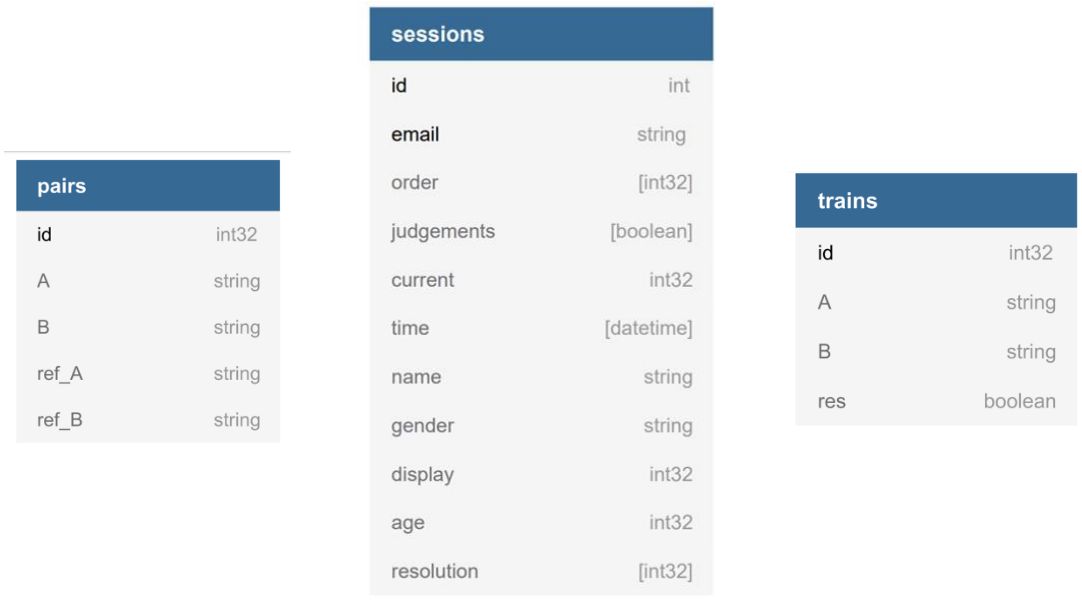


Figure 37 – Architecture of the crowdsourcing web platform

There are several frameworks that were used in the crowdsourcing web platform. In this case, due to their popularity, the server uses Node JS [88] and the Express framework [89]. The event-driven architecture of Node JS in conjunction with Express framework made the process of creating the crowdsourcing platform, easy and intuitive.

In combination with the Node JS, a simple MongoDB database [90] was deployed. This database contained 3 collections:

- *Pairs collection*: Serves the purpose of organizing the images in the test set and the pairs that would be shown during the evaluation. Also, includes information relating to the ID of the reference images in the pair, which is useful when restricting the viewing order to consecutive pairs containing the same reference image. This translates to a schema as shown in Figure 38 (left).
- *Session collection*: Stores the information regarding each subject, namely personal information (name, age, gender, and email) and viewing conditions (display size and resolution). It is also stored a specific order in which the test pairs shown to the subject and the judgments make in each comparison as well as the time-stamped of the binary decision. In Figure 38 (center), is depicted the schema that this collection follows.
- *Training collection*: Includes the image pairs used for the training phase and the information relevant to the training phase, namely which image is the least influenced by the GAN introduced artefacts, which image is the correct choice in the context of this assessment. The schema of this collection is presented in Figure 38 (right).



pairs	
id	int32
A	string
B	string
ref_A	string
ref_B	string

sessions	
id	int
email	string
order	[int32]
judgements	[boolean]
current	int32
time	[datetime]
name	string
gender	string
display	int32
age	int32
resolution	[int32]

trains	
id	int32
A	string
B	string
res	boolean

Figure 38 – Schema of the collections used in the subjective assessment web application. Namely, pairs collection (left), sessions (center) and training (right). Where in bold are the keys of each collection.

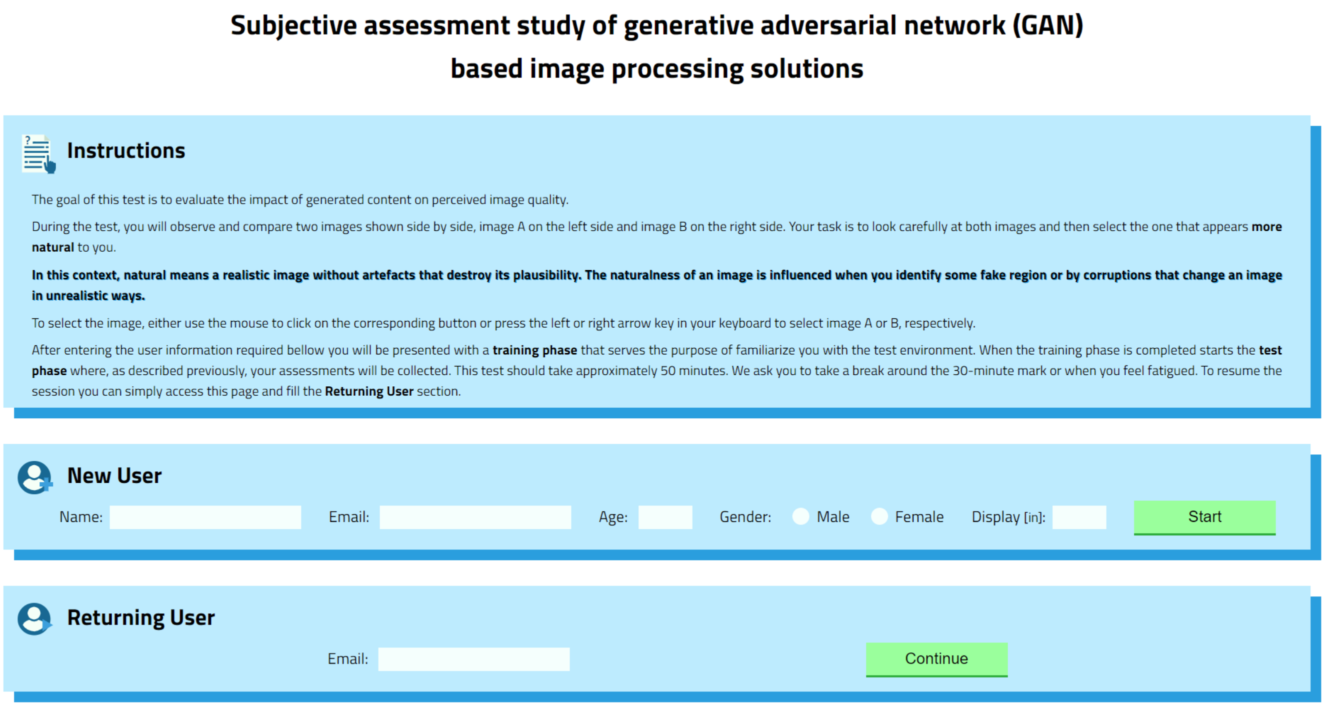
Regarding the user interface served from the server to the client it was developed using a template engine called Pug [91]. This engine allows the deployment of HTML pages with complete abstraction from the content that would be served to the client. The HTML pages contain placeholders for the content that were filled accordingly by the server when responding to a given request from some subject. Finally, to organize and style the resources in the web page layouts it was used Cascading Style Sheets (CSS).

The crowdsourcing web platform has the following flow: the subject connects by making a `HTTP GET` request to the root (`/`) and the server presents the HTML page shown in Figure 39. This page provides a description of the test and instructs the subject of the required steps. The client can select one of two options in this page:

- *Client is a new subject starting the subjective test:* In the case of a new subject, the client is required to provide some relevant personal information: name, email, age, gender, and display size in inches. When the subject clicks on the start button a `HTTP POST` request is made and the information introduced by the subject along with the resolution of the screen (determined from a script in the HTML page) is added to a new entry in the database session collection. In this new entry, the server computes and stores the order in which the test images will be shown to the client. The order is pseudo-random and follows the restriction that consecutive pairs cannot contain the same reference material. The subject is identified by the server using a session cookie, which is also stored in the database and serves as a primary key.
- *Client is returning for a second session of the test*: The other option provided by this page is to

resume a subjective test, namely to continue the evaluation of the subjective test. The client must provide the email used before, the server accesses the database with the session cookie and the test is resumed from the point where it was left.

It is also important to note that there are some limitations imposed to prevent the test from starting. One of these limitations is regarding the screen resolution that is set to a minimum of 1920x1080p. This limitation guarantees that the images are presented side by side as intended by the test design, all parts fully visible. Another limitation regards to the minimum display size, set to 13 inches to ensure that displays which are considered difficult to observe the content degradations are not used.



Subjective assessment study of generative adversarial network (GAN) based image processing solutions

Instructions

The goal of this test is to evaluate the impact of generated content on perceived image quality.

During the test, you will observe and compare two images shown side by side, image A on the left side and image B on the right side. Your task is to look carefully at both images and then select the one that appears **more natural** to you.

In this context, natural means a realistic image without artefacts that destroy its plausibility. The naturalness of an image is influenced when you identify some fake region or by corruptions that change an image in unrealistic ways.

To select the image, either use the mouse to click on the corresponding button or press the left or right arrow key in your keyboard to select image A or B, respectively.

After entering the user information required bellow you will be presented with a **training phase** that serves the purpose of familiarize you with the test environment. When the training phase is completed starts the **test phase** where, as described previously, your assessments will be collected. This test should take approximately 50 minutes. We ask you to take a break around the 30-minute mark or when you feel fatigued. To resume the session you can simply access this page and fill the **Returning User** section.

New User

Name: Email: Age: Gender: Male Female Display [in]: Start

Returning User

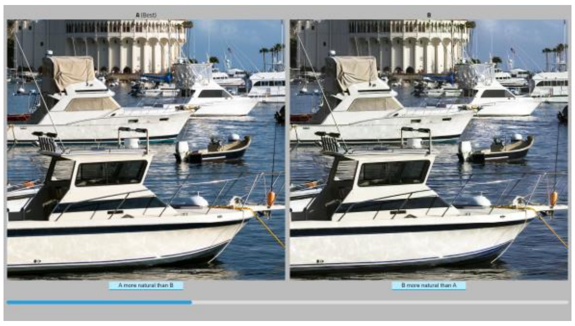
Email: Continue

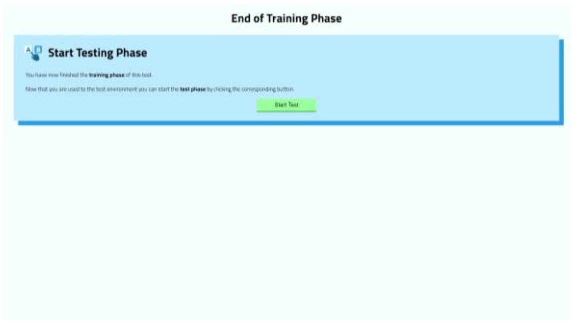
Figure 39 -Homepage of the subjective test web application

When the subjective test starts, all subjects have to perform a small training phase, which serves the purpose of familiarize the subject to the web quality evaluation environment. The user interface of this phase, depicted in Figure 40 (left), has the same layout of the user interface of the evaluation phase. However, provides additional information regarding correctness of the choice having into consideration the context of the assessment and the quality of images presented. This forces the client to select the correct image (with better quality) before continuing. The goal is to allow to familiarize the subject with some of the artefacts that will find in future image (in the evaluation phase) and ensures that the subject is understanding correctly the interface (e.g. meaning of the buttons) and the goal of the subjective test.

During the training or evaluation phase, the binary selection of an image can be done by clicking the corresponding button or by pressing the right or left arrows on the keyboard. The corresponding left arrow key is assigned to the option/button “A more natural than B” and the right arrow key to the “B more natural than A" option. Both methods result in a `HTTP POST /train` message containing the decision made.

During the training phase, the server accesses the training collection to check if the decision is correct. If correct, the next page with another pair of images is shown, otherwise the page remains the same and it is given a notification asking the client to repeat the evaluation. When the user reaches the end of the training phase, an informative page is shown that creates the appropriate separation between the training phase and the evaluation phase. When the user clicks the "Start Test" button the server responds with the first page of the evaluation phase.




End of Training Phase

Start Testing Phase

You have now finished the **training phase** of this test.

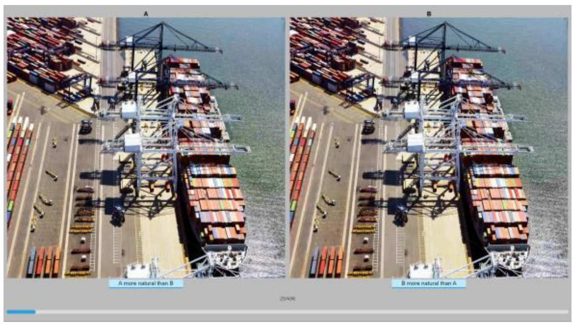
Now that you are used to the test environment you can start the **test phase** by clicking the corresponding button.

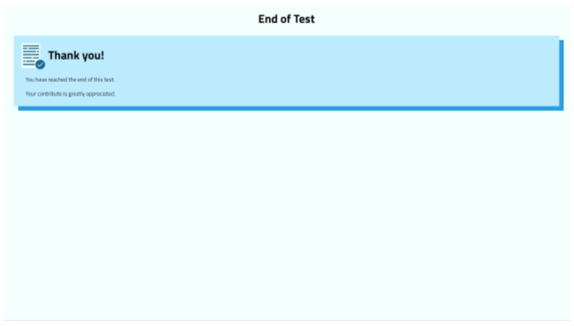
Start Test

Figure 40 – Left, training phase user interface; right, information shown in the end of training phase

The evaluation phase user interface shown in Figure 41 (left), is very similar behavior to the training user interface. In this case, the server connects to the database storing the decision and the associated timestamp to the session collection.

At any point of the subjective test a subject can stop the test by simply closing the browser window. Later, a subject can return to the exact point in which the test was stopped by returning to the crowdsourcing web platform and introduce its user information (email address) in the home page. The `POST /` with a valid email address results in the server connecting to the database to access the correct state of the test session and then the server returns the page containing the pair of images that still need to be evaluated before the pause, effectively resuming the subjective test.




End of Test

Thank you!

You have reached the end of this test.

Your contribute is greatly appreciated.

Figure 41 - Test environment of the subjective test. On the left selection screen and on the right the

end of test screen.

When the last image pair is shown to the subjects the final page is shown to the user. This page is merely informative and serves to inform the user that the test is finished, as illustrated in Figure 41 (right).

4.4. Data Processing: converting pairwise-comparison to psychometric scores

This section describes all the processing made to the binary decisions data obtained from the subjective test. First, the process that eliminates unreliable observers is described, thus removing some bad decisions that could influence the final outcome. Then, the computation of the psychometric scores is described, this means the transformation of the binary decision data to reliable scores that represent the quality of each image in the subjective study.

4.4.1. Outlier Detection

Since subjects are not supervised, it is common practice to identify and remove any outliers. In this case, subjects are mostly University students and researchers which understand the importance of this study and thus have little incentive to submit malicious decisions. Nevertheless, outlier detection was applied after the crowdsourcing experiment was completed, especially to filter subjects which do not paid much attention or misunderstand the goal of the test. There are several outlier detection procedures, but all aim to check if the responses of some subject in the study agree with the rest of the subjects, while allowing some reasonable deviation.

To perform outlier detection in single or double stimulus quality assessment tests, ratings of a subject are compared to the ratings of other subjects. However, outlier detection in pairwise comparison tests can be performed by only inspecting the decisions made by each subject [92]. In this case, to verify the reliability of a given subject, the transitivity rule is used [93] [94]. This rule states that for any given three stimuli A, B and C if A is preferred against B and B is preferred against C, then the relation is transitive and A should be preferred to C. Therefore, a subject breaks the transitivity rule when the the decisions made result in the connected triad that verifies (15), and is identified by *Batagelj et al.* [95] by 030C.

$$(A > B) \cap (B > C) \cap (C > A) \tag{15}$$

In the context of the outlier detection for a pairwise comparison test, a subject constantly breaking the transitivity rule is considered as unreliable since its assessments are contradictory. To quantify the violation of the transitivity rule, it is employed a metric called Transitivity Satisfaction Rate (R_i), also called consistency rate, defined by the frequency in which the rule was satisfied (16).

$$R_i = 1 - \frac{M_i}{N_i} \tag{16}$$

, where M_i is the number of the circular triads of (15) and N_i is the number of all possible triads. To compute R_i it was constructed an adjacency matrix from the raw data given by each user. These adjacency matrixes (A^k) have dimensions $N \times N$, where N is the number of the images (104) under evaluation and contain all the binary decisions made by the user. More specifically, the matrix is initially set to 0 in all entries. Then, during the construction process, every entry (i, j) set to 1 symbolizes the choice of i over j.Which means, if the subject k when comparing the images i and j prefers i over j in the matrix A^k the entry (i, j) would be set to 1. Due to the incomplete design of this experiment the adjacency matrixes are very sparse containing only the information that relates to the tested pairs. Pairs that compare images i and j that are not included in the test are represented in this matrix by having both entries $A(i, j)$ and $A(j, i)$ to 0, representing no preference for one over the other.

These matrixes can be interpreted as directed graphs which will facilitate the computation of R_i. The python package [96] *NetworkX* is used to transform the adjacency matrixes in directed graphs and compute the R_i values for all the subjects in the study. To transform the adjacency matrix in a directed graph is used the function `DiGraph`. Having the directed graph object created then it is applied the function `triadic_census` to the graph object with the purpose of counting the number of occurrences of triads described in *(15)*. It is also created an auxiliary matrix that helps compute the number of possible triads N_i.

As simple example of this process is presented in Figure 42 (left) a decision matrix (A) that represents the comparisons made between images A, B, C, D and E. This decision matrix translates an incomplete design, similarly to our subjective assessment, where only 7 comparisons are made. This decision matrix denotes the preference of image i over image j by setting $A_{ij} = 1$ and $A_{ji} = 0$. Furthermore, if the pair (i, j) is not compared in the assessment, then the matrix has $A_{ij} = 0$ and $A_{ji} = 0$.

This matrix can be interpreted as a directed graph where the nodes are the test images A, B, C, D and E are the nodes and the comparisons made are the edges. Moreover, the preferences displayed in the comparisons correspond to the direction of said edge. In Figure 42 (right) is shown the directed graph (G) corresponding to the matrix (A).

The nodes B, D, E follow the pattern described in *(15)*, thus breaking the transitivity rule ($M_i = 1$). From G is also possible to see that it is the only case of this cycles present, and that the number of possible decisions that break the transitivity rule are only 2 ($N_i = 2$).

This leads to a transitivity satisfaction rate of 0.5. As the number of images and comparisons increases the visual analysis of these graphs becomes more and more daunting which motivates the usage of *NetworkX* library.

	A	B	C	D	E
A	-	1	1	0	0
B	0	-	0	1	0
C	0	0	-	0	0
D	0	0	1	-	1
E	0	1	1	0	-

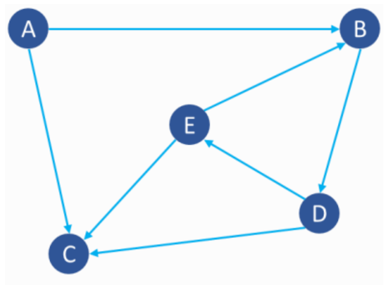


Figure 42 – Example of decision matrix (left) and corresponding directed graph (right)

Any subject that presented a transitivity satisfaction rate lower than a threshold would be considered unreliable and thus an outlier. All data from associated from an outlier subject is then removed. In this work, the threshold applied was 0.8. For the crowdsourcing experiment performed in the context of this Thesis, no users were identified as outliers, since R_i was always lower than the predefined threshold. The R_i values are shown in Table 7.

Table 7 - Transitivity Satisfaction Rate (R_i) associated with the ratings (binary decisions) of each subject.

User ID	R_i
8	0,9875
16	0,986458
25	0,986458
22	0,984375
1	0,98125
11	0,98125
2	0,979167
17	0,975
5	0,972917
32	0,972917
19	0,971875
3	0,963542
20	0,963542
12	0,960417
30	0,959375

User ID	R_i
13	0,947917
4	0,945833
29	0,945833
14	0,941667
0	0,934375
23	0,933333
10	0,93125
27	0,927083
9	0,898958
21	0,892708
31	0,889583
6	0,869792
7	0,865625
15	0,865625
18	0,844792

4.4.2. Psychometric Quality Scores Computation

The results of the paired comparison (raw binary data) experiment are often stored in a comparison matrix C, in which element c_{ij} counts the number of times a stimulus (image) i was selected as better than j. The objective is to convert such comparison matrix to interpretable quality scores, often called psychometric quality scores which are analogous to median opinion scores in double stimuli experiments. This can be performed by several methods available in the literature, where the two most widely used are based on the Thurstone [97] and the Bradley-Terry [98] observer models.

In other works [99], quality estimates are computed directly by summing up columns (or rows) of the comparison matrix. For example, quality scores can be computed as the average number of votes (wins in pairwise comparisons) that each stimuli received, more precisely, the number of votes of each image divided by the number of times the image was included in a pair and considered for evaluation. This approach is usually referred as vote counts or winning x. However, such simple method is not very suitable for incomplete experiment designs and may not produce the correct ranking. Moreover, it also lacks the ability to capture the perceptual magnitude of difference between the images in a principled way and the quality scores obtained are not in a scale that can be easily interpretable.

To obtain more reliable quality scores, it is necessary to employ a probabilistic framework that provides psychometric scaling, i.e., scales the data in a way to describe the full extent of variation of the image quality.

In this work, it was applied the Bradley-Terry Model, which given a set of past results involving elements i and j calculates the probability of i winning against j. This model is extremely helpful since this subjective assessment had an incomplete design, and the winning frequency can be reliably used as subjective quality score. There are several variations of the Bradley-Terry model, in our analysis we use the implementation provided by the python library *Choix* [100] that follow the approach presented by Maystre *et al.* [101]. More specifically, it is used the function `opt_pairwise` that given a list of comparison results computes a Maximum Likelihood Estimation (MLE) for each element present in the comparisons. Furthermore, it maximizes the likelihood function ($L(p)$), presented in (17), where w_{ij} corresponds to outcomes of the previous comparisons between elements i and j, and the probability of i or j wining is given by p_i and p_j, respectively.

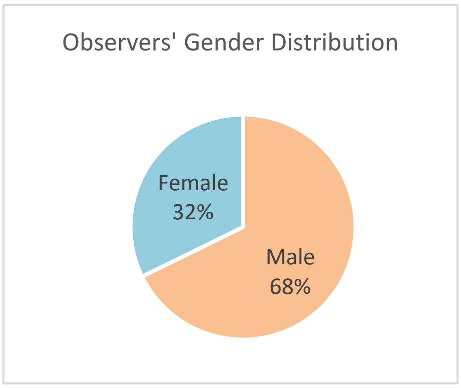
$$\mathrm{L(p)} = \sum_{i}^{n} \sum_{j}^{n} \left[w_{ij} \ln(p_i) - w_{ij} \ln\left(p_i + p_j\right) \right] \qquad (17)$$

4.5. Test Results and Analysis

This section provides an analysis of the data collected in the subjective assessment. Starting with a description of the subjects and the viewing conditions. Followed by an analysis of the winning frequency by content that will serve to identify potential bias towards specific contents. Then it is presented a winning frequency comparison between compression tasks, HiFiC and HEVC Intra, giving insight to the GAN's performance when compared with a traditional solution. Next is shown a winning frequency comparison between the different GAN-based models. And, lastly, is presented psychometric scores, the MLE values, for each GAN-based model.

4.5.1. Statistical characterization of the subjects and viewing conditions

A total of 31 subjects have participated in this study. Some subjects were experts (in image processing and compression) while others were non-experts; however, none had any prior contact with the test material. Regarding gender, 21 subjects were male and 10 were female, as illustrated by Figure 43 (left). Regarding age, it ranges between 20 and 55 and follow the distribution shown in Figure 43 (right).



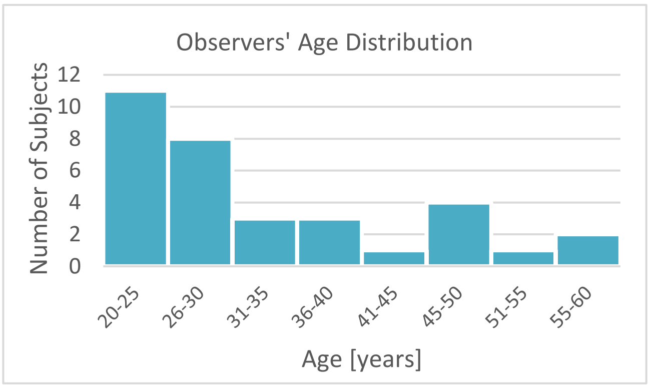
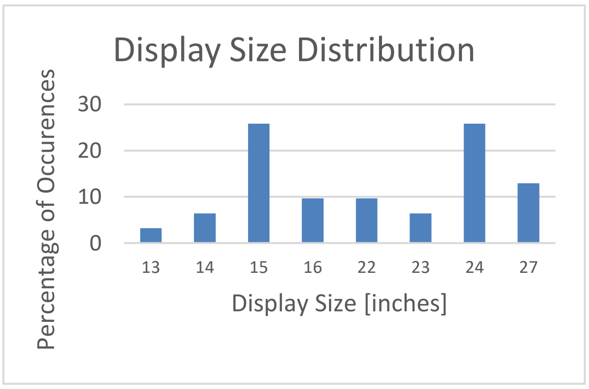


Figure 43 – Distribution of the subjects' gender (left) and age (right)

Due to the crowdsourcing nature of the survey, the viewing conditions can vary from subject to subject, most importantly the display characteristics. In this test, each subject had to submit the size and resolution of the display before performing the test. These statistics are shown in Figure 44. As shown, many subjects performed the test with a display at least of 20 inches; however, displays of 15 inches were also very common. The most common display resolution is 1920x1080, which is the minimum enforceable by the quality assessment framework.



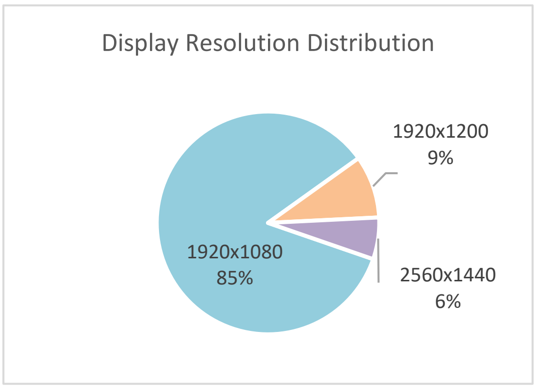


Figure 44 - Distribution of the displays size (left) and resolution (right)

Every subject had to perform 456 comparisons resulting in a total of 14 136 judgements collected during the study. Due to the high number of comparisons, the test was divided in two sessions. These 2 sessions had durations between 20 to 30 minutes depending on the user's response time and fatigue.

The average total test duration was approximately 58 minutes. From the response times gathered during the test, average response times were computed. These results are shown in Figure 45**;** most subjects take on average less than 7 seconds to make a judgment.

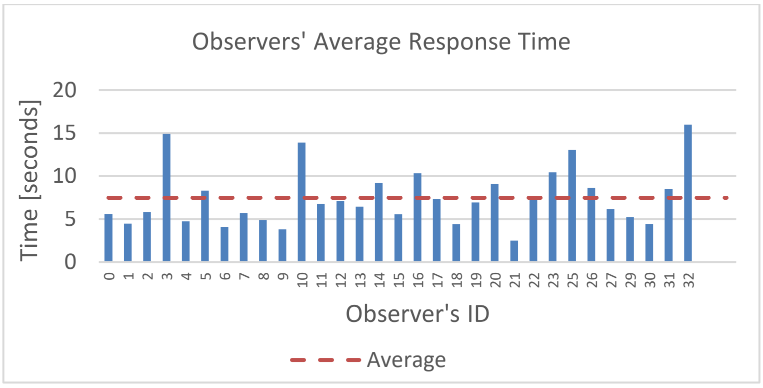


Figure 45 – Distribution of observers' average response time.

4.5.2. Winning Frequency by Content

The analysis of the results using the winning frequency as defined in Section 4.4.2. First, the subject's preference for content is computed, in this case, the winning frequency by content, i.e., the vote count is obtained by grouping the assessments by reference image. If there is no bias towards the content of the images, then the winning frequency would be 0.5 for every reference image. In Figure 46**,** the results obtained for this experiment are shown, and, overall, the images fall very close to the 0.5 winning frequency that implies no preference for content. However, some images such as *Transmission Towers* (00008) and *Woman* (00012) present a considerably lower winning frequency when compared with other images. This means that the GAN-based solutions evaluated performed worse on these images when compared with others, i.e. the artifacts are more perceptually noticeable. The main focal point of the *Transmission Towers* image is a set of towers that contain a lot of small intricate details that can be challenging for the GAN-based solutions. Regarding the *Woman* image, the main focal point is a human face, which has a complex texture and detail that is difficult to generate. On the other hand, the *Racing Car* (00002) image presents a higher winning frequency suggesting a better performance of the GAN-based solutions for this image. In this case, the image has some complex texture that can be hard to reproduce by the solutions, however those areas are not main focal point of the image, which might justify more lenient assessments.

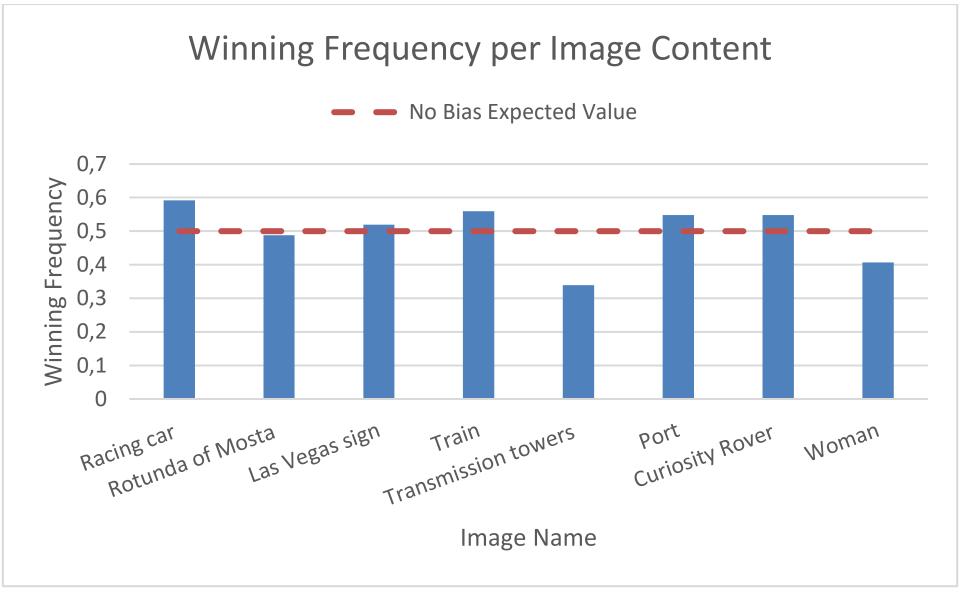


Figure 46 – Distribution of the winning frequencies by image content.

4.5.3. Winning Frequency in Compression

The winning frequency for the test pairs with an HEVC Intra and a HiFiC image for a similar bitrate, as describe in, is presented in Figure 47**.** These results show that the HiFiC GAN-based image coding solution is largely preferred when compared with HEVC Intra. This means that the HiFiC solution produces images that are considered *more natural* than one of the best conventional image codecs.

With the decrease in bitrate (Figure 47 right to left) there is an increase in preference for the images produced by the HiFiC solution. This highlights the fact that HiFiC has better performance for lower bitrates, where the generative capability of the decoder is more exploited. Also, the artefacts introduced by the GAN-based image coding solution do not make the *naturalness* of the image lower, especially compared to the usual blurring-type artifacts of HEVC Intra.

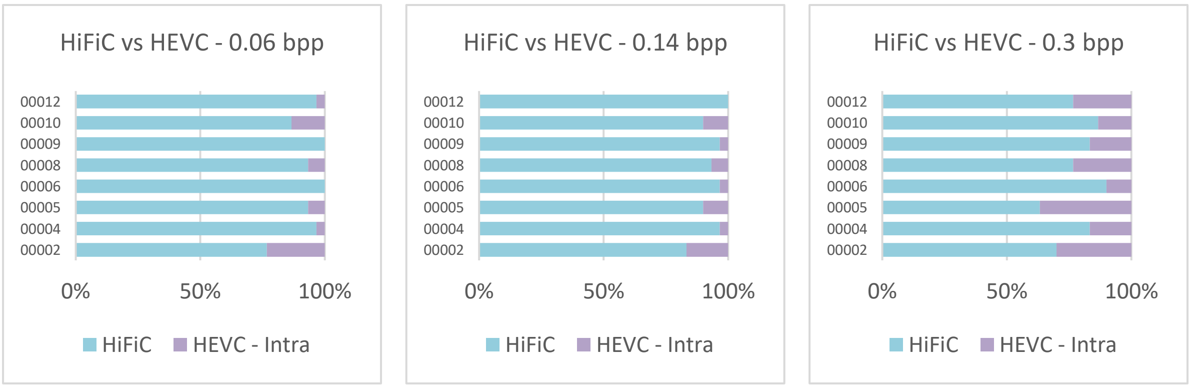
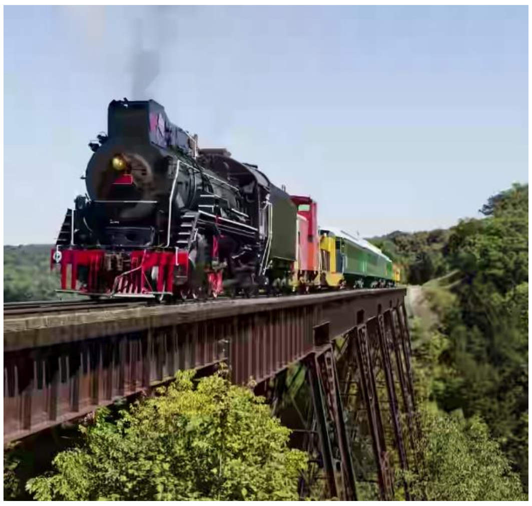
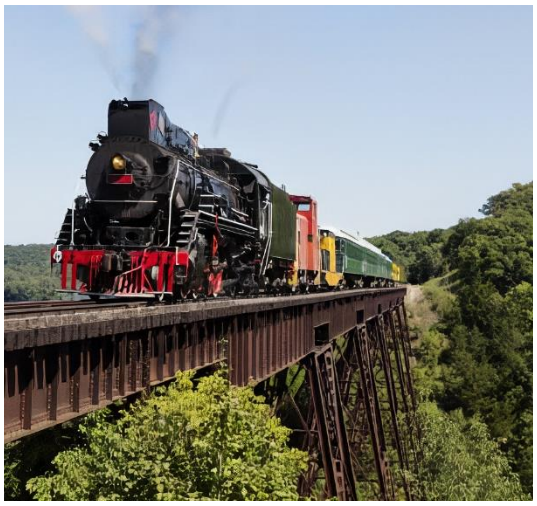
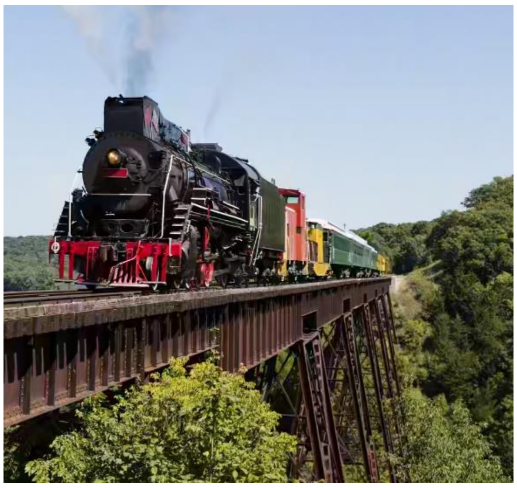
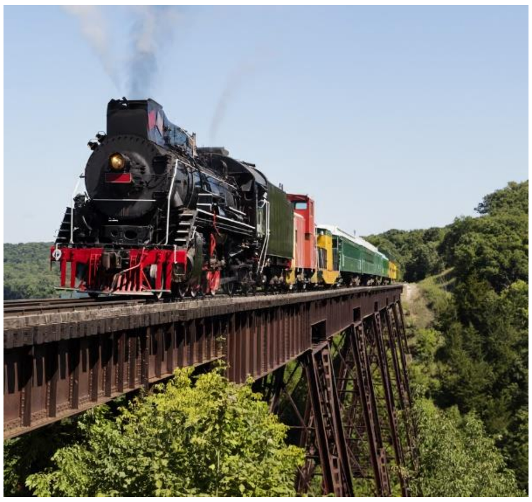


Figure 47 – Comparison of the winning frequency between HiFiC and HEVC-Intra for the same target bitrate. Left: 0.06 bpp. Centre: 0.14 bpp. Right: 0.3 bpp.

In Table 8 is showcased an illustrative example of the impact of the blurring artefacts, produced by HEVC Intra, when compared with the artefacts that derive of the generative model. This is particularly noticeable when looking at the lower bitrate scenarios (Lo). These images also help illustrate the increase in the gap of perceived quality from lower bitrates to higher, as mentioned before.

Table 8 - Side by side comparison the *Train* (00006) image compressed by HiFiC and HEVC Intra for similar scenarios.

	HEVC	HiFiC
Lo		
Hi		

4.5.4. Winning Frequency by GAN-based Solution

In this evaluation the winning frequency of each GAN-based solution is presented for all three models trained (Lo, Mi, Hi), including also the original reference image. The results are shown in Figure 48. The main difference between these models (Lo, Mi, Hi) regards the hyperparameter λ, that controls the weight of the generative content in the training process and that increases from Lo to Hi. When more weight is given to the adversarial loss the generative capabilities of the GAN based solution is augmented. As shown, there is an increasing preference for models with a larger weight for the adversarial loss, which can lead to the conclusion that the GAN-based discriminator should play a large role during training. Also, as expected, there is a large preference for the original reference image.

For the ESRGAN solution, Figure 48 (left) there is a constant increase in winning frequency, approximately doubling as the model progress from Lo to Hi. Regarding the HiFiC solution the Figure 48 (center) shows that although it follows the expected increase in winning frequency, the HiFiC Mi and HiFiC Hi models are more comparable with each other than with the HiFiC Lo model or the reference original image. Also, there is an average winning frequency of less than 40% for the reference image when compared with HiFiC images, which showcases the high performance of this solution. Finally, the ArNet solution winning frequency distribution presented in Figure 48 (right) highlights the low performance of the Lo model. This was expected since the model was trained for less severe degradations then the one used in ArNet Lo.

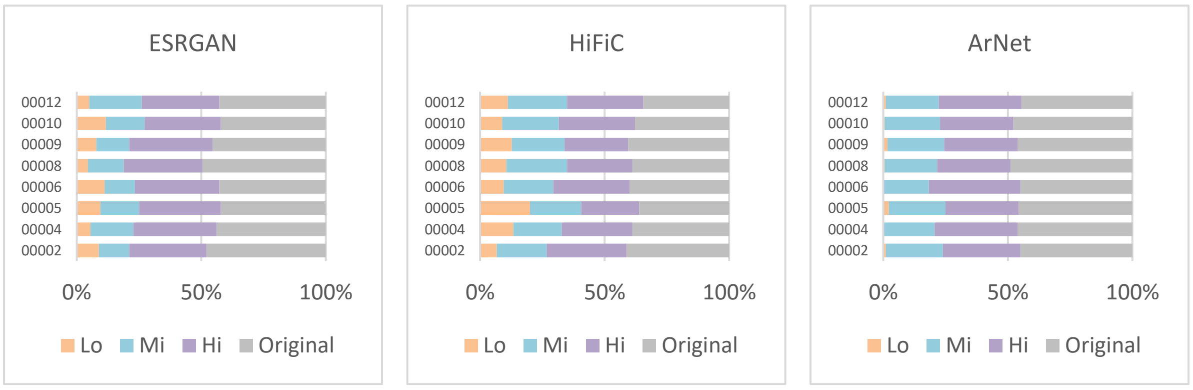


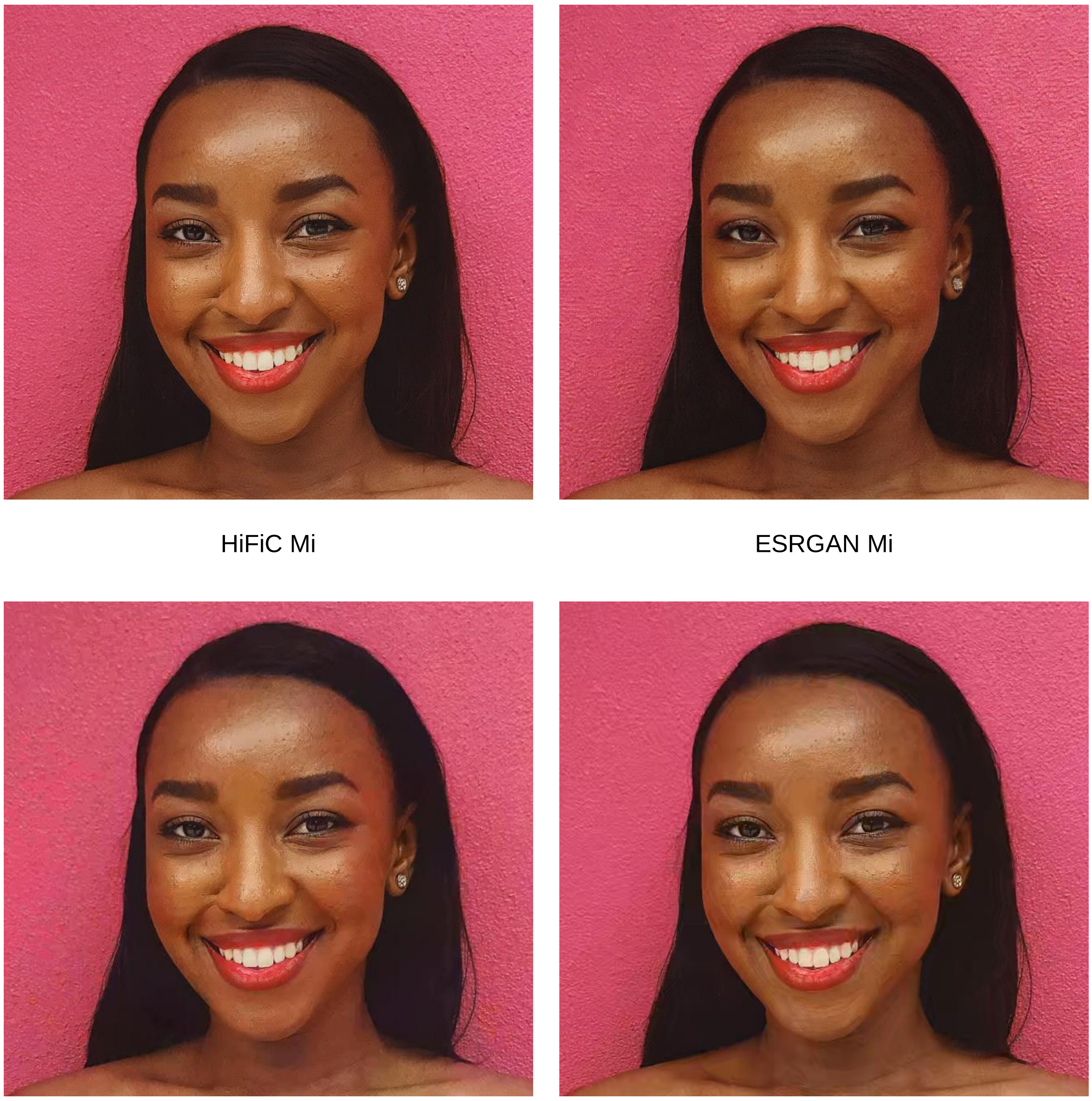
Figure 48 - Comparison of the winning frequency of each model by solution. Left: ESRGAN. center:HiFiC. Right: ArNet.

4.5.5. Psychometric scores by GAN-based Solution

Besides the winning frequency analysis, it was also performed a scaling of the raw data using the *Choix* python library. More specifically, we obtained a penalised Maximum Likelihood Estimation (MLE) using the Bradley-Terry model. These results are presented in Figure 51 and shown grouping by images' contents, also including the average MLE.

Once again it is noticeable the increase of perceived quality (measured by psychometric scores) progressively from the Lo to the Hi model, for all GAN-based solutions. However, for the specific case of the *Woman* (00012) images obtained with HEVC-Intra coding it is visible that the psychometric scores of the Mi model breaks this tendency since it has a very low value. This comes from the fact that the HEVC codec was only introduced in the assessment by comparisons with HiFiC and had no votes leading the MLE value to be very low. In Figure 49 are presented the images obtained from the *Woman* reference image using all the Mi model to further justify the MLE obtained does not translate the perceived quality of the HEVC intra Mi but is a result of the Bradley-Terry model adjusting the lack of victories of this image in the context of the study.

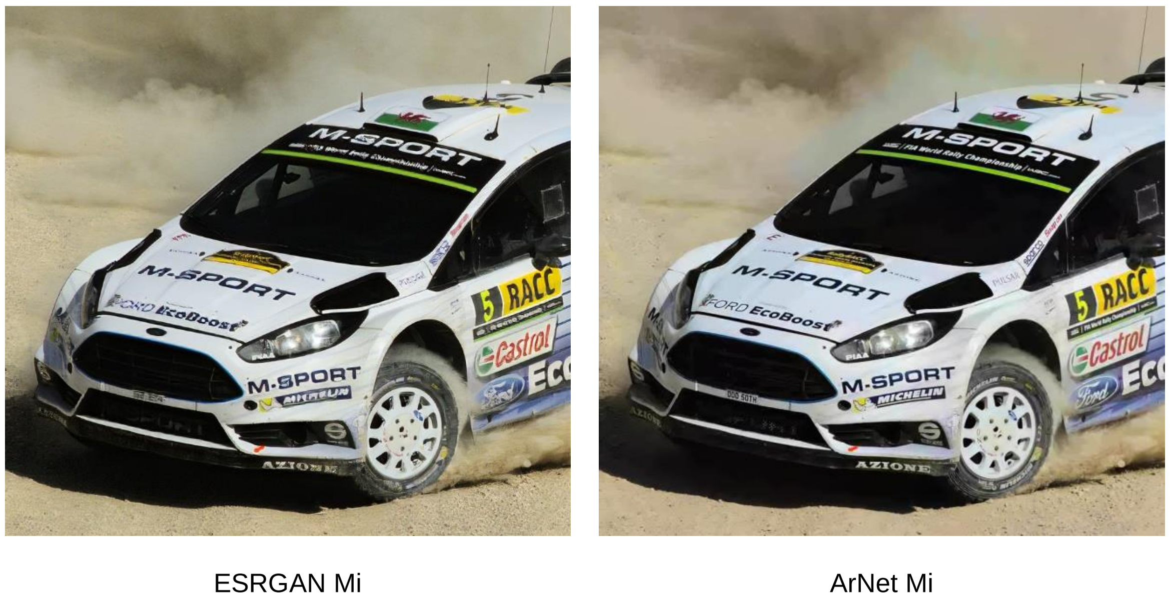
Figure 49 - Comparison of the *Woman* (00012) image obtained with different Mi models.



As shown in Figure 51, the compression solution HiFiC produced the more *natural* looking images. Once again, the variation in quality between the Mi and Hi models are very small, confirming the analysis made with the winning frequency data. The ArNet solution follows the HiFiC model when it comes to the naturalness of its images for the model Mi and Hi. However, for the Lo model the ESRGAN takes second place. The ArNet Lo produces images of poor quality as shown in the winning frequency analysis which is a fact also supported by psychometric scores.

Although, the ESRGAN has the worst image quality when compared with other GAN-based solution in this study, it is important to note that the 4x super resolution task is rather challenging when compared with other tasks in this study. In the *Racing car* (00002) and *Transmission Towers* (00008) images the ESRGAN produces very poor images which show the limitations of this solution in the reconstruction of small details. In Figure 50 is highlighted an example in which the ArNet model has better performance due to the difficulty of the super resolution task applied to the content of the *Car* (00002) image.

Figure 50 - Comparison of the *Car* (00002) images obtained with ESRGAN Mi and ArNet Mi.



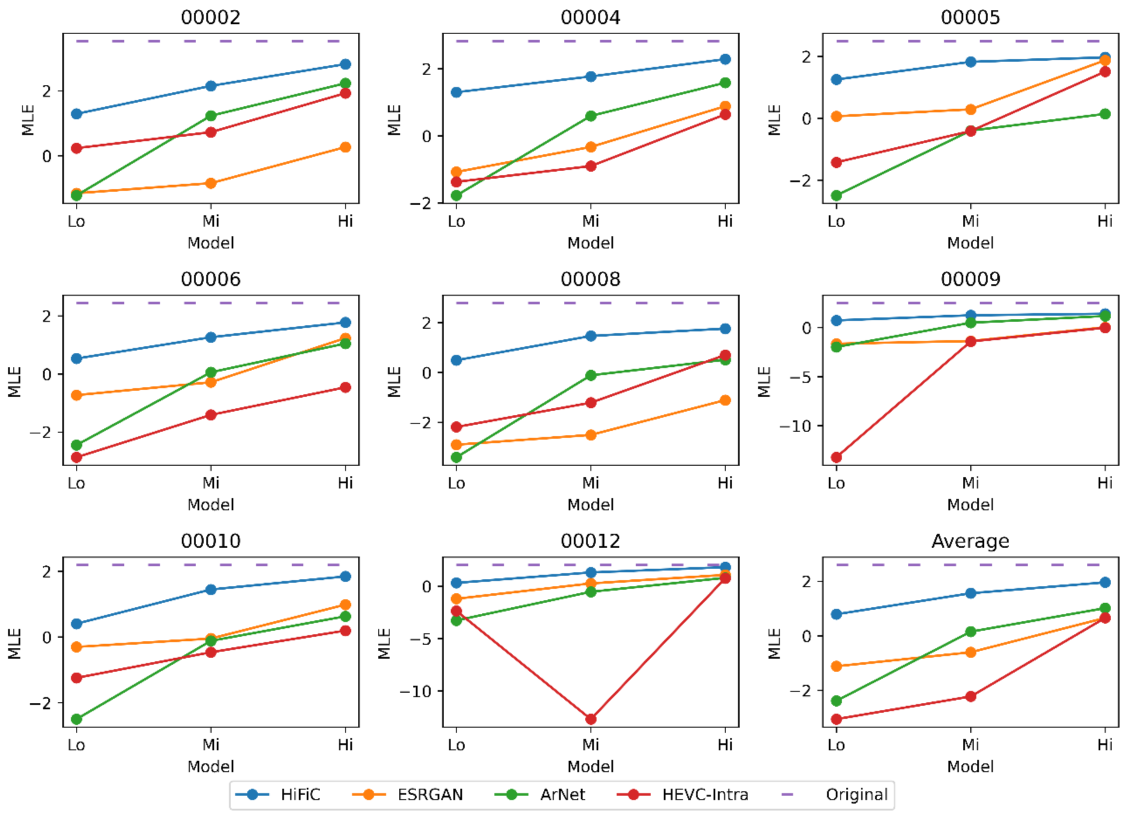


Figure 51 -Comparison between different solutions MLE by image of reference.

Chapter 5

5. Quality Metric performance evaluation

The human eye is the best judge of image quality. Subjective test assessments of image quality, if performed correctly, give a lot of insight to any solution performance. However, relying on subjective assessments to evaluate image quality in unfeasible since it is very time consuming and is always dependent on the participation of people outside the project. Which motivates the need to have objective quality metrics that can accurately quantify the image quality. In this chapter will be briefly presented a set of objective image quality metrics, will be described the procedure followed to apply these metrics and lastly will be presented an analysis of performance of the metrics.

5.1. Quality Metrics

In this section is provided a brief overview of some objective image quality metrics. The metrics studied in this chapter are divided into two sub-categories: Full-reference and no-reference. Full-reference metrics are metrics that rely on the unaltered image and use it evaluate the degradation on the tested image. On the other hand, no-reference metrics are blind to the expected image.

5.1.1. Full-Reference Metrics

There are a variety of full-reference metrics well known metrics, in this study we will focus only on the following:

- **Peak Signal-to-Noise Ratio (PSNR)** – this metric is defined by the ratio of maximum possible pixel value of the image (MAX_I) and the Mean Square Error (MSE), as depicted in (18).

$$\mathrm{PSNR} = 10 \times \log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20 \times \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \tag{18}$$

Being the MSE computed as presented by (19). Where m and n are the image's width and height respectively. Also, I denotes reference image and I' refers to the altered image.

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - I'(i,j)]^2 \tag{19}$$

- **Structural Similarity Index (SSIM) -** Unlike the previously mentioned metric, the SSIM [102] is designed with the human visual system (HVS) in mind. Moreover, this measure explores the structural information of an image. That is, the aspects of an image that define its elements. And can be calculated using (20)

$$SSIM(x,y) = \frac{\left(2\mu_x\mu_y + C_1\right)\left(2\sigma_{xy} + C_2\right)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(\sigma_x^2 + \sigma_y^2 + C_2\right)} \tag{20}$$

  Where x and y denote samples from the tested and reference images, respectively. And μ represents the mean of the signals and σ the variance. Regarding the variables C_1 and C_2, they are constants and can be calculated as shown in (21)**.** Where L is the dynamic range of the pixel values and the default values of constants K_1 and K_2 are 0.01 and 0.03 respectively.

$$\begin{gathered} C_1 = (K_1L)^2 \\ C_2 = (K_2L)^2 \; and \; C_3 = C_2/2 \end{gathered} \tag{21}$$

- **Multi-Scale Structural Similarity Index (MS-SSIM) –** The MS-SSIM [103] provides an improvement over the SSIM metric by computing it on variations of the image resolution and viewing conditions.
- **Visual Information Fidelity (VIF) –** The VIF [104] is another metric that explores the characteristics of the HVS in order to better quantify image quality. This metric exploits natural scene statistics (NSS) together with models for the distortion channel and the HVS. It computes the mutual information between the reference image and the reference image as altered by the HVS model to quantify the perceived information that the brain could under ideal conditions obtain. Similarly, calculates the mutual information between the reference image and the image affected by both the HVS model and the channel distortion model.

5.1.2. No-reference Quality Metrics

The no-reference images tested in this analysis were:

- **Blind/Referenceless Image Spatial QUality Evaluator (BRISQUE) –** This metric employs a natural scene statics model to predict the severity of distortions present in a given image. That is to say that BRISQUE [105] measures the deviation between some local luminance signal's statistics and the expected result using the natural image model.
- **Unified No-reference Image Quality and Uncertainty Evaluator** (**UNIQUE) –** This metric

consists of a DNN trained to infer the image quality. The UNIQUE [106] network has an ResNet architecture and is trained on both synthetically and realistically distorted images.

- **hyperIQA –** The hyperIQA [107] is a metric that also consists of a NN. However, the architecture of this NN differs from the simpler UNIQUE network. This NN first gets the semantic features and then makes quality predictions. It also utilizes multi-scale content to better describe the local and global distortions.

5.2. Quality Metrics Performance Evaluation Procedure

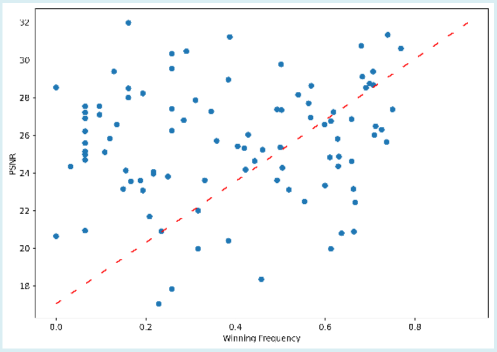
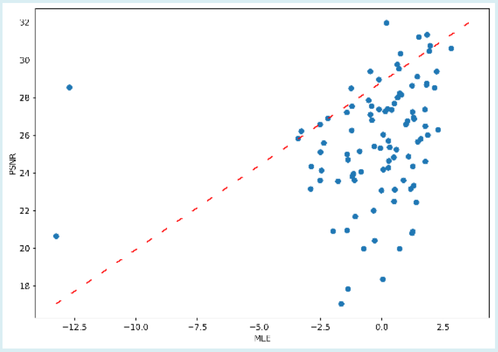
In order to evaluate the test set using the metrics mentioned previously we use a variety the python libraries. Moreover, we use the *skimage.metrics* [108] to compute the PSNR and SSIM values. The *Sewar* [109] package to obtain the values for MS-SSIM and VIF metrics.

Regarding, the no-reference images the BRISQUE implementation used was the *image-quality* [110] package. The UNIQUE values were computed using the first party software. And for the hyperIQA metric it was also used the official software.

5.3. Results and Analysis

Having obtained all the previously mentioned metric values for all images in the assessment it is performed an analysis that will give insight to the correlation between objective and subjective quality scores.

Starting with the full-reference metrics it was plotted, in Figure 52**,** each metric value as a function of the corresponding both the winning frequency and MLE. These plots can help highlight possible correlations between the objective and subjective scores. Moreover, if the sparsity of the scatter points is greater the correlation between the variable is probably lower.

Metric	***Winning-Frequency***	***MLE***
PSNR		

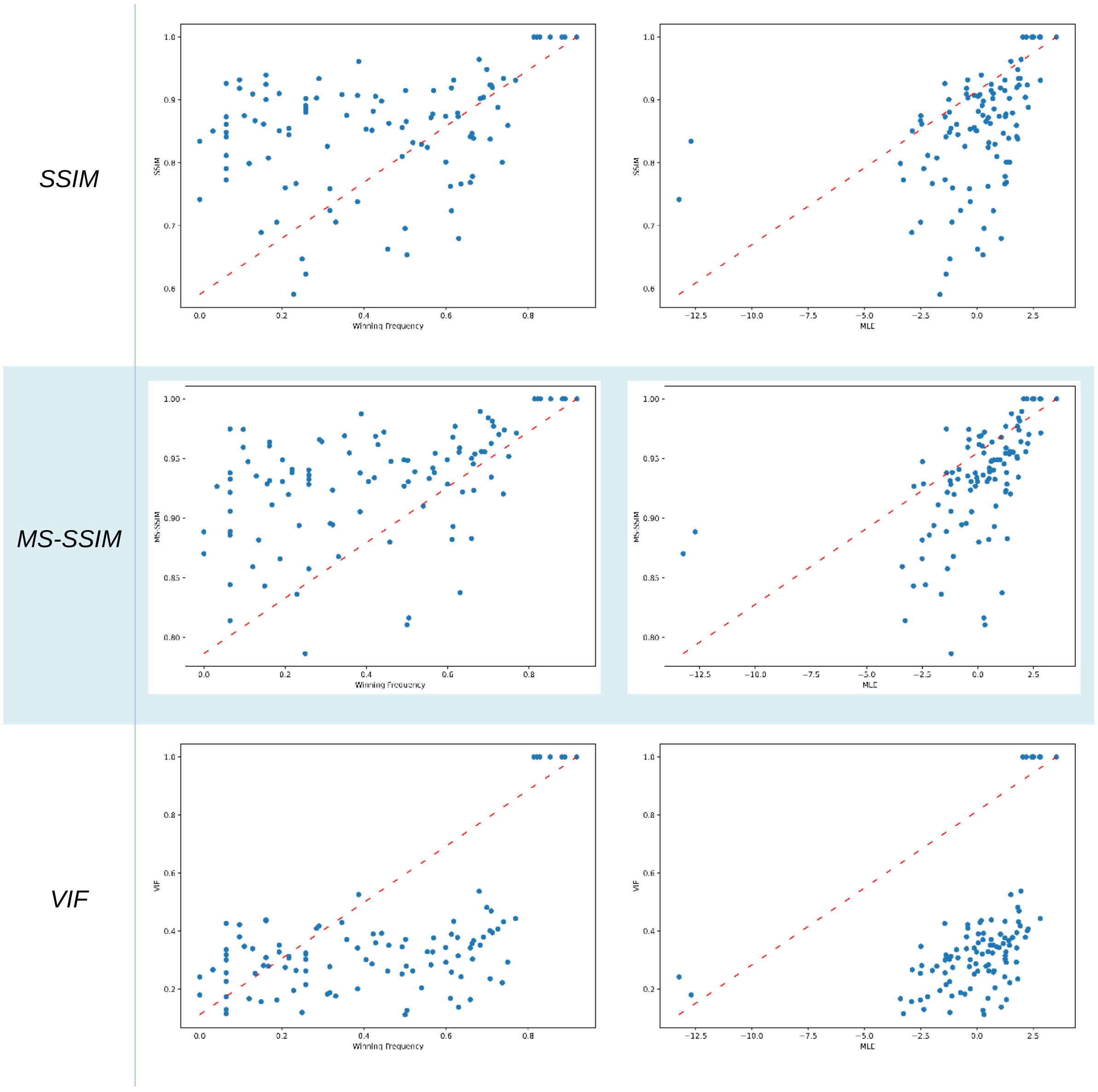


Figure 52 -Wining frequency and MLE plots as function of full-reference metrics.

From the visual data presented in Figure 52 is evident that the PSNR metric does not seem to be correlated to either the winning frequency or the MLE. The SSIM, MS-SSIM and VIF, look to be comparable in terms of sparsity. These metrics are an improvement when compared with the PSNR, however the correlation seems to be only moderate.

The intuitive notions provided by the visual data when studying the correlation between two variables is clearly insufficient and, thus, the analysis continues by computing two different correlation metrics: the Pearson Correlation Coefficient (PCC) and the Spearman's Rank Correlation Coefficient (SRCC).

The PCC is widely used when it comes to quantifying correlations between 2 variables. This coeffienct is given by (22) where X and Y correspond to each variable, and σ and cov is the standard deviation and covariance, respectively. Its values range from -1 to 1, where -1 signifies to total negative correlation and 1 total positive correlation.

$$PCC = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{22}$$

Even though SRCC is less used, it has some advantages when compared with the Pearson, namely the is much more resistant to the existence of outliers and data entry. This correlation metric is given by (23) where $R(X)$ and $R(Y)$ correspond to the rank of each variable, and σ and cov is the standard deviation and covariance, respectively.

$$SRCC = \frac{cov\big(R(X), R(Y)\big)}{\sigma_{R(X)} \sigma_{R(Y)}} \tag{23}$$

The values of PCC and SRCC were computed for every full-reference metric and are shown in Table 9**.** Has suspected, the PSNR has a weak positive correlation to both winning frequency and MLE. Which means it cannot accurately reflect the quality of the GAN-based solutions, which was expected as per the motivation of this assessment.

The SSIM, MS-SSIM and VIF are an improvement over the PSNR being barely moderate correlated to the subjective quality scores.

Table 9 - Correlation coefficients for full-reference metrics. Left: PCC. Right: SRCC.

Metric	***Winning-Frequency***	***MLE***	***Metric***	***Winning-Frequency***	***MLE***
PSNR	0.12468232379228833	0.2537244253360623	*PSNR*	0.13491130896638656	0.36734943027672273
SSIM	0.3371972436546484	0.40622694269718496	*SSIM*	0.33821214312811415	0.5414047879493028
MS-SSIM	0.46789373458246747	0.5207435643611026	*MS-SSIM*	0.5131871916870002	0.6731424045067567
VIF	0.5427486783070593	0.4603639832940975	*VIF*	0.43608015080889506	0.6135602261815855

Regarding the no-reference metrics the methodology was the same. Firstly, are obtained the plots of winning frequency and MLE with each metric. From the graphics shown in Figure 53 is evident that UNIQUE metric seems to be uncorrelated to the subjective scores. And BRISQUE and hyperIQA having apparently some correlation with the winning frequency and MLE, having BRISQUE a negative correlation and hyperIQA positive correlation.

Metric	***Winning-Frequency***	***MLE***

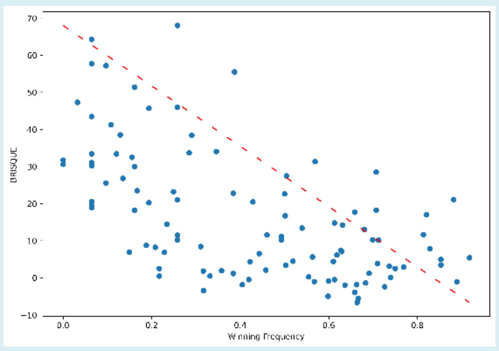
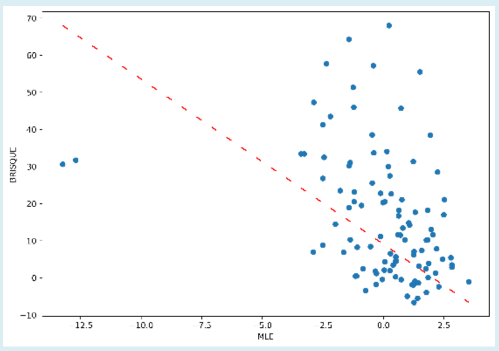
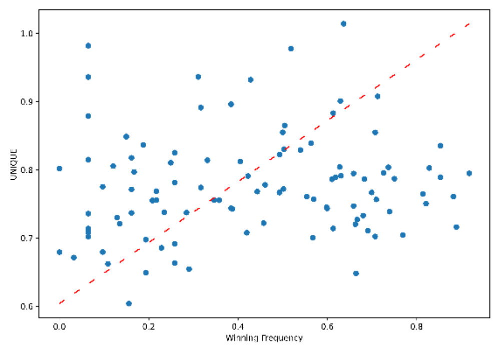
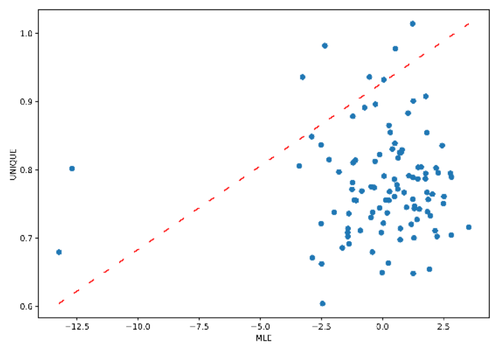
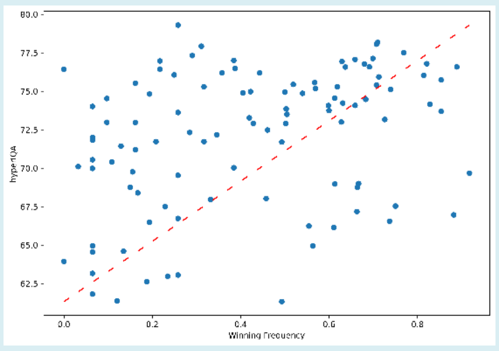
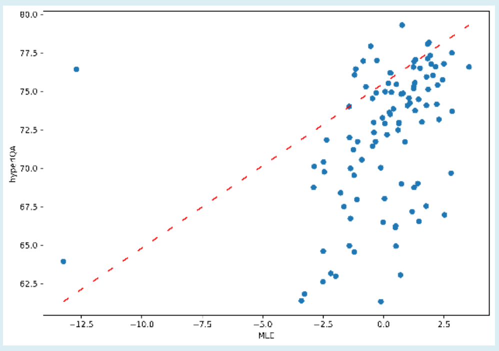
BRISQUE		
UNIQUE		
hyperIQA		

Figure 53 - Wining frequency and MLE plots as function of no-reference metrics.

Once again, the procedure is the same has the one used for the full-reference images and are calculated both PCC and SRCC scores. Analysing the correlation coefficients displayed in Table 10 is clear that BRISQUE is the best no-reference metric to evaluate this type of solutions since is the one with the highest correlation with the subjective scores. Nevertheless, this correlation is only moderate thus not constituting a very reliable assessment.

Table 10 - Correlation coefficients for no-reference metrics. Left: PCC. Right: SRCC.

Metric	***Winning-Frequency***	***MLE***
BRISQUE	-0.5857174895442	-0.3593243584985

Metric	***Winning-Frequency***	***MLE***
BRISQUE	-0.585645492201680	-0.42889149685266

UNIQUE	0.114379791057289	-0.10306306383169
hyperIQA	0.33866825570325	0.3731512515146372

UNIQUE	0.1355992920662703	-0.12975568121199
hyperIQA	0.3620766330363353	0.483697855542515

Overall, the full-reference metrics showed better correlation coefficients when compared with the no-reference metrics, like it was expected since the no-reference assessment is a more complex task.

The results enforce the idea that there is a need to develop better suited metrics for solutions that utilize generative networks, being the no-reference metric of particular relevancy since some computer vision tasks might not allow for the use of a ground truth image by design.

Chapter 6

6. Summary and Future Work Plan

The focus of this Thesis was to design, perform and analyze the outcome of a subjective quality assessment test for GAN-based solutions and measure the performance of well-known objective image quality metrics. To perform the subjective assessment study, 3 GAN-based solutions for different image processing tasks, were first identified and selected. These solutions are ESRGAN, a solution that performs super resolution, HiFiC, an image compression solution and finally ArNet, an artefact removal solution. Then, a crowdsourcing pairwise comparison methodology was applied for the results obtained were processed in order to obtain a reliable quality measure.

The results obtained provided information about the performance of each solution, namely the perceptual impact of increasing the generative capabilities of each GAN. It was found that that the *naturalness* of the GAN-derived images was found very satisfactory. Moreover, regarding image compression, the GAN based solution has outperformed the HEVC Intra codec very significantly.

Regarding the objective metrics evaluated during this study, the results show the low correlation of almost every metric to the scores obtained for the subjective experiment. Moreover, no-reference quality metrics have a lower correlation (and thus are less suitable) when compared to full-reference metrics.

With the information collected from this study a dataset was built for image quality assessment consisting of generative images and the results of the subjective evaluation. Moreover, a crowdsourcing pairwise comparison platform was developed to allow pairwise comparison subjective studies. Both the dataset and the platform are relevant contributions of this Thesis and will be public available for future work.

Considering the low performance of image quality metrics to evaluate GAN-based solutions, future work should address this limitation by: 1) study the performance of other objective quality metrics available in literature not shown in this Thesis and 2) development of a new quality metric with a higher performance for GAN-based solutions.

References

[1] D. Chai and A. Bouzerdoum, “JPEG2000 image compression: an overview,” in *Australian and New Zealand Conference on Intelligent Information Systems*, Perth, Western Australia, Australia, November 2001.

[2] D. M. Cabrita and W. Godoy, “PNG Optimization Techniques Applied to Lossless Web Images,” *IEEE Latin America Transactions,* vol. 10, no. 1, pp. 1398-1401, January 2012.

[3] M. Marcellin, “An overview of JPEG-2000,” in *Proceedings DCC 2000. Data Compression Conference*, 2000.

[4] J. Li, S. Huang, R. He and K. Qian, “Image Classification Based on Fuzzy Support Vector Machine,” in *International Symposium on Computational Intelligence and Design*, Wuhan, China, October 2008.

[5] A. Krizhevsky, I. Sutskever and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Neural Information Processing Systems*, Lake Tahoe, NV, USA, December 2012.

[6] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision,* vol. 115, no. 3, pp. 211-252, December 2015.

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, “Generative Adversarial Networks,” arXiv:1406.2661v1 [stat.ML] , June 2014.

[8] A. Radford, L. Metz and S. Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,” arXiv:1511.06434v2 [cs.LG] , January 2016.

[9] M.-Y. Liu and O. Tuzel, “Coupled Generative Adversarial Networks,” arXiv:1606.07536v2 [cs.CV], September 2016.

[10] T. Karras, T. Aila, S. Laine and J. Lehtinen, “Progressive Growing of GANs for Improved Quality, Stability, and Variation,” arXiv:1710.10196v3 [cs.NE], February 2018 .

[11] T. Karras, S. Laine and T. Aila, “A Style-Based Generator Architecture for Generative Adversarial

Networks," arXiv:1812.04948v3 [cs.NE], March 2019.

[12] Stanford University, "CS231n Convolutional Neural Networks for Visual Recognition," [Online]. Available: http://cs231n.github.io/. [Accessed 10 October 2019].

[13] R. Solomon and J. Leo van Hemmen, "Accelerating backpropagation through dynamic self-adaptation," *Neural Networks,* vol. 9, no. 4, pp. 589-601, June 1996.

[14] M. Mahsereci, L. Balles, C. Lassner and P. Hennig, "Early Stopping without a Validation Set," arXiv:1703.09580v3 [cs.LG], June 2017.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research,* vol. 15, no. 56, pp. 1929-1958, June 2014.

[16] W. Wang and J. Gang, "Application of Convolutional Neural Network in Natural Language Processing," in *International Conference on Information Systems and Computer Aided Education*, Changchun, China, July 2018.

[17] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, June 2014.

[18] C. Szegedy et. all, "Going deeper with convolutions," in *Conference on Computer Vision and Pattern Recognition*, Boston, MA, USA, June 2015.

[19] A. Soleimany, "Deep Learning for Computer Vision MIT 6.S191," January 2019. [Online]. Available: http://introtodeeplearning.com/2019/materials/2019_6S191_L3.pdf. [Accessed 21 November 2019].

[20] I. Goodfellow, Y. Bengio and A. Courville, "Deep Learning," MIT Press, November 2016.

[21] J. T. Springenberg, A. Dosovitskiy, T. Brox and M. Riedmiller, "Striving for Simplicity: The All Convolutional Net," arXiv:1412.6806v3 [cs.LG], April 2015.

[22] S. Flores, "Variational Autoencoders are Beautiful," April 2019. [Online]. Available: https://www.compthree.com/blog/autoencoder/. [Accessed 29 February 2020].

[23] A. Dertat, "Applied Deep Learning - Part 3: Autoencoders," October 2017. [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798.

[Accessed 29 February 2020].

[24] C. Doersch, “Tutorial on Variational Autoencoders,” arXiv:1606.05908v2 [stat.ML], August 2016.

[25] D. P. Kingma and M. Welling, “An Introduction to Variational Autoencoders,” *Foundations and Trends in Machine Learning,* vol. 12, no. 4, pp. 307-392, November 2019.

[26] K. Kurita, “Machine Learning Explained,” December 2017. [Online]. Available: https://mlexplained.com/2017/12/28/an-intuitive-explanation-of-variational-autoencoders-vaes-part-1/. [Accessed 6 March 2020].

[27] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen and T. Aila, “Analyzing and Improving the Image Quality of StyleGAN,” arXiv:1912.04958v2 [cs.CV], March 2020.

[28] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang and D. N. Metaxas, “StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 41, no. 8, pp. 1947-1962, August 2019.

[29] P. Isola, J. Zhu, T. Zhou and A. A. Efros, “Image-to-Image Translation with Conditional Adversarial Networks,” in *Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, July 2017.

[30] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte and L. V. Gool, “Generative Adversarial Networks for Extreme Learned Image Compression,” in *International Conference on Computer Vision*, Seoul, Korea (South), November 2019.

[31] Y. Kwon and M. Park, “Predicting Future Frames Using Retrospective Cycle GAN,” in *Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, June 2019.

[32] T. Silva, “An intuitive introduction to Generative Adversarial Networks (GANs),” January 2018. [Online]. Available: https://www.freecodecamp.org/news/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394/?utm_content=buffere27d9&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer. [Accessed 16 March 2020].

[33] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta and A. A. Bharath, “Generative Adversarial Networks: An Overview,” *IEEE Signal Processing Magazine,* vol. 35, no. 1, pp. 53-65, January 2018.

[34] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford and X. Chen, “Improved Techniques for Training GANs,” arXiv:1606.03498v1 [cs.LG], June 2016.

[35] J. Ballé, D. Minnen, S. Singh, S. J. Hwang and N. Johnston, “Variational image compression with a scale hyperprior,” arXiv:1802.01436v2 [eess.IV] , May 2018.

[36] Google, “WebP Image Format,” July 2019. [Online]. Available: https://developers.google.com/speed/webp. [Accessed 29 April 2020].

[37] F. Bellard, “BPG Image format,” April 2018. [Online]. Available: https://bellard.org/bpg/. [Accessed 28 April 2020].

[38] M. Mirza and S. Osindero, “Conditional Generative Adversarial Nets,” arXiv:1411.1784v1 [cs.LG], November 2014.

[39] H. Zhao, J. Shi, X. Qi, X. Wang and J. Jia, “Pyramid Scene Parsing Network,” in *Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, July 2017.

[40] K. He, G. Gkioxari, P. Dollár and R. Girshick, “Mask R-CNN,” in *International Conference on Computer Vision*, Venice, Italy, October 2017.

[41] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte and L. V. Gool, “Generative Adversarial Networks for Extreme Learned Image Compression,” arXiv:1804.02958v3 [cs.CV] , August 2019.

[42] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte and L. V. Gool, “Conditional Probability Models for Deep Image Compression,” in *Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, June 2018.

[43] I. Krasin, T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija, A. Kuznetsova, H. Rom, J. Uijlings, S. Popov, A. Veit, S. Belongie, V. Gomes, A. Gupta, C. Sun, G. Chechik, D. Cai, Z. Feng, D. Narayanan and K. Murphy, “Open Images Dataset,” [Online]. Available: https://opensource.google/projects/open-images-dataset. [Accessed 4 April 2020].

[44] Eastman Kodak Company, “ Kodak Lossless True Color Image Suite,” [Online]. Available: http://r0k.us/graphics/kodak/. [Accessed 3 April 2020].

[45] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter and G. Boato, “RAISE – A Raw Images Dataset for Digital Image Forensics,” ACM Multimedia Systems, Portland, Oregon, March 2015. [Online]. Available: http://loki.disi.unitn.it/RAISE/. [Accessed 4 April 2020].

[46] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth and a. B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in *Conference on Computer Vision and Pattern Recognition* , Las Vegas, NV, USA, June 2016.

[47] Z. Wang, J. Chen and S. C. H. Hoi, “Deep Learning for Image Super-resolution: A Survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* pp. 1-1, March 2020.

[48] W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue and Q. Liao, “Deep Learning for Single Image Super-Resolution: A Brief Review,” *IEEE Transactions on Multimedia,* vol. 21, no. 12, p. 3106–3121, 12 December 2019.

[49] A. Ducournau and R. Fablet, “Deep learning for ocean remote sensing: an application of convolutional neural networks for super-resolution on satellite-derived SST data,” in *Pattern Recogniton in Remote Sensing*, Cancun, Mexico, December 2016.

[50] J. Lu and W. Liu, “Unsupervised Super-Resolution Framework for Medical Ultrasound Images Using Dilated Convolutional Neural Networks,” in *International Conference on Image, Vision and Computing*, Chongqing, China, June 2018.

[51] A. Singh and J. Singh, “Review and Comparative analysis of various Image Interpolation Techniques,” in *International Conference on Intelligent Computing, Instrumentation and Control Technologies*, Kannur, Kerala, India, July 2019.

[52] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang and W. Shi, “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network,” in *Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, July 2017.

[53] K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image Recognition,” in *Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, June 2016.

[54] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” in *International Conference on International Conference on Machine Learning*, Lille, France, July 2015.

[55] K. He, X. Zhang, S. Ren and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” in *International Conference on Computer Vision*, Santiago, Chile, December 2015.

[56] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert and Z. Wang, "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," in *Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, June 2016.

[57] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv:1409.1556v6 [cs.CV], April 2015.

[58] M. Bevilacqua, A. Roumy, C. Guillemot and M.-L. Alberi Morel, "Low-Complexity Single-Image Super-Resolution based on Nonnegative Neighbor Embedding," in *British Machine Vision Conference* , Guildford, Surrey, United Kingdom, September 2012.

[59] R. Zeyde, M. Elad and M. Protter, "On Single Image Scale-Up Using Sparse-Representations," in *International Conference on Curves and Surfaces*, Avignon, France, June 2010.

[60] D. Martin, C. Fowlkes, D. Tal and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *International Conference on Computer Vision.*, Vancouver, BC, Canada, July 2001.

[61] C. Dong, C. C. Loy, K. He and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 38, no. 2, pp. 295-307, February 2016.

[62] J. Huang, A. Singh and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Conference on Computer Vision and Pattern Recognition*, Boston, MA, USA, June 2015.

[63] J. Kim, J. K. Lee and K. M. Lee, "Deeply-Recursive Convolutional Network for Image Super-Resolution," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016.

[64] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, June 2009.

[65] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv:1412.6980v9 [cs.LG] , January 2017.

[66] K. N. Satone, A. S. Deshmukh and P. B. Ulhe, "A review of image compression techniques," in

International Conference of Electronics, Communication and Aerospace Technology, Coimbatore, India, April 2017.

[67] E. Aldemir, G. Tohumoglu and M. A. Selver, “Performance Evaluation of Lossless Compression Algorithms for Medical Images,” in *Signal Processing and Communications Applications Conference*, Sivas, Turkey, April 2019.

[68] L. Galteri, L. Seidenari, M. Bertini and A. D. Bimbo, “Deep Universal Generative Adversarial Compression Artifact Removal,” *IEEE Transactions on Multimedia,* vol. 21, no. 8, pp. 2131-2145, August 2019.

[69] A. Odena, V. Dumoulin and C. Olah, “Deconvolution and Checkerboard Artifacts,” Distill, October 2016.

[70] L. Galteri, L. Seidenari, M. Bertini and A. D. Bimbo, “Deep Generative Adversarial Compression Artifact Removal,” in *International Conference on Computer Vision* , Venice, Italy, October 2017.

[71] C. Yim and A. C. Bovik, “Quality Assessment of Deblocked Images,” *IEEE Transactions on Image Processing,* vol. 20, no. 1, pp. 88-98, January 2011.

[72] H. Sheikh, Z. Wang, L. Cormack and A. Bovik, “LIVE image quality assessment database release 2,” April 2014. [Online]. Available: https://live.ece.utexas.edu/research/quality/subjective.htm. [Accessed 4 March 2020].

[73] A. Foi, V. Katkovnik and K. Egiazarian, “Pointwise Shape-Adaptive DCT for High-Quality Denoising and Deblocking of Grayscale and Color Images,” *IEEE Transactions on Image Processing,* vol. 16, no. 5, pp. 1395-1411, May 2007.

[74] C. Dong, Y. Deng, C. C. Loy and X. Tang, “Compression Artifacts Reduction by a Deep Convolutional Network,” in *International Conference on Computer Vision*, Santiago, Chile, December 2015.

[75] P. Svoboda, M. Hradis, D. Barina and P. Zemcik, “Compression Artifacts Removal Using Convolutional Neural Networks,” arXiv:1605.00366v1 [cs.CV] , May 2016.

[76] L. Cavigelli, P. Hager and L. Benini, “CAS-CNN: A deep convolutional neural network for image compression artifact suppression,” in *International Joint Conference on Neural Networks*, Anchorage, AK, USA, May 2017.

[77] T.-Y. Zitnick, L. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. Lawrence, “Microsoft COCO: Common objects in context,” in *European Conference on Computer Vision*, Zurich, Switzerland, September 2014.

[78] J. Ascenso and P. Akayzi, “JPEG AI Image Coding Common Test Conditions,” in *ISO/IEC JTC 1/SC 29/WG 1 N84035, 84th Meeting*, Brussels, Belgium, July 2019.

[79] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao and C. Change Loy, “ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks,” in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018.

[80] A. Jolicoeur-Martineau, “ The relativistic discriminator: a key element missing from standard GAN,” in *International Conference on Learning Representations*, 2019.

[81] E. Agustsson and R. Timofte, “NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.

[82] F. Mentzer, G. D. Toderici, M. Tschannen and E. Agustsson, “High-Fidelity Generative Image Compression,” *Advances in Neural Information Processing Systems,* vol. 33, 2020.

[83] F. Mameli, M. Bertini, L. Galteri and A. Del Bimbo, “Image and Video Restoration and Compression Artefact Removal Using a NoGAN Approach,” in *Proceedings of the 28th ACM International Conference on Multimedia*, New York, NY, USA, 2020.

[84] J. Antic, J. Howard and U. Manor, *Decrappification, DeOldification, and Super Resolution,* 2019.

[85] International Telecommunication Union, “Methodology for the subjective assessment of the quality of television images,” *ITU-R BT.500-14,* October 2019.

[86] F. Ribeiro, D. Florencio and V. Nascimento, “Crowdsourcing subjective image quality evaluation,” in *2011 18th IEEE International Conference on Image Processing*, 2011.

[87] E. Zerman, V. Hulusic and G. Valenzise, “Rafal Mantiuk„ Frederic Dufaux. The relation between MOS and pairwise comparisons and the importance of cross-content comparisons,” in *Human Vision and Electronic Imaging Conference, IS&T International Symposium on Electronic Imaging (EI 2018)*, Burlingame, United, 2018.

[88] S. Tilkov and S. Vinoski, “Node.js: Using JavaScript to Build High-Performance Network

Programs," *IEEE Internet Computing,* vol. 14, pp. 80-83, 2010.

[89] OpenJS Foundation, "Express JS," [Online]. Available: http://expressjs.com/. [Accessed 4 March 2021].

[90] MongoDB, Inc., "MongoDB," [Online]. Available: https://www.mongodb.com/. [Accessed 4 March 2021].

[91] "PUG," [Online]. Available: pugjs.org. [Accessed 9 March 2021].

[92] J.-S. Lee, "Paired comparison for subjective multimedia quality assessment: Theory and practice," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2013.

[93] K.-T. Chen, C.-C. Wu, Y.-C. Chang and C.-L. Lei, "A Crowdsourceable QoE Evaluation Framework for Multimedia Content," in *Proceedings of the 17th ACM International Conference on Multimedia*, New York, NY, USA, 2009.

[94] Z. Zhang, J. Zhau, N. Liu, X. Gu and Y. Zhang, "An improved pairwise comparison scaling method for subjective image quality assessment," in *2017 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2017.

[95] V. Batagelj and A. Mrvar, "A subquadratic triad census algorithm for large sparse networks with small maximum degree," *Social Networks,* vol. 23, pp. 237-243, 2001.

[96] A. Hagberg, D. Schult and P. Swart, "NetworkX - Network Analysis in Python," [Online]. Available: https://networkx.org/. [Accessed 7 June 2021].

[97] L. Thurstone, "A law of comparative judgment," *Psychological Review,* vol. 34, p. 273–286, 1927.

[98] R. A. Bradley and M. E. Terry, "Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons," *Biometrika,* vol. 39, p. 324–345, 1952.

[99] H. Ko, D. Y. Lee, S. Cho and A. C. Bovik, "Quality Prediction on Deep Generative Images," *IEEE Transactions on Image Processing,* vol. 29, pp. 5964-5979, 2020.

[10 L. Maystre, "choix," [Online]. Available: https://github.com/lucasmaystre/choix. [Accessed 4 July
0] 2021].

[10 L. Maystre and M. Grossglauser, "ChoiceRank: Identifying Preferences from Node Traffic in

1] Networks," in *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[102] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing,* vol. 13, pp. 600-612, 2004.

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

[104] H. R. Sheikh and A. C. Bovik, "Image information and visual quality," *IEEE Transactions on Image Processing,* vol. 15, pp. 430-444, 2006.

[105] A. Mittal, A. K. Moorthy and A. C. Bovik, "Blind/Referenceless Image Spatial Quality Evaluator," in *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, 2011.

[106] W. Zhang, K. Ma, G. Zhai and X. Yang, "Uncertainty-Aware Blind Image Quality Assessment in the Laboratory and Wild," *IEEE Transactions on Image Processing,* vol. 30, pp. 3474-3486, 2021.

[107] S. Su, Q. Yan, Y. Zhu, C. Zhang, X. Ge, J. Sun and Y. Zhang, "Blindly Assess Image Quality in the Wild Guided by a Self-Adaptive Hyper Network," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[108] S. V. D. Walt, J. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart and T. Yu, " scikit-image: Image processing in Python," *PeerJ,* vol. 2, p. 453, 2014.

[109] A. Khalel, "Sewar," [Online]. Available: https://github.com/andrewekhalel/sewar. [Accessed 4 July 2021].

[110] R. Ocampo, "Image Quality," [Online]. Available: https://github.com/ocampor/image-quality. [Accessed 2021 July 7].

[111] H. Ko, D. Y. Lee, S. Cho and A. C. Bovik, "Quality Prediction on Deep Generative Images," *IEEE Transactions on Image Processing,* vol. 29, p. 5964–5979, April 2020.

[112] A. Jolicoeur-Martineau, "The relativistic discriminator: a key element missing from standard GAN," *arXiv preprint arXiv:1807.00734,* 2018.

[113] K. Tsukida and M. R. Gupta, “How to analyze paired comparison data,” 2011.

[114] M. P. Ortiz, R. Mantiuk and A. Mikhailiuk, “pwcmp,” [Online]. Available: https://github.com/mantiuk/pwcmp. [Accessed 2021 June 6].

[115] M. Perez-Ortiz and R. K. Mantiuk, *A practical guide and software for analysing pairwise comparison experiments,* 2017.

[116] S. Su, Q. Yan, Y. Zhu, C. Zhang, X. Ge, J. Sun and Y. Zhang, “HyperIQA,” [Online]. Available: https://github.com/SSL92/hyperIQA. [Accessed 4 July 2021].

[117] P. Rao and L. Kupper, “Ties in Paired-Comparison Experiments: A Generalization of the Bradley-Terry Model,” *Journal of the American Statistical Association,* vol. 62, p. 194–204, 1967. July 2021.

Annexes

A.1 Subjective Test Images Cropped Areas

In Figures Figure 54 to Figure 61 are presented the crops performed on the test set as described in Section 4.1. and the corresponding cropping details used are shown in Table 11.

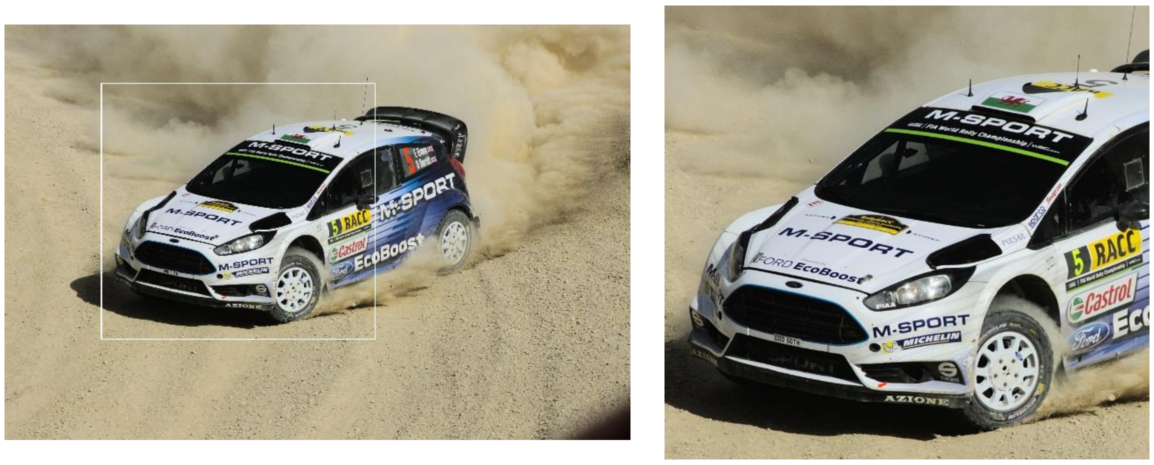


Figure 54 - Cropped area on the *Racing Car* (00002) image (left) and final test image (right)

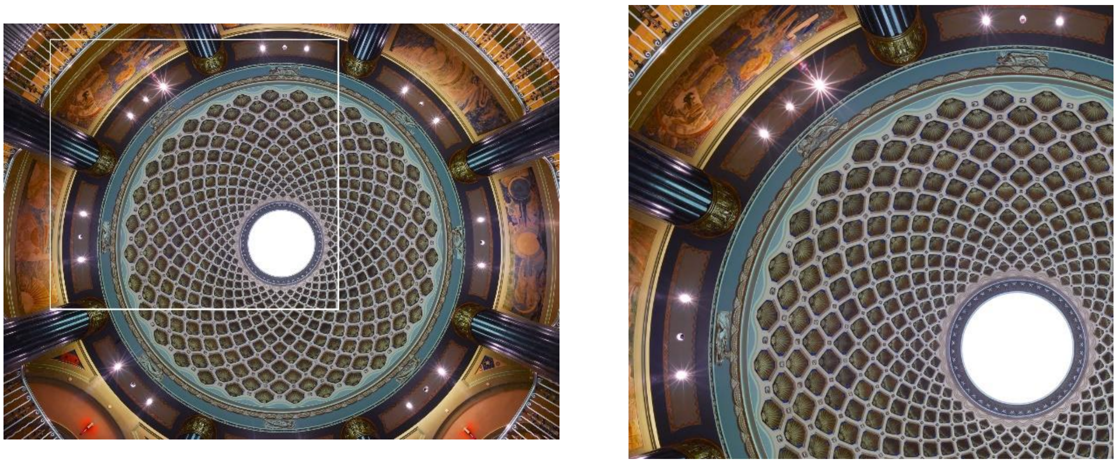


Figure 55 - Cropped area on *Rotunda of Mosta* (00004) image (left) and final test image (right)

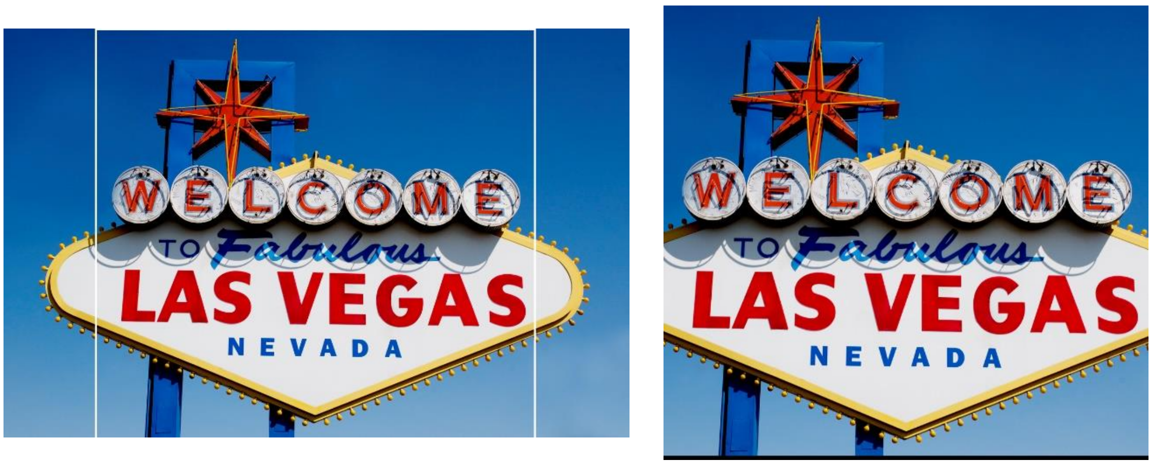


Figure 56 - Cropped area on *Las Vegas Sign* (00005) image (left) and final test image (right)

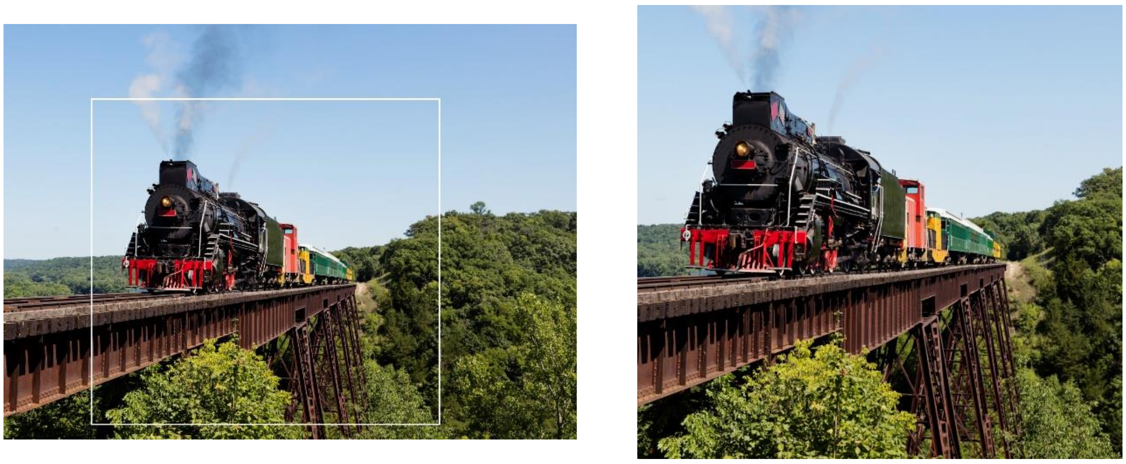


Figure 57 - Cropped area on *Train* (00006) image (left) and final test image (right)

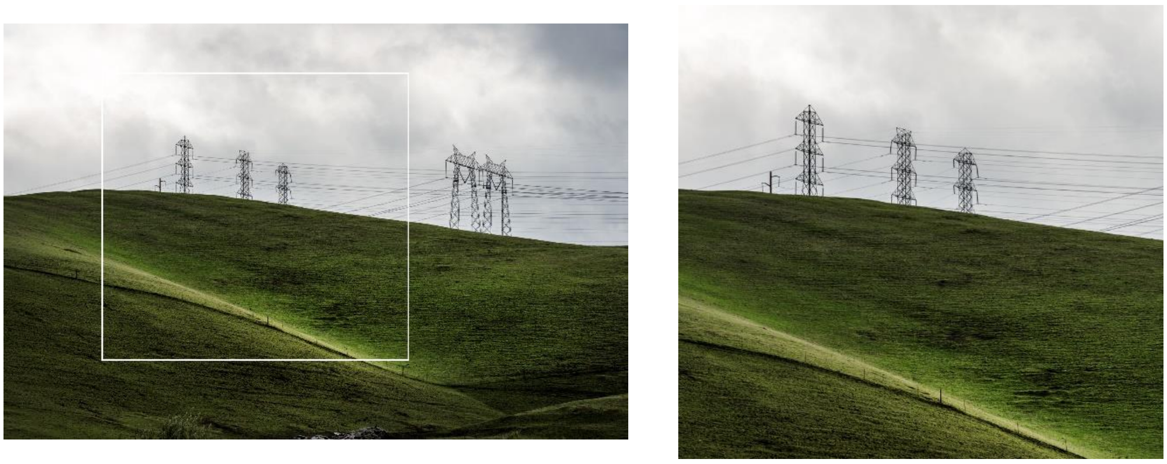


Figure 58 - Cropped area on *Transmission Towers* (00008) image (left) and final test image (right)

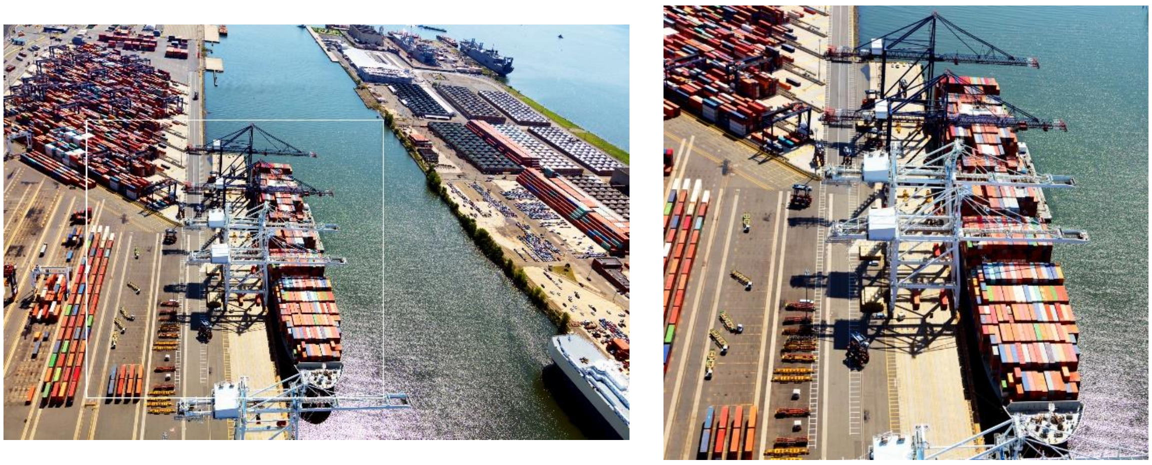


Figure 59 - Cropped area on *Port* (00009) image (left) and final test image (right)

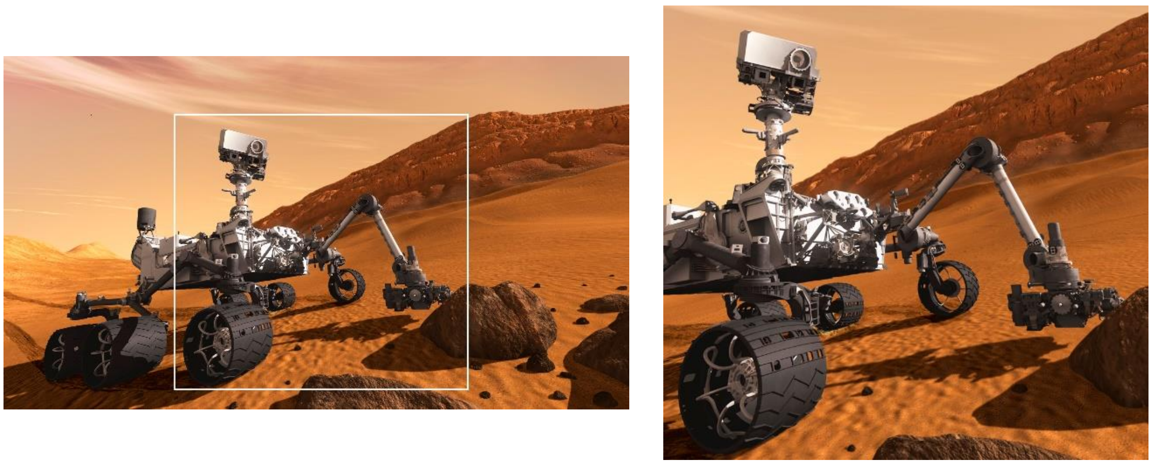


Figure 60 - Cropped area on *Curiosity Rover* (00010) image (left) and final test image (right)

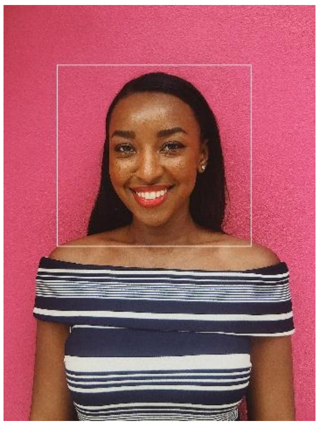
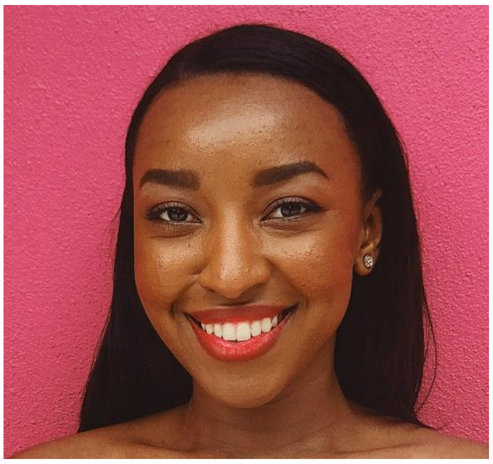



Figure 61 - Cropped area on *Woman* (00012) image (left) and final test image (right)

Table 11 – Subjective test images cropping details-

Input image	Original size	Crop size	Top left corner point (x,y)
Racing Car (00002)	2144x1424	940×880	(330, 200)
Rotunda of Mosta (00004)	1808x1352	940×880	(150, 50)
Las Vegas Sign (00005)	1336x872	940×872	(195, 0)
Train (00006)	1544x1120	940×880	(235, 200)
Transmission Towers (00008)	1912x1272	940×880	(300, 150)
Port (00009)	1976x1312	940×880	(260, 300)
Curiosity Rover (00010)	2000x1128	940×880	(545, 185)
Woman (00012)	1512x2016	940×880	(260, 290)

A.2 Models Training Configuration Files

The training configuration files representing the training details employed in the models trained during this study are presented in

A.2.1 *Configuration file of model ESRGAN Lo*

```
# general settings
name: ESRGAN_X4_DIV2K_5e-2
model_type: ESRGANModel
scale: 4
```

```yaml
num_gpu: 1  # set num_gpu: 0 for cpu mode
manual_seed: 0

# dataset and data loader settings
datasets:
  train:
    name: DIV2K
    type: PairedImageDataset
    # dataroot_gt: datasets/DIV2K/DIV2K_train_HR_sub
    # dataroot_lq: datasets/DIV2K/DIV2K_train_LR_bicubic/X4_sub
    # (for lmdb)
    dataroot_gt: datasets/DIV2K/DIV2K_train_HR_sub.lmdb
    dataroot_lq: datasets/DIV2K/DIV2K_train_LR_bicubic_X4_sub.lmdb
    filename_tmpl: '{}'
    io_backend:
      # type: disk
      # (for lmdb)
      type: lmdb

    gt_size: 128
    use_flip: true
    use_rot: true

    # data loader
    use_shuffle: true
    num_worker_per_gpu: 1
    batch_size_per_gpu: 4
    dataset_enlarge_ratio: 100
    prefetch_mode: ~

  val:
    name: Set14
    type: PairedImageDataset
    dataroot_gt: datasets/Set14/GTmod12
    dataroot_lq: datasets/Set14/LRbicx4
    io_backend:
      type: disk

# network structures
network_g:
  type: RRDBNet
  num_in_ch: 3
  num_out_ch: 3
  num_feat: 64
  num_block: 23


network_d:
  type: VGGStyleDiscriminator128
  num_in_ch: 3
  num_feat: 64

# path
path:
  pretrain_network_g: ~
  strict_load_g: true
  resume_state: ~

# training settings
train:
  optim_g:
    type: Adam
    lr: !!float 1e-4
    weight_decay: 0
    betas: [0.9, 0.99]
  optim_d:
    type: Adam
    lr: !!float 1e-4
    weight_decay: 0
    betas: [0.9, 0.99]
  scheduler:
    type: MultiStepLR
    milestones: [50000, 100000, 200000, 300000]
    gamma: 0.5

  total_iter: 400000
  warmup_iter: -1  # no warm up

  # losses
  pixel_opt:
    type: L1Loss
    loss_weight: !!float 1e-2
    reduction: mean
  perceptual_opt:
    type: PerceptualLoss
    layer_weights:
      'conv5_4': 1  # before relu
    vgg_type: vgg19
    use_input_norm: true
    range_norm: false
    perceptual_weight: 1.0
    style_weight: 0
    criterion: l1
  gan_opt:
    type: GANLoss
    gan_type: vanilla
    real_label_val: 1.0
    fake_label_val: 0.0
    #lambda value
    loss_weight: !!float 5e-2

  net_d_iters: 1
  net_d_init_iters: 0

# validation settings
val:
  val_freq: !!float 5e3
  save_img: true

  metrics:
    psnr: # metric name, can be arbitrary
      type: calculate_psnr
      crop_border: 4
      test_y_channel: false

# logging settings
logger:
  print_freq: 100
  save_checkpoint_freq: !!float 5e3
  use_tb_logger: true
  wandb:
    project: ~
    resume_id: ~

# dist training settings
```

A.2.2 Configuration file of model ESRGAN Mi

```
# general settings
name: ESRGAN_X4_DIV2K_1e-2
model_type: ESRGANModel
scale: 4
num_gpu: 1  # set num_gpu: 0 for cpu mode
manual_seed: 0

# dataset and data loader settings
datasets:
  train:
    name: DIV2K
```

```yaml
    type: PairedImageDataset
    # dataroot_gt: datasets/DIV2K/DIV2K_train_HR_sub
    # dataroot_lq: datasets/DIV2K/DIV2K_train_LR_bicubic/X4_sub
    # (for lmdb)
    dataroot_gt: datasets/DIV2K/DIV2K_train_HR_sub.lmdb
    dataroot_lq: datasets/DIV2K/DIV2K_train_LR_bicubic_X4_sub.lmdb
    filename_tmpl: '{}'
    io_backend:
      # type: disk
      # (for lmdb)
      type: lmdb

    gt_size: 128
    use_flip: true
    use_rot: true

    # data loader
    use_shuffle: true
    num_worker_per_gpu: 1
    batch_size_per_gpu: 4
    dataset_enlarge_ratio: 100
    prefetch_mode: ~

  val:
    name: Set14
    type: PairedImageDataset
    dataroot_gt: datasets/Set14/GTmod12
    dataroot_lq: datasets/Set14/LRbicx4
    io_backend:
      type: disk

# network structures
network_g:
  type: RRDBNet
  num_in_ch: 3
  num_out_ch: 3
  num_feat: 64
  num_block: 23


network_d:
  type: VGGStyleDiscriminator128
  num_in_ch: 3
  num_feat: 64

# path
path:
  pretrain_network_g: ~
  strict_load_g: true
  resume_state: ~

# training settings
train:
  optim_g:
    type: Adam
    lr: !!float 1e-4
    weight_decay: 0
    betas: [0.9, 0.99]
  optim_d:
    type: Adam
    lr: !!float 1e-4
    weight_decay: 0
    betas: [0.9, 0.99]

  scheduler:
    type: MultiStepLR
    milestones: [50000, 100000, 200000, 300000]
    gamma: 0.5

  total_iter: 400000
  warmup_iter: -1  # no warm up
  # losses
  pixel_opt:
    type: L1Loss
    loss_weight: !!float 1e-2
    reduction: mean
  perceptual_opt:
    type: PerceptualLoss
    layer_weights:
      'conv5_4': 1  # before relu
    vgg_type: vgg19
    use_input_norm: true
    range_norm: false
    perceptual_weight: 1.0
    style_weight: 0
    criterion: l1
  gan_opt:
    type: GANLoss
    gan_type: vanilla
    real_label_val: 1.0
    fake_label_val: 0.0
    #lambda value
    loss_weight: !!float 1e-2

  net_d_iters: 1
  net_d_init_iters: 0

# validation settings
val:
  val_freq: !!float 5e3
  save_img: true

  metrics:
    psnr: # metric name, can be arbitrary
      type: calculate_psnr
      crop_border: 4
      test_y_channel: false

# logging settings
logger:
  print_freq: 100
  save_checkpoint_freq: !!float 5e3
  use_tb_logger: true
  wandb:
    project: ~
    resume_id: ~

# dist training settings
```

A.2.3 Configuration file of model HiFiC Lo ($r_t 0.06\ bpp$)

```
"""Configurations for HiFiC."""

from . import helpers

_CONFIGS = {
    'hific': helpers.Config(
        model_type=helpers.ModelType.COMPRESSION_GAN,
        lambda_schedule=helpers.Config(
            vals=[2., 1.],
            steps=[50000]),
        lr=1e-4,
        lr_schedule=helpers.Config(
            vals=[1., 0.1],
            steps=[500000]),
        num_steps_disc=1,
        loss_config=helpers.Config(
            # Constrain rate:
            #   Loss = C * (1/lambda * R + CD * D) + CP * P
            #       where
            #          lambda = lambda_a if current_bpp > target
            #                   lambda_b otherwise.
            CP=0.1 * 1.5 ** 1,  # Sweep over 0.1 * 1.5 ** x
            C=0.1 * 2. ** -5,
            CD=0.75,
            target=0.06,  # This is $r_t$ in the paper.
            lpips_weight=1.,
            target_schedule=helpers.Config(
                vals=[0.20/0.14, 1.],  # Factor is independent of target.
                steps=[50000]),
            lmbda_a=0.1 * 0.409 * 2. ** -5,
            lmbda_b=0.1 * 2. ** -1,
            )
        ),
    'mselpips': helpers.Config(
        model_type=helpers.ModelType.COMPRESSION,
        lambda_schedule=helpers.Config(
            vals=[2., 1.],
            steps=[50000]),
        lr=1e-4,
        lr_schedule=helpers.Config(
            vals=[1., 0.1],
            steps=[500000]),
        num_steps_disc=None,
        loss_config=helpers.Config(
            # Constrain rate:
            #   Loss = C * (1/lambda * R + CD * D) + CP * P
            #       where
            #          lambda = lambda_a if current_bpp > target
            #                   lambda_b otherwise.
            CP=None,
            C=0.1 * 2. ** -5,
            CD=0.75,
            target=0.06,  # This is $r_t$ in the paper.
            lpips_weight=1.,
            target_schedule=helpers.Config(
                vals=[0.20/0.14, 1.],  # Factor is independent of target.
                steps=[50000]),
            lmbda_a=0.1 * 2. ** -6,
            lmbda_b=0.1 * 2. ** 1,
            )
        )
}


def get_config(config_name):
  if config_name not in _CONFIGS:
    raise ValueError(f'Unknown config_name={config_name} not in '
                     f'{_CONFIGS.keys()}')
  return _CONFIGS[config_name]


def valid_configs():
  return list(_CONFIGS.keys())
```

A.3 HEVC-Intra Configurations

In Table 12 is presented the quality parameter used for each level of target quality.

Table 12 -Values of quality parameter used when compressing images with HEVC-Intra for each target quality.

ID	Lo	Mi	Hi
00001	43	41	37
00002	40	37	33
00003	41	37	32
00004	40	38	34
00005	40	37	31
00006	43	40	36
00007	36	33	28
00008	40	38	34
00009	44	42	39
00010	39	35	30
00011	41	39	34
00012	38	36	32
00013	40	37	32
00014	40	37	32
00015	40	37	33
00016	37	33	29