INSTITUTO TECNOLÓGICO DE BUENOS AIRES – ITBA

ESCUELA DE INGENIERÍA Y GESTIÓN

AUTOENCODERS AND PRINCIPAL COMPONENT ANALYSIS: Proposal for the generation of adversarial examples in the context of face recognition systems.

AUTHORS: Fuster, Marina (Leg. Nº 57613)

Vidaurreta, Ignacio Matías (Leg. Nº 57250)

TUTOR(S): Pierri, Alan

RESEARCH THESIS PRESENTED FOR THE DEGREE OF

COMPUTER SCIENCE ENGINEER

Place: Buenos Aires, Argentina
Date: 11/10/2021

Summary

This paper proposes a new methodology for generating adversarial examples, manipulated data to confuse classification algorithms. The objective in this case is to breach face recognition systems with a strategy based on the use of *autoencoders,* a type of neural networks, and principal component analysis

For the construction of these examples, the autoencoder's architecture had to be selected and then, principal component analysis was applied to the representations of the data in the latent space of the *autencoder*.

Numerous experimentrs were carried out to study the suitability of the proposed methodology as a plausible way of generatin adversarial examples. It is concluded that the proposal does not lead to a realistic attack scheme. Both the theoretical context and decision making that led to this conclusion are discussed in depth in this thesis.

1. Introduction

The arrival of COVID-19 took the world by surprise and caused the computer industry to reach new levels of relevance. One aspect of daily life that was particularly impacted by the pandemic was the way people worked. Companies had to modify their operating methodology to preserve the health of workers. One of the main measures was the advent of 100% remote working for non-essential employees..

This change implied a very large modification in the technological infrastructure of companies. According to a survey conducted by Forbes [34], some of the concerns of executives around the world about how the pandemic has changed their IT resources include the following:

- 85% of executives surveyed believe that investments in identity security are critical.
- 55% invested in new capacity in this area since the start of the pandemic.
- 60% increased identity-related spending due to remote working and 69% of executives expect investments in identity and access management (IAM) to increase in the coming year.

As can be seen, during the pandemic, authentication took on a very important role as a consequence of the accelerated adoption of remote working. A *McKinsey Global Institute* study indicates that, in a post-pandemic scenario, remote working will maintain four to five times higher adoption than pre-pandemic levels [36]. This raises the question of how to improve authentication security. The answer, generally recommended by the industry, is multi-factor authentication (or MFA).

Traditionally, MFA takes two forms [35]: "something you know" (passwords) and "something you have" (mobile phone, yubikey). However, in recent years, another method has been gaining popularity: biometric authentication. This type of authentication answers the question "something you are" and can take the form of iris, fingerprint or, as is the case of interest in this paper, face detection. As an example, LastPass, a well-known password manager, offers an MFA solution using facial recognition to authenticate employees in their work applications [37]. While these systems are easy to use and difficult to fool, the strongest criticism is the impossibility to modify the complementary information of the authentication system: if it were compromised, it would not be possible to use the system.

The area of biometric systems and, in particular, facial recognition systems, show great progress in this new paradigm. According to research firm *MarketsandMarkets*, the global facial recognition market is estimated to grow from USD 3.8 billion in 2020 to USD 8.5 billion by 2025 [6].

When adopting a facial recognition system, it is necessary to be cautious. In a context where the incorporation of these tools shows a growing trend, attention must be paid to the uses to which they are put. No system is free from vulnerabilities but, by better understanding its weaknesses, it is possible to work on mitigating and making the risks visible, achieving transparency and honesty towards the society that will ultimately make use of the tool.

Despite growing investment in the facial recognition sector, especially from the APAC (Asia-Pacific) area, there are concerns about the security and correct use of the technology, especially when considering the areas of greatest use today: security (including surveillance), healthcare, finance and retail. At the same time as some countries are

increasing their adoption (e.g. China), in the United States, cities such as San Francisco, Oakland and San Diego are wary of the sensitivity of biometric data and its use in law enforcement, resulting in the banning of facial recognition technology in 2019 [7].

This difference in trends provides evidence of the lack of a current regulatory framework for both security checks (on the technology and the data that can be accessed with biometric authentication) and the correct uses that should be made of this tool.

As explained in the *MarketsandMarkets* article, *deep learning* is the core technology behind facial recognition systems [6]. While *deep* neural networks are widely used, a study was published in 2014 that analyses a technical vulnerability of deep neural networks: adversarial examples [28]. In the context of face recognition system it is possible to describe adversarial examples as images modified in such a way that they cause failures in the classification or correct identification of people.

Confusing algorithms considered to be highly accurate about the identity of individuals can have serious consequences. Given the sectors where facial recognition systems are used, medical data, bank accounts and the authorship of actions that are performed every day both on the street and *online* could be at risk.

2. Theoretical Framework

2.1. Facial Recognition System

Let be $S : I \rightarrow R$ a face recognition system. S receives an image i belonging to the set I of digital face images, whose appearance is realistic for human beings and returns an answer $r \in R$, which results from the comparison of i with a set of faces S has stored. In r there are not labels corresponding to specific classifications, but the percentage of similarity of the image i with the faces stored in S.

Moreover, the oracle $O : I \rightarrow T$ receives an image $i \in I$ and returns a label $t \in T$, where T is the set of possible labels The label corresponds to the identity of the person in the image i. The results of the oracle are consistent with the identification made by natural persons. The definition of oracle was taken from [1].

2.2. Attack

An attack is defined as sending an image i^* to the system S, in such a way as to alter the correct behaviour of the system according to one (or both) of the definitions set out in Sections **2.3.** y **2.4.**

2.3. Evasion

Given an image i_e belonging to the set of images I with its respective label $t_e = O(i_e)$, evasion is defined if it is verified that, for the images stored in S, the average of the similarities of i_e with all those images i_j complying with $t_e = O(i_j)$, is below 80%. The justification for this threshold will be explained in section **5.2.2.**

2.4. Impersonation

Given an image i_i belonging to the set of images I, its respective label is $t_i = O(i_i)$ and let t_x be a label selected by the attacker such that $t_i \neq t_x$. An impersonation attack occurs if it is verified that, for the images stored in S, the average of the similarities of i_i with all those images imágenes i_j complying with $t_x = O(i_j)$, is above 80%. The justification of this threshold will be explained in section **5.2.2.**

2.5. Adversarial Example

An adversarial example is an input to a machine learning model, designed by an attacker to cause the model to return an "error". Error means a result that is different from what humans would expect.

In ***Fig. 1*** an example of misclassification is displayed. On the left there is an image of a panda bear whose classification according to GoogleLeNet provides 57.7% confidence on the panda label. By manipulating the image (with one of the many existing techniques for generating adversarial examples) the classifier determines that the image is an ape species called a gibbon, with 99.3% confidence [15].

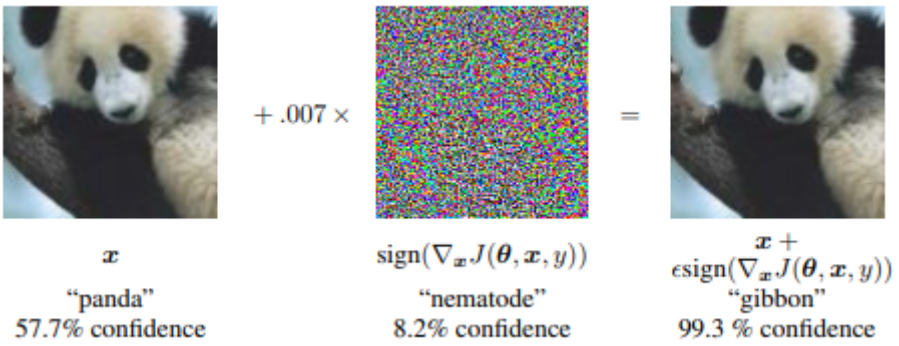


Figure 1*: Example of an adversarial example, cheating the GoogleLeNet network.*

In the context of the work, an adversarial example will be defined as an image $i^* \in I$ that allows an attack A, according to the definition of attack in section **2.2.**, on a face recognition system S.

Note that i^* may be the product of manipulation of a base image or an image generated from scratch.

2.6. Principal Componente Analysis

Working with datasets whose elements belong to a high-dimensional set, such as images, can be difficult: it is not possible to visualise the data, which can lead to problems when analysing interesting properties. However, there is an empirical hypothesis, called the Manifold Hypothesis, which states that high dimensional data often have representations in lower dimensional spaces [22]. This hypothesis supports the methods used to find correlations in information, to reorganise it and to obtain relevant data from it.

Principal Component Analysis (PCA) is a technique whose best known use is linear dimensionality reduction, in the area of machine learning. It provides the possibility, depending on the dimension to which the data is reduced, to plot a set of elements whose visualisation was previously impossible.

PCA analyses the frequent correlation between high-dimensional data, looking for a rotation of the data to describe them in terms of uncorrelated features through linear combinations [11]. The principal component search process is strongly related to singular value decomposition.

Let be N the number of observations made, p the number of features present in an observation and q the number of target features $(q \leq p)$. The solution to the principal component search is given by equation.

$$X = UDV^T \quad (1)$$

Where X refers to the data set, U is an orthogonal matrix of dimension $N \times p$, whose columns are called left eigenvectors. V is an orthogonal matrix of dimension $p \times p$ whose columns are called right eigenvectors. Finally D is a diagonal matrix of dimension $p \times p$ where the elements of the diagonal meet $d_1 \geq d_2 \geq ... \geq d_p \geq 0$ and are known as the singular values. For the complete derivation up to this expression see [14].

The columns of UD are the projections in principal components of X. If needed to obtain the first q principal components, take the dimension matrix $U_q D_q$ of dimensions $N \times q$, which corresponds to taking the first q columns of the product.

As can be seen, no eigenvalues and eigenvectors were mentioned at any point, since they only appear when working with square matrices, which means $N = p$. In this situation, the columns of V are the eigenvectors, while the eigenvalues are given by D^2.

Throughout this work **research involves dealing with non-square matrices**, more frequent and less restrictive. However, reference to **right eigenvectors** will be in terms of **eigenvectors** and singular values as **eigenvalues** (which are the squared matrices' singular values). This decision was taken because the tools used in the project employ this nomenclature and the documentation often speaks in these terms rather than the mathematically correct ones.

The singular value decomposition orders the eigenvectors (with their eigenvalues) according to the direction of maximum variability of the data. This means that there will be a greater dispersion of the data for the first components with respect to the last ones. The percentage of variability represented by each eigenvector is calculated as the ratio of the corresponding eigenvalue to the sum of all eigenvalues.

Before applying principal component analysis, the characteristics of the dataset must be standardised: zero mean and standard deviation 1. This is extremely important as the PCA process uses the covariance matrix and, in case of having characteristics whose units of

measurement are very disparate, those with greater magnitude will be favoured when analysing the weighting that explains the variability of the data [16].

2.7. Autoencoder

The *autoencoder* is a neural network whose learning is unsupervised, i.e. there is no agent to label the data. The goal is to find compact representations of a data set, from which the original elements can be retrieved with as little loss of information as possible.

This objective can be modelled as an optimization problem from the function (2), where X is the original entry and X' the output of the autoencoder. During training, the aim is to minimise this loss function.

$$L(X, X') = \|X - X'\|^2 \quad (2)$$

The architecture of an autoencoder can be thought of as two multilayer perceptron networks, where the output of the first network is connected to the input of the second. The first network is often referred to as the encoder and the second as the decoder. A visual representation of an autoencoder can be seen in ***Fig. 2*** .

.

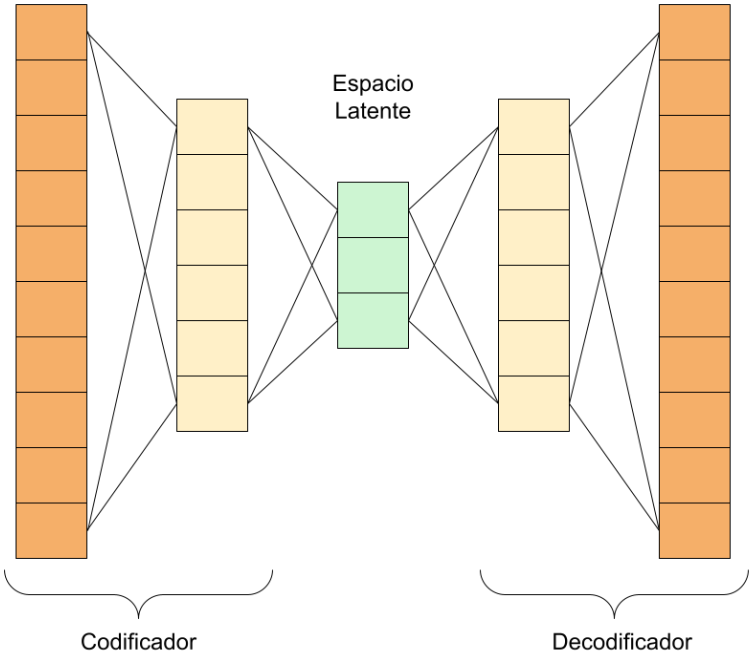


Figure 2*: Graphical representation of an auto-encoder*

When an image is sent through the *autoencoder*, it passes through the encoder which reduces its dimensionality to the corresponding size of the intermediate layer. This is called the **latent** space.

The point of greatest interest in the current work is in this latent space, as it is in this layer that the experimentation for the generation of adversarial examples takes place. This process will be explained in more detail in sections **4** and **8.**

3. State of the art

Facial recognition algorithms have achieved high accuracy in recent years. From an error rate of 4.1% in 2014, these algorithms managed to decrease their error rate to 0.08% as of today. Not only that, but the number of algorithms that achieved this accuracy also increased compared to that year [21]. This creates the need to pay special attention both to the correct functioning of the technology and to the regulations that must be formulated for its proper use.

In 2014, a study was published that analysed unusual properties in neural networks, one of which is known as adversarial examples: imperceptibly modified inputs that can cause classification failures [28]. One of the leading researchers on this topic, Nicholas Carlini, has compiled in an article on his website all the papers on adversarial examples that he has found on the *arXiv* platform [8]. This compilation shows an exponential trend in the number of published papers, which seems to be an indication of the interest surrounding this topic.

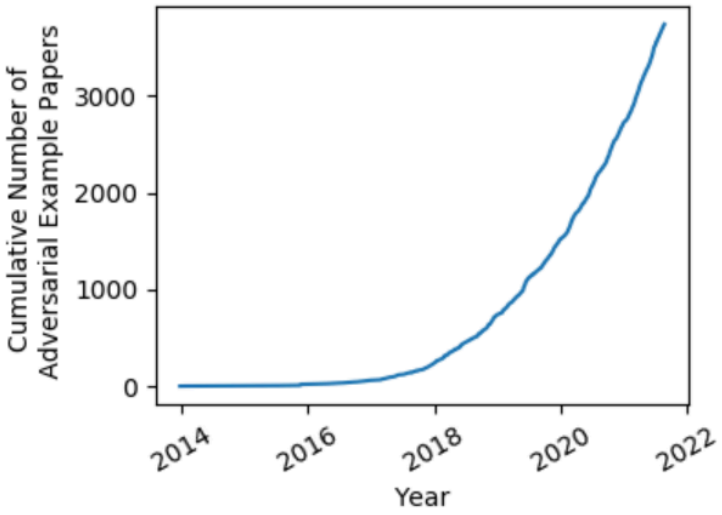


***Figure 3**: Scholarly articles on adversarial examples, cumulative since 2014,*

.

Two of the best known methods for generating these examples are the Carlini and Wagner (CW) method [38] and the *Fast Gradient Sign Method* (FGSM) [15]. The adversarial examples of these papers are the result of modifying an original element through perturbations. In addition to the aforementioned methodology, there are also *Unrestricted Adversarial Examples*, which are generated from scratch with generative algorithms, such as Generative Adversarial Networks, also known as GANs [1]. In this second category there is no original image to start from.

It is of interest to clarify that, throughout the paper, the definition given in section **2.5** will be the reference when the term adversarial example appears. This definition focuses on the effect it has on a face recognition system and not on its generation, as it is not the aim of this paper to study the taxonomy of adversarial examples in depth.

In the literature it can be observed that researchers study adversarial examples from both the attacker's and the defender's point of view. The attack methodology can be divided into two modes: *white box* (one has access to the network to be attacked and its weights) [29] and *black box* (one does not know how the network to be attacked works) [9]. An interesting

feature of adversarial examples that can be observed in [28] is the verification of the so-called **transferability principle**: those examples that manage to cause failures in the classification of a network, replicate this behaviour in other networks with different architectures or, even, with a different training set, causing the adversarial examples to be robust to different classifiers.

As mentioned in section **1.**, it is of interest to analyse the role of *autoencoders* and principal component analysis in the study of adversarial examples. In relation to PCA it could be observed that there are studies for both attack [24] and defence [25]. On the other hand, the use of *autoencoders* appears mostly in the defence of opposing examples, which is why this trend was analysed in greater depth.

On the one hand, one can look at the case of PuVAE [31] in which a variational *autoencoder* (VAE) was used to "purify" an adversarial example. The approach involves receiving an image and, before sending it to a classification network, processing it with the VAE in a way that renders the attack harmless by minimising perturbations. The authors point out that it turns out to be faster than methodologies that do the same with GAN.

In addition to purification of adversarial examples, *autoencoders* are also used in the detection of adversarial examples. In [32] they study the difference in the distribution of unmodified images compared to that of adversarial images. For this, they train one *autoencoder* for each hidden layer in the classifier to be defended: in the latent space of each *autoencoder*, the *manifold* (distribution) of the original images corresponding to a layer can be found. The distance of latent representations of adversarial examples to the distribution of original images can be analysed. This metric is used as a detection method. The proposed mechanism was successfully tested experimentally on 5 different attacks, one of which is the previously mentioned CW.

Although no papers were found that penetrate these defences, this does not mean that they are valid for all types of adversarial examples. One aspect discussed in [27] on the study of defences is the lack of a rigorous framework to analyse whether they are effective. There are works that adapt new attacks to exploit proposed defences (in [33] thirteen defences are violated and in [26] another ten). Although [26] was published in November 2017, even today, concerns remain about the lack of proper checks at the time of publication.

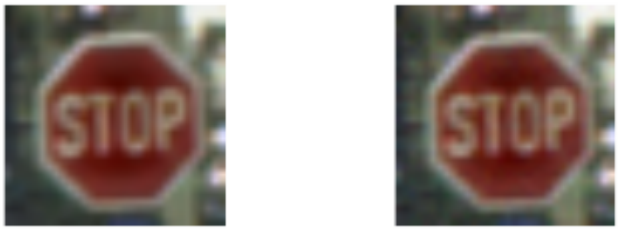


***Figure 4**: On the left is the original image. On the right, the imperceptibly modified image, which the classifier recognises as a "yield" signal. Examples obtained from [9].*

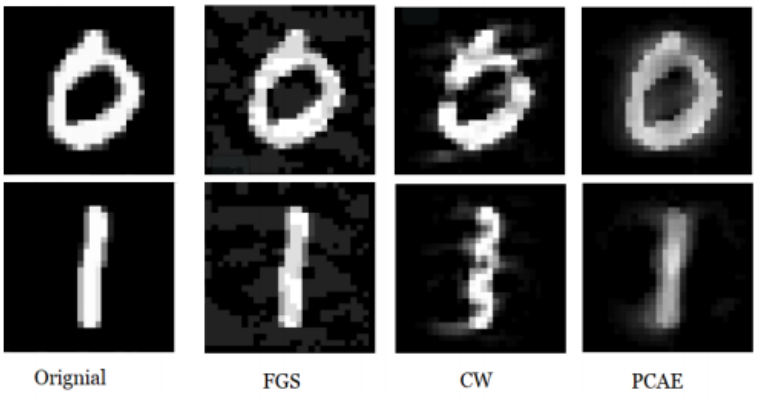


***Figure 5**: Generation of adversarial examples from different published methods. Examples obtained from [24].*

With regard to the study of the construction of adversarial examples, the lack of quality analysis of the images generated was striking. While there are adversarial examples with imperceptible modifications, as can be seen in ***Fig. 4***, there are also attacks that appear to lower image quality, such as FGSM, CW and PCAE. In ***Fig. 5*** examples of these attacks can be found for digits 0 and 1 of the MNIST dataset. In this figure it is possible to notice that the added perturbations are not equivalent, as some images have worse quality than others.

4. Proposal

4.1. Context of the Project

When deciding on the topic of the final project, the two main areas of interest of the authors were taken into account: artificial intelligence and computer security. It was decided to continue exploring the path of artificial intelligence and, within it, the problem of adversarial examples in deep learning algorithms was found. This area captured the authors' attention, as it demonstrated the existence of vulnerabilities that could have serious consequences when using these algorithms in everyday operations.

An important choice was to determine whether the research would be positioned from the point of view of attack or defence. The initial premise was that the attacking position was simpler when approaching the subject as it would allow us to understand how to generate an adversarial example and, this being the authors' first research project, it was decided to continue in this direction.

4.2. Metodology of Attack

A new methodology for generating adversarial examples is proposed by combining the use of *autoencoders* with principal component analysis. As can be seen in ***Fig. 6***, from the representation of an image in the latent space, its projection in principal components is obtained. Once the obtained projection has been manipulated, the inverse operation is applied, resulting in a vector belonging to the latent space. Finally, this vector is sent to the decoder portion of the autoencoder to obtain an adversarial example.

Note that this methodology seeks to obtain an image whose class can be recognised by an oracle O (explained in section **2.1.**). On the other hand, in ***Fig. 6*** the response of the facial recognition system S is not real but is intended to illustrate an example of an evasion attack:

S decreases the individual's recognition rate when receiving the adversarial example. The individual's true label, "Meryl Streep", is the one assigned by the oracle O, consistent with human classification.

This methodology will be used to breach facial recognition systems, in a black box attack scheme. In the section 2.6. It was mentioned that a common use of PCA was dimensionality reduction but, in this case, the possibility of ordering the features according to the variability they represent is the key element of interest. In this case the dimensionality is not reduced, since a compact version of the original data was already obtained by sending it to the encoder.

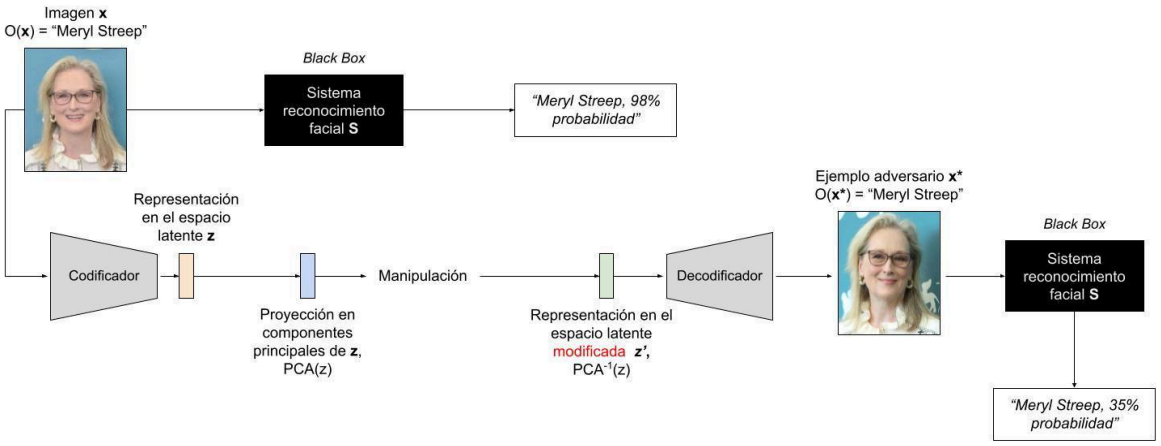


***Figure 6**: Graphical representation of the proposal for generating adversarial examples.*

Finally, the systematic lack of studying the quality of adversarial examples is a differential proposed in this work. Facial recognition services, such as Amazon Rekognition, provide functionalities to analyse features such as brightness, contrast, etc., providing information about the quality of an image. It is considered important to study this aspect since negative effects on quality diminish recognition.

4.3. Objectives

The work consists of an experimental section, where an attempt will be made:

- Carrying out evasion attacks, as defined in **2.3.**
- Carrying out impersonation attacks, as defined in **2.4.**

It is expected to find a relationship between people's identity and projections in major components of the latent representation of people's images. It is assumed that it will be possible to describe the identity of the individuals in the dataset with the first components and that they will represent a significant amount of information with respect to the others.

On the other hand, this work will seek to ensure that the quality levels of the images generated are in a stipulated range from unmanipulated images. As mentioned in **3.**,

Although there is talk of "imperceptible modifications", visible differences can be noticed with respect to the original image, as seen in ***Fig. 5***.

4.4. Framework of the problem

4.4.1. General Purpose Hardware

It was decided the use of general-purpose hardware instead of specialized hardware because the project was initiated remotely and the authors only had access to this prior kind. Once the project was advanced, the possibility arose of using an NVIDIA GPU that was in the institute but, because the project was in an advanced state with such a limitation in mind, it was decided not to take that path.

4.4.2. Data Set

For the experiment, it was arranged to use only two subjects for the dataset. This not only allows for a smaller set (reducing the computational cost to train the autoencoder), but also simplifies the subsequent analysis, since the goal will be to move from one person to another. If more individuals were added, the number of relationships would increase.

Over the course of the work, the dataset evolved in three ways:

- **Number of images**: it went from 40 to 100 images per individual
- **Pre-processing**: It was decided to transform the images from color to black and white. In this case, the decision was for training optimization at the cost of losing information about the individual, such as eye color.
- **Original Photos**: in the first iteration there were two individuals: Ignacio and Marina. As can be seen in ***Fig. 7***, the initial images of Ignacio featured a beard and glasses. It was decided to remove these features to make the process of generating adversarial examples easier, as both the beard and the glasses were very distinctive features.

Figure 7: *On the left, initial images.*
On the right, the images that are used in the work.

4.4.3. Amazon Rekognition as a target system

The last limitation imposed concerns the facial recognition system to be used. Only the Amazon Rekognition system was selected. In [9] and [10], adversarial examples with different classifiers are analysed to study the principle of transferability.

Since it was the first approach to research on the subject, there was a significant cost in the implementation of infrastructure that would allow experimentation. Focusing on implementing

a system solely for Amazon Web Services (AWS) saved time spent planning and designing an abstraction that could be used for any facial recognition system. The disadvantage of this decision is the loss of scalability and the possibility of verifying the principle of transferability, but, in this case, it is considered beneficial because it allows more time to be spent on the choice of the autoencoder and subsequent experimentation.

5. System Diagram

Throughout the development of the work, the architecture of the system was iterated to improve and adapt it to changing requirements. In order to show the system, it was decided to separate the explanation into three diagrams: autoencoder training, experimentation process and, finally, visualization of results. For each of them, an explanation of its components and the modules necessary for implementation will be made.

5.1. Autoencoder Training

This section is mainly composed of two processes: first, the preprocessing of the dataset, characteristics of which were discussed in section **4.** Secondly, it is represented in ***Fig. 8*** the training of the autoencoder with the result of the first process.

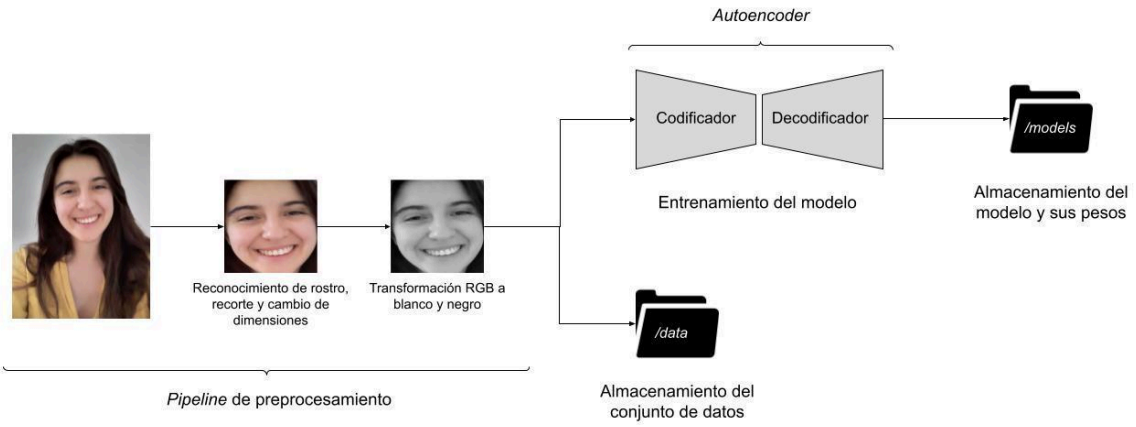


Figure 8 *Flowchart on the autoencoder training process..*

Several modules had to be developed for this section. First, there is a module for pre-processing called `face_detection`, the functionality of which is described in detail during the section **5.1.1.** which is found below.

On the other hand, there is the implementation corresponding to the autoencoder that includes three modules: autoencoder [1], `encoder`[2] y `decoder`[3]. They are responsible for the relevant functionality for training and subsequent data prediction. The modularization allowed for greater flexibility when working with image representations in the latent layer, which was particularly useful for the next section., **5.2.**

[1] Implementation: `modules.autoencoder`
[2] Implementation: `modules.encoder`
[3] Implementation: `modules.decoder`

Finally, auxiliary modules were implemented, such as: `data_loader`[4] and `model_loader`[5], for storage and retrieval of the dataset and models. After training the autoencoder, the model was saved to a .json file, along with its weights to a .h5 file, for later reuse. This is necessary to avoid training the autoencoder every time you want to use it, as this process can be time consuming.

5.1.1. Data Pre-processing

The original data are color images, taken with each individual's cell phone. Before using the images in the application (both for autoencoder training and experimentation), the following transformations had to be applied:

- **Face detection and clipping**: the images contain information that is of little interest to the research framework, such as the shirt worn by the person or some object or texture of the background. The MTCNN [17] face detection algorithm was used to crop the image and thus leave only the face.
- **Scales image to 256x256**: this is an important step for uniform image size and, in turn, a manageable size.
- **Black and White**: transforming the color images to black and white brings as an advantage a great simplification of the image. However, this simplification comes at the expense of the loss of some characteristic details of the individual such as eye colour. It was decided that the gain in performance was greater than the loss in detail, so this modification was maintained.

These transformations were made to simplify the problem and reduce the load that is generated on the autoencoder. It is important to highlight this because of the hardware limitation that was raised in the section 4.4.1. It can be observed that the images in ***Fig. 9*** have gone through the preprocessing pipeline resulting in the corresponding images in ***Fig. 10***.

Figure 9*: raw images of the dataset.*

[4] Implementation: `modules.data_loader`
[5] Implementation: `modules.model_loader`

***Figure 10**: images of the dataset after preprocessing.*

5.2. Process of Experimentation

The process of experimentation shown in ***Fig. 11*** has two important sections: the first, enclosed in a dotted line, will be executed only once and then those results will be used for the experiment section, the second, outside the dotted line.

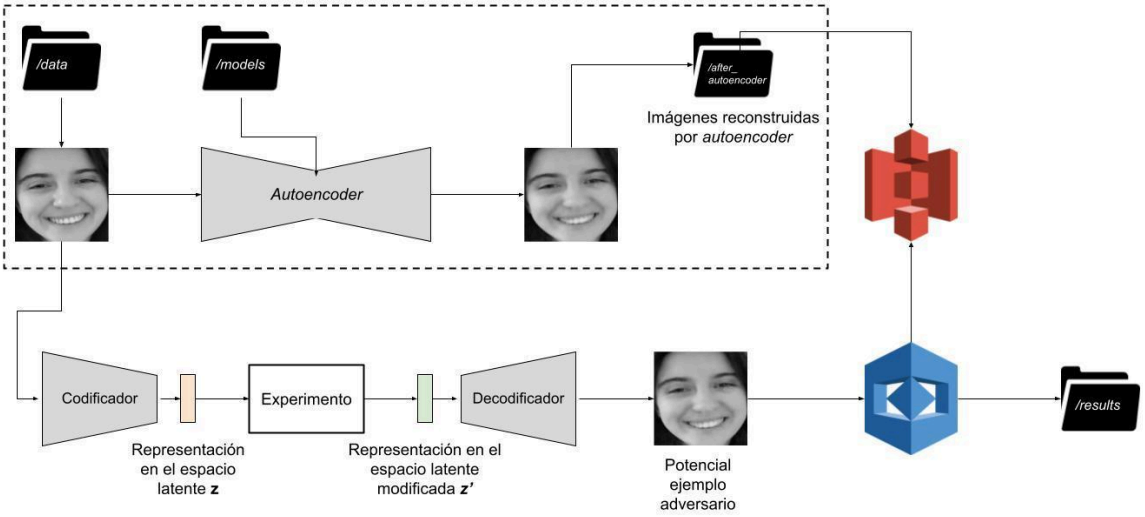


***Figure 11**: Flow diagram of the experimentation process.*

A problem that arose was to determine against which data set the altered images (potential adversarial examples) would be compared with. Since the autoencoder performs a reconstruction of an image from a compressed representation and, furthermore, that it is not a state of the art model, the images that could be obtained from this neural network do not have the same quality as the faces in the original dataset. This could result in a loss of similarity that corresponds to the difference in quality and not to the manipulation performed. Consequently, the images that are used to compare against potential adversarial examples are those resulting from the output of the autoencoder.

Inside the dotted line, it is possible to find the preprocessed images stored in `/data` and the autoencoder selected for experimentation, stored in the folder `/models`. The details of the selection can be found in the section **6.** The goal of the process within the flowchart is to get all the images reconstructed by the autoencoder, as mentioned previously. The result of this procedure was stored in another folder, called `/after_autoencoder`.

These images are saved in the Simple Storage Service (S3) of AWS, because Amazon Rekognition will make use of those images to obtain the characteristics of these faces, from which it will make comparisons.

On the other hand, an experiment will be carried out using the encoder and decoder parts of the autoencoder separately. To manipulate an image, use the module `data_loader` to load it into the program and send it to the encoder to get its representation in latent space. Then, the manipulation corresponding to the experiment in question shall be carried out. (details of the experiments can be found in section **8.**). Once the vector corresponding to the modified latent space is obtained, it is sent to the decoder to acquire the corresponding reconstruction.

Finally, it is of interest to analyze how the manipulated image alters the similarity results against unaltered images by sending this modified input to the Amazon Rekognition service to retrieve the relevant results and store them in the folder `/results`.

5.2.1. Conection to Amazon Web Services

To use services necessary for this work, it was decided to take advantage of the *Free Tier* of Amazon Web Services, which allows you to make use of certain technologies (with some limitations) without cost, for a year.

Top Free Tier Services by Usage — View all

Service	Free Tier usage limit	Month-to-date usage
Amazon Rekognition	Analyze 5,000 images per month for Amazon Rekognition	66.78% (3,339.00/5,000 Images Processed)
Amazon Simple Storage Service	2,000 Put, Copy, Post or List Requests of Amazon S3	35.05% (701.00/2,000 Requests)
Amazon Simple Storage Service	20,000 Get Requests of Amazon S3	13.04% (2,607.00/20,000 Requests)
Amazon Simple Storage Service	5 GB of Amazon S3 standard storage	0.04% (0.00/5 GB-Mo)
AWS Lambda	1,000,000 free requests per month for AWS Lambda	0.01% (104.00/1,000,000 Requests)

***Figure 12**: Services that are part of the Free Tier of Amazon Web Services.*

The integration of the necessary services with the implementation of the authors was carried out using `boto3` [18], a *Software Development Kit* (SDK) which allows you to connect to AWS using Python.

5.2.2. Amazon Rekognition

The most important service used is Amazon Rekognition (which will be abbreviated as Rekognition throughout the work), Amazon's facial recogition system. With `boto3` the functionalities necessary for the experimentation that was to be carried out were implemented:

1. Analyse the quality of an image.
2. Compare the similarity between two images.

3. Compare the similarity of an image against a set of images.

All functionality relevant to this service was implemented within the `rekognition`[6] module. It is important to mention that functionalities 2 and 3 are not executed in the same way. To compare an image against a set (functionality 3) a collection is created within the system. Collections persist the features of the faces present in the images of the dataset. This is why, with `boto3`, the functionality to create/delete a collection and add/remove faces from a collection was added. Persistence is carried out in Amazon Simple Storage Service (S3), which will be presented in the next section.

For the quality analysis functionality of an image (1), the `DetectFaces` function [19] is used: it receives an image and returns a large amount of information about it, such as age range, emotions, gender, etc. Within this data dictionary, the Quality section is of interest.

On the other hand, to compare the similarity between two images (2), the function `CompareFaces` [19] is used, which receives a source image and a target image. In addition to returning the similarity between both images, another data of interest is the confidence with which it recognises that there is a face in the source image. Both are returned as percentages.

The confidence is a parameter that can serve as a quality filter, since it seeks to indicate with what confidence a face was found in the image to be compared. If this confidence is low, it means that Rekognition did not even find a person in the data in question.

In reference to the comparison of an image with a collection of images (3), `SearchFacesByImage function` [19] is used. The objective is to analyse the similarity of the altered image with respect to the complete dataset. A fragment of the response sent by Rekognition can be seen in **Fig. 13**. As a result, it is possible to obtain the average of the similarity of the image in question with respect to each of the individuals present in the image set.

Finally, as mentioned in sections **2.3.** and **2.4.** above, it was necessary to define a threshold to determine when an evasion and/or impersonation attack occurred. Since Rekognition's default value for returning similarity results is 80%, it was decided to use this as the threshold. While AWS recommends modifying this parameter to 95%, or higher for sensitive uses, it is believed that it is important to analyse whether it is feasible to breach the tool when using it with default parameters. If there is such a possibility, it would be a point of discussion whether the company should use more secure defaults because, although the responsibility for deciding a threshold lies with the customers, it is an extra barrier of protection in case a user overlooks it.

[6] Implementation: `infra_aws.rekognition.Rekognition`

```
→ comparisons git:(master) ✗ cat pf-photo-dump-nachito9_8-comparison.info
{
    "SearchedFaceBoundingBox": {
        "Width": 0.7555350661277771,
        "Top": 0.07716283202171326,
        "Left": 0.04681132733821869,
        "Height": 1.033298134803772
    },
    "SearchedFaceConfidence": 99.99996185302734,
    "FaceMatches": [
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.772454023361206,
                    "Top": -0.10266999900341034,
                    "Left": 0.04762300103902817,
                    "Height": 1.0444999933242798
                },
                "FaceId": "a56e8cfe-5fe7-4d68-9c84-6b334e623362",
                "ExternalImageId": "marina2.jpg",
                "Confidence": 100.0,
                "ImageId": "4222dcc5-528b-353b-a53b-e781d073f5fd"
            },
            "Similarity": 99.997314453125
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.7548850178718567,
                    "Top": 0.0700140967965126,
                    "Left": 0.05071999877691269,
                    "Height": 1.038159966468811
                },
                "FaceId": "35ba94d5-a2f2-4c57-b74e-a30f9b629645",
                "ExternalImageId": "marina2.jpg",
```

Figure 13: *Response fragment resulting from comparing an image against the face collection..*

5.2.3. Amazon S3

Amazon Simple Storage Service is a service that allows storing objects in the cloud within repositories called buckets. As briefly mentioned in section **5.2.2**, it was used to store the images against which the comparison of an image provided by the user is to be performed.

Within the functionalities that were implemented in the system for this work, it is possible to:

- Create new *buckets*
- Delete an existing *bucket*
- Upload and delete files of your choice

5.3. Visualisation of results

For the analysis of the results of the previous section, the Python library `matplotlib` [20] (version 3.3.2) was used, which allowed conclusions to be drawn and new experiments to be devised from them.

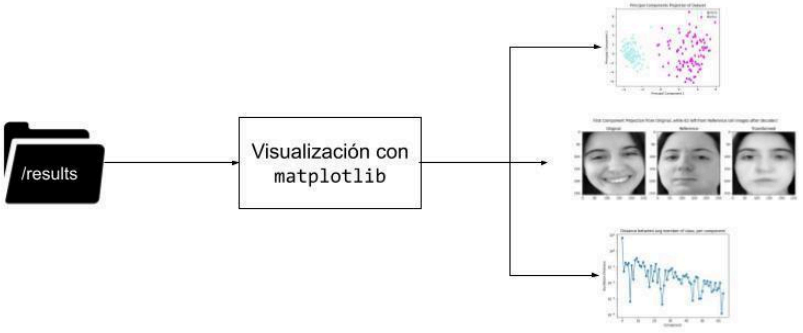


***Figure 14**: Flowchart on visualisation of results. The images on the right are actual results obtained from the experiments.*

6. Autoencoder Architecture

The dataset to be used is the one resulting from section **5.1.1**: a hundred (100) images per subject (Marina and Ignacio), in black and white, of dimensions 256x256. The separation into training and test sets will be 80-20 respectively.

To generate the necessary *autoencoder*, the architecture designed by Adrián Rosenbrock in an article on his website, *PyImageSearch* [2], was used as the base. This architecture used the MNIST dataset, a set of handwritten digits from 0 to 9, in black and white, with dimensions 28x28. It was from this architecture that an attempt was made to find the best combination of parameters for Marina and Ignacio's image set. We will call Rosenbrock's autoencoder the "base autoencoder".

The decision process for the optimal configuration will be defined in the following, in which four sections are distinguished. First, the base autoencoder will be described. Then, the choice of architecture parameters to be modified and the values they take in the experimentation. In the third section, a filter of architectures was made to analyse in more detail. Finally, a selection methodology based on metrics provided by Rekognition, the face recognition system, was designed.

It was decided to use a previous *autoencoder*, rather than generate one from scratch, because when testing the Rosenbrock autoencoder with the data set of this work, the reconstruction results (observed subjectively, by the authors) were very good. It was assumed that this architecture had the potential to generate satisfactory results, as long as a more structured analysis of its capabilities is performed.

On the other hand, the time factor had to be taken into account: the hardware available is that provided by a general-purpose computer. This limitation translated into expectations regarding the quality of the reconstructed images. To train the different architectures, both the dataset and the autoencoder itself were factors that strongly influenced the time it would take to train. Given that the space of possible autoencoders that can be studied is inordinate, it is consistent to take an "anchor" point (the base autoencoder), from which an improvement in reconstruction can be sought.

6.1. Base Autoencoder

The first point to highlight is that it is a convolutional autoencoder, also called CAE, built with the Python library, Keras (version 2.4.3). When deciding what type of neural network to use, it was decided to focus the search resources on one with convolutional layers, given the success demonstrated when working with images.

As mentioned in section **2.7**, the autoencoder can be thought of as the composition of two components: the encoder and the decoder. From this neural network, some changes were applied to build what was called the "corrected base autoencoder", which will be abbreviated as ABC.

The first thing that had to be adapted was the dimensions of the input, as they are not the same. On the other hand, the dimension of the latent layer was increased from 16 to 64 (as a consequence of the change in dimension of the images in the dataset) and a third convolutional layer was added.

In the ABC, the Minimum Squared Error (MSE) cost function, the optimiser parameters, the parameters of the activation functions, the type of padding and the window shift when applying the filters of the convolutional layer, called strides, were maintained. The following is a summary of the ABC encoder and decoder architecture (a summary of the base autoencoder architecture can be found in the Appendix, under Base autoencoder architecture), obtained from the summary function provided by Keras.

```
Model: "encoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 256, 256, 1)]     0
_________________________________________________________________
conv2d (Conv2D)              (None, 128, 128, 16)      160
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 128, 128, 16)      0
_________________________________________________________________
batch_normalization (BatchNo (None, 128, 128, 16)      64
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 32)        12832
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 64, 64, 32)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 64, 64, 32)        128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 64)        100416
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 32, 32, 64)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 32, 32, 64)        256
_________________________________________________________________
flatten (Flatten)            (None, 65536)             0
_________________________________________________________________
dense (Dense)                (None, 64)                4194368
_________________________________________________________________
```

```
leaky_re_lu_3 (LeakyReLU)    (None, 64)                0
=================================================================
Total params: 4,308,224
Trainable params: 4,308,000
Non-trainable params: 224
```

```
Model: "decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 64)]              0
_________________________________________________________________
dense_1 (Dense)              (None, 65536)             4259840
_________________________________________________________________
reshape (Reshape)            (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 64, 64, 64)        200768
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 64, 64, 64)        0
_________________________________________________________________
batch_normalization_3 (Batch (None, 64, 64, 64)        256
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 128, 128, 32)      51232
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 128, 128, 32)      0
_________________________________________________________________
batch_normalization_4 (Batch (None, 128, 128, 32)      128
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 256, 256, 16)      4624
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 256, 256, 16)      0
_________________________________________________________________
batch_normalization_5 (Batch (None, 256, 256, 16)      64
_________________________________________________________________
conv2d_transpose_3 (Conv2DTr (None, 256, 256, 1)       145
_________________________________________________________________
activation (Activation)      (None, 256, 256, 1)       0
=================================================================
Total params: 4,517,057
Trainable params: 4,516,833
Non-trainable params: 224
```

For both the encoder and decoder, it can be seen at the end of his description that trainable and untrainable parameters are defined. In Keras' summary of the neural network architecture, untrainable parameters are those that will not be optimised during the autoencoder iterations, but must be defined a priori. Examples of untrainable parameters could be the number of hidden layers, number of nodes per layer, etc. On the other hand, trainable parameters (as the name suggests) are those that are modified as the training progresses, such as neuron weights.

6.2. Parameter Analysis

The first challenge when studying the base autoencoder was to decide which parameters would be subject to modification and analysis, as well as to choose which values they would take.

For the performance analysis of each of the parameters, it was decided to study, per training epoch, the reconstruction of the images as long as overtraining was minimised. This means that not only was the reconstruction of the images of the test set observed, but also that this value was not found to be at a significant distance from the reconstruction error in the training set (in reference to the other differences corresponding to the values taken by the parameter in question).

For the reconstruction, the average of the difference between the autoencoder output, $\hat{x}$, and the original image of the training set, x, was taken, according to equation (3). This equation represents the error value of the autoencoder, which will be called the *Loss*. The parameter n corresponds to the number of elements in the test set.

$$Loss = \frac{1}{n}\sum_{i=1}^{n} \|\hat{x} - x\| \quad (3)$$

On the other hand, to examine overtraining, the difference between the average loss value for the test set, $Loss^{test}$, and the average loss value for the training set, $Loss^{train}$, were taken. This difference is called *Loss Gap*, defined according to equation (4).

$$Loss\ Gap = Loss^{test} - Loss^{train} \quad (4)$$

Below are the sections corresponding to the parameters that have been analysed. A table listing the initial parameters of the autoencoder can be found in the Appendix section, ABC Initial Parameters.

Once the corresponding data has been obtained to study the parameters, values will be selected according to the following criteria:

1. *The Loss* and *Gap Loss* graph (corresponding to the difference between training set and test set) are analysed together to validate reconstruction and overtraining.
2. The reconstructions of the images are inspected (see **Fig. 15** for an example of the reconstructions) to determine if this value is maintained in the selection. The complete reconstructions, per modified parameter, can be found in the Appendix.

***Figure 15**: visual reconstructions of the autoencoder according to values of the learning rate of the Adam optimiser. For each parameter, these reconstructions were available.*

6.2.1. Optimiser learning rate

When training a neural network, it is necessary to bear in mind the importance of the choice of the optimiser and its parameters. In ABC, the Adam algorithm is used as the optimiser. It was decided to keep this algorithm because it is computationally efficient, of great relevance when taking into account that general purpose *hardware* is used, and because it shows better performance than other optimisers, such as AdaGrad, RMSProp, Stochastic Gradient Descent (SGD) and AdaDelta, in the evaluation performed with a convolutional neural network, both with the MNIST data set and with CIFRAR10 [3].

While Adam is a popular optimiser, there is debate about its superiority in generalising (avoiding overfitting) over a dataset as opposed to SGD [4]. Despite this, it was determined to limit the evaluation to the use of Adam with variations in its learning rate due to the computational efficiency mentioned above. In case of not obtaining satisfactory results, the alternative would be to change the optimiser.

The values taken into account vary between 10^{-1} and 10^{-5} with a step of 10^{-1}.

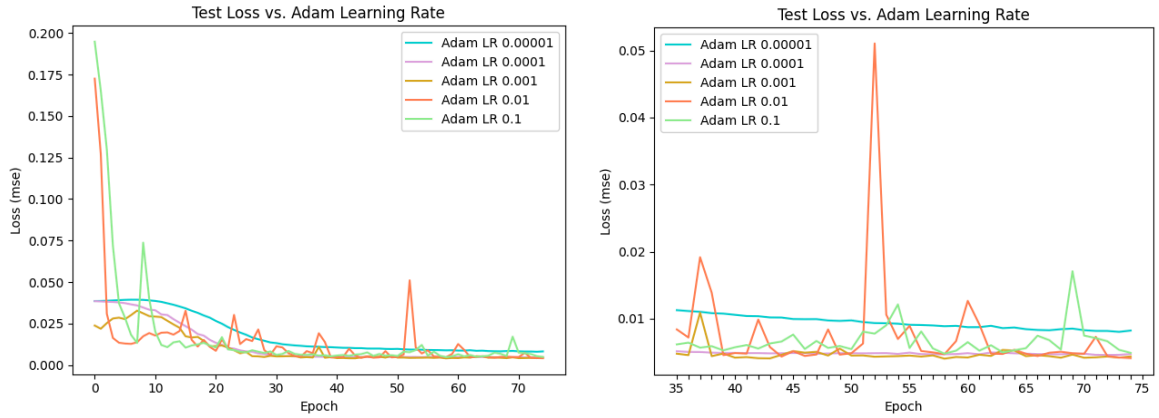


***Figure 16**: Loss for the test set for 75 epochs, according to the optimiser's learning rate. On the right, the values are from epoch 35 onwards for better interpretation of the results.*

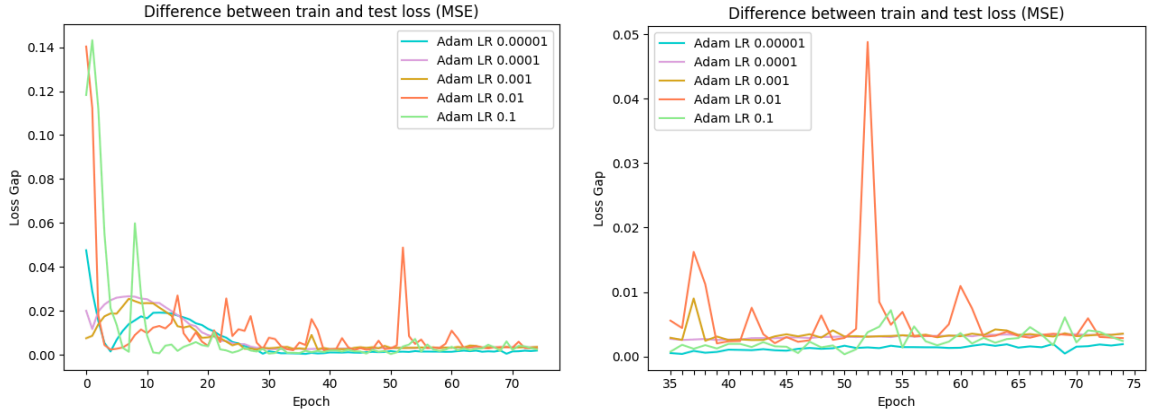


***Figure 17**: Gap Loss over the testing of an autoencoder for 75 epochs, according to the optimiser's learning rate. On the right, the values are from epoch 35 onwards for better interpretation of the results.*

According to the graph in **Fig. 16**, it is possible to notice that the learning rate of 10^{-3} and 10^{-4} are among the lowest values in terms of the error function. On the other hand, in **Fig. 17**, the Gap Loss value also remains low with respect to other learning rates. In addition, the reconstruction of the images was satisfactory according to the authors.

Finally, the 10^{-5} learning rate was selected as its *Gap Loss* is the lowest of all. When analysing the image reconstruction, it could be noticed that the images were blurred, which could be an indication that this architecture lacked training epochs. The graph on the right in ***Fig. 16*** supports this reflection, as it can be seen that the trend of the loss value for 10^{-5} is decreasing.

6.2.2. Number of convolutional layers

With respect to the architecture of the neural network, one of the things to define was the number of convolutional layers suitable for the network. The base autoencoder started with 3 convolutional layers, with their respective activation function and subsequent *BatchNormalization* layer, but there was some doubt as to whether this would be sufficient for the dataset used in the research. Another element that varied as the number of layers increased (growing the number of filters in each layer), was the kernel size for the layer in question.

The purpose of modifying this parameter is to analyse whether the network architecture is "sufficient" to learn the dataset. On the other hand, care needs to be taken not to fall into over-layering, which can be counterproductive to the generalization goal.

The following configurations were analysed in this section:

- 3 convolutional layers of 16 (3x3 core), 32 (5x5 core) and 64 (7x7 core) filters respectively.
- 4 convolutional layers of 16 (core 3x3), 32 (core 5x5), 64 (core 7x7) and 128 (core 7x7) filters respectively.
- 5 convolutional layers of 16 (3x3 core), 32 (5x5 core), 64 (7x7 core), 128 (7x7 core), and 256 (7x7 core) filters respectively.

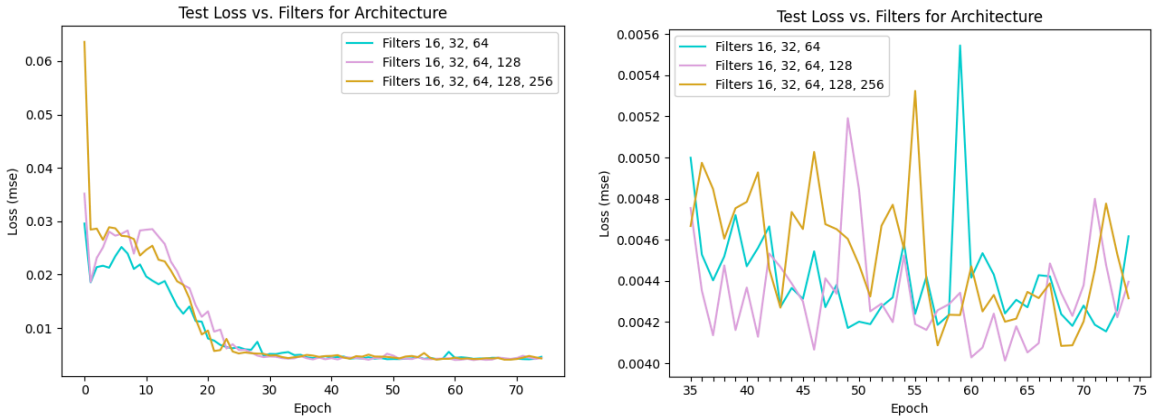


***Figure 18**: Loss for the test set for 75 epochs, according to the number of convolutional layers. On the right, values are from epoch 35 onwards for better interpretation of the results.*
interpretation of the results.

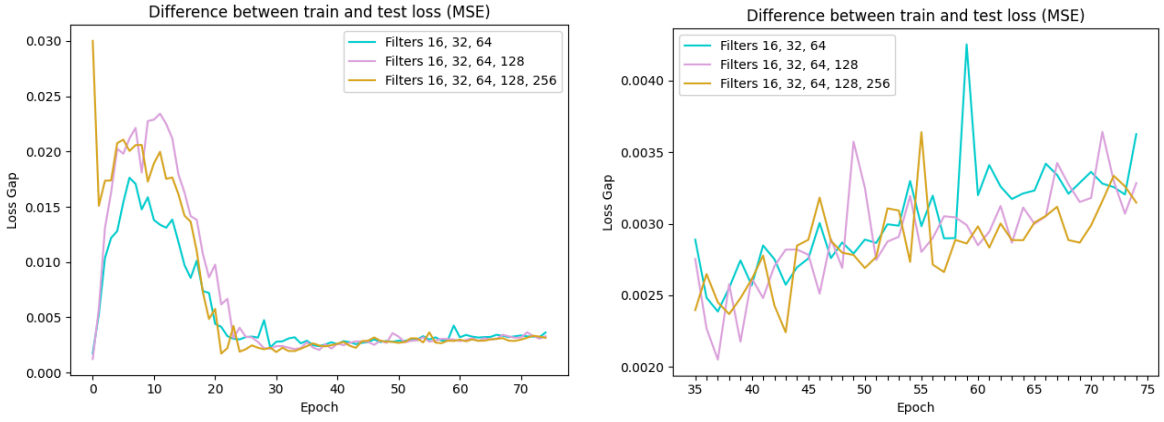


***Figure 19**: Gap Loss over the test set of an autoencoder during 75 epochs, according to the number of convolutional layers. On the right, the values are from epoch 35 onwards for better interpretation of the results..*

In this case, it is possible to observe that the values studied give similar results. From the data shown in the graphs, there is no reason to believe that one number of layers (with their respective specifications) is better than the other. Analysing the reconstructions, it was noted that those architectures with 3 and 5 layers show the best results (and it even appears that the 5-layer one is the most satisfactory).

Although 5 layers could have been selected for the later section (or both values so as not to leave out possibilities), it was decided to keep only 3 convolutional layers. This was because there is no significant superiority in terms of *Loss, Gap Loss* and reconstructed images (although there is a small improvement, the images are not the same for both architectures, so this is not conclusive). The advantage of selecting the 3-layer value is that the network takes less time to train.

6.2.3. Amount of data for gradient update

Another parameter that it was decided to analyse was the *batch* size during training. This value defines the number of elements in the training set that are used to perform a gradient update. This update is intended to be an estimate of the "true" gradient, as it should be calculated with all the data in the set.

The values that were taken for this parameter are 4, 8, 12 and 32.

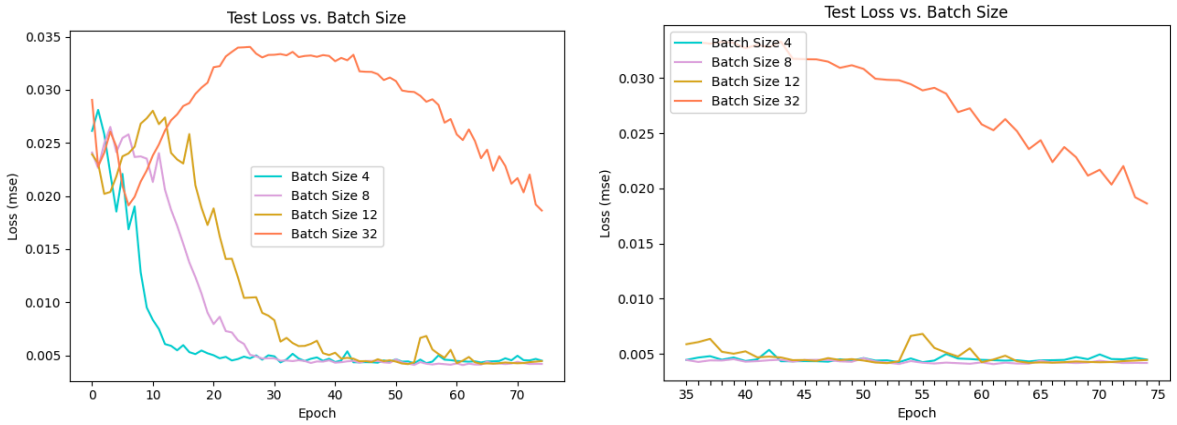


***Figura 20**: loss for the test set for 75 epochs, according to the amount of data to be updated. On the right, the values are from epoch 35 onwards for better interpretation of the results.*

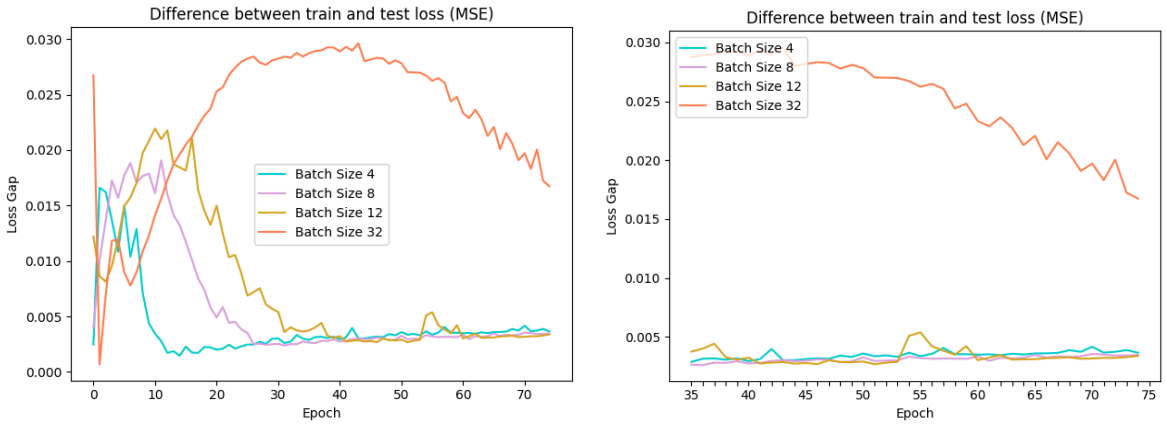


***Figura 21**: Gap Loss over the test set of an autoencoder for 75 epochs, amount of data for update. On the right, the values are from epoch 35 onwards for better interpretation of the results..*

The selected values were 8 and 32. As it is possible to see in **Fig. 20** and **Fig. 21**, the best results are for sizes 4 and 8, which corresponds to the analysis of the image reconstruction. Although the value 4 could have been chosen for the later section, the value 8 seems to be slightly below 4 in *Loss* and, moreover, it was already the default value in the initial

parameters. Adding the value 4 would have resulted in more autoencoders to study in the following sections.

On the other hand, the value 32 was selected for similar reasons as the 10^{-5}, learning rate, in section **6.2.1.** Although the performance in ***Fig. 20*** and ***Fig. 21*** is low with respect to the other values, a decreasing trend is clearly noticeable. Additionally, the reconstructed images are somewhat blurred, which may indicate a lack of training. It was decided to keep the value in order to analyse whether the most exaggerated expressions (which are usually the worst reconstructed) could be learned with these models, when combined with more training epochs.

6.2.4. Latent layer dimension

The choice of latent layer dimension was a parameter of special interest for this work. The representations of the images in this space would later be used for the principal component analysis.

For data collection, 3, 9, 16, 32, 64 and 128 neurons were taken.

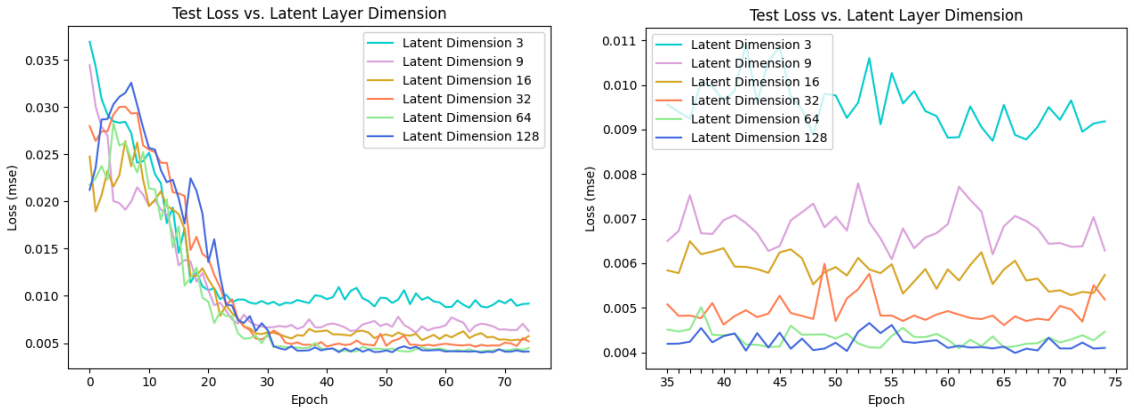


***Figure 22**: Loss for the test set over 75 epochs, according to the number of neurons in the latent layer. On the right, values are from epoch 35 onwards for better interpretation of the results.*

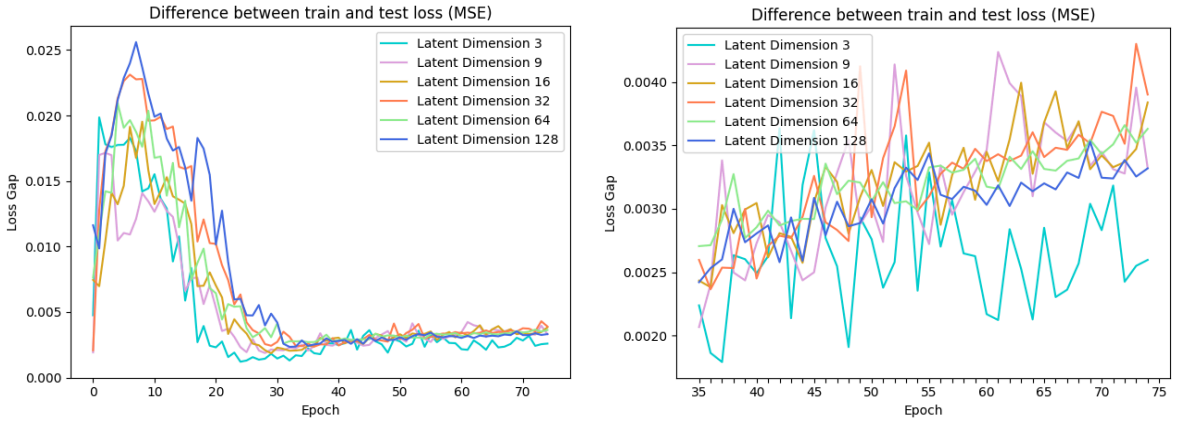


***Figure 23**: Gap Loss along the test set of an autoencoder during 75 epochs, according to the number of neurons in the latent layer. On the right, the values are from epoch 35 onwards for better interpretation of the results..*

The values selected were 64 and 128. As can be seen in **Fig. 22**, they reach the lowest values compared to using 3, 9, 16 or 32 dimensions in the latent layer. Not a minor detail is that all the values seem to have reached some kind of convergence in this graph. This resulted in discarding the latent layer of dimension 3 which, in **Fig. 23**, seemed to have the best performance in terms of *Gap Loss* (under overtraining). As the graph of **Fig. 22** did not provide any indication that this 3D latent layer architecture would improve, it was decided to discard it.

One of the advantages of having a latent layer with enough dimensions is the possibility of achieving a better reconstruction. On the other hand, when manipulating the vector corresponding to the representation of an image in latent space, we have more “places” to do so. In turn, there is a potential risk that the network stores the data instead of generalizing since the set of images is not large.

6.2.5. Activation function

As can be seen from both the encoder and decoder architecture summary, the Leaky ReLU activation function is applied to the output of each convolutional layer. One of the main advantages of this activation function, which it shares with its predecessor ReLU, is the computational cost compared to other nonlinear activation functions, such as the hyperbolic tangent or the sigmoid function. Bearing in mind that computational distances are important due to the constraints already mentioned in section **4. 4**, it was decided to keep this activation function and to change only the parameter α which represents the slope of the activation function for the case of values below zero.

ReLU could have been used instead of Leaky ReLU, but it was decided not to take this path for simplicity because in a first iteration, testing with the base autoencoder, satisfactory results had been obtained. According to [5], the ReLU activation function can cause several neurons to “die” (their output always ends up being zero), so Leaky ReLU would be the alternative to minimize this problem. This is a point in favor of the activation function chosen here, but according to this same article, this problem does not significantly affect training, at least using the MNIST dataset, so ReLU remains a viable option. Even in some discussion forums, community members have not noticed significant differences in the use of one or the other.

The values with which the slope of the activation function was varied are 0. 01, 0. 05, 0. 1, 0. 3, 0. 6, 0. 9 and 1. 3. To avoid increasing the difficulty of analysis, the same value of α was used for all instances where the activation function is used, both for the encoder and the decoder

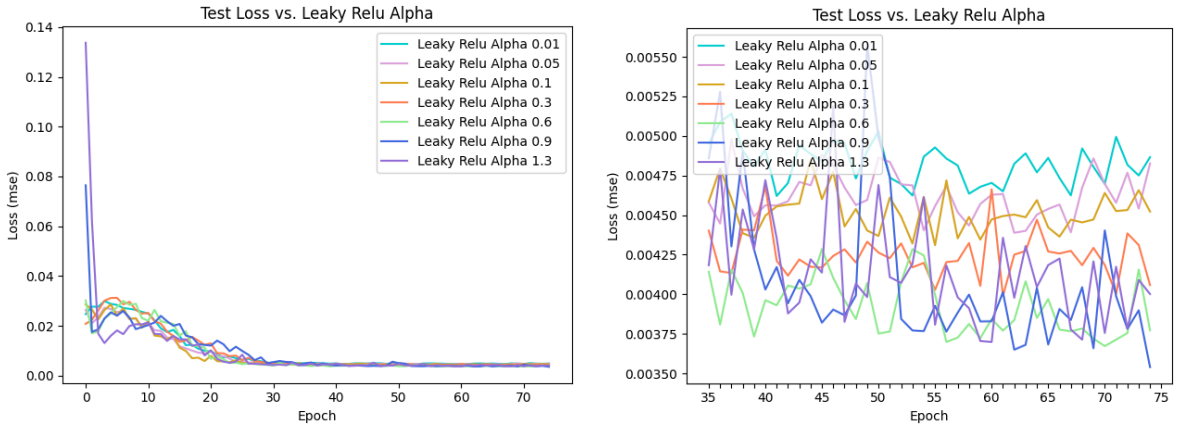


***Figure 24**: Loss for the test set for 75 times, according to the α value of the Leaky ReLU function. On the right, the values are from epoch 35 for better interpretation of the results.*

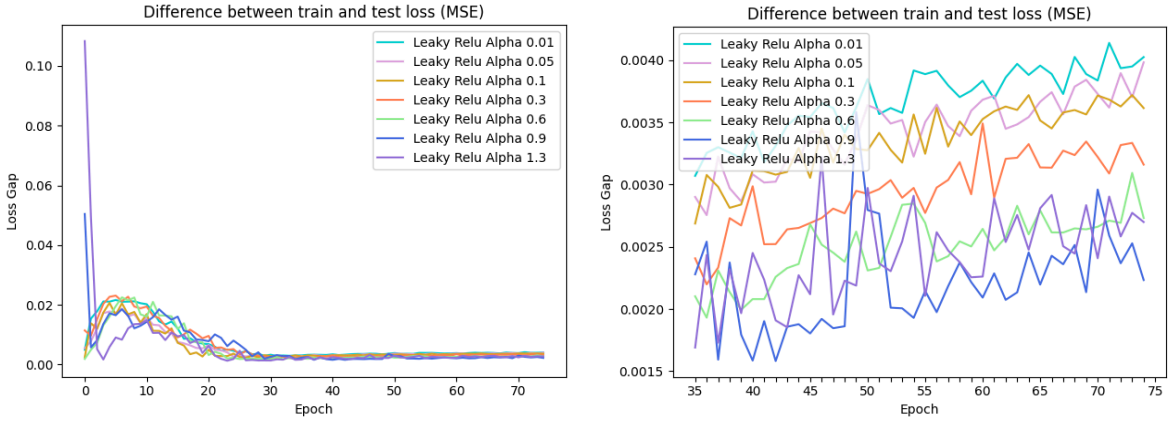


***Figura 25**: Gap Loss during the testing of an autoencoder for 75 times, according to the α value of the Leaky ReLU function. On the right, the values are from the time 35 for better interpretation of the results.*

The values selected were 1. 3 and 0. 9. At the time of observation Fig. 24 and Fig. 25, 0. 6, 0. 9 and 1. 3 seemed to have the best performance, despite their erratic behavior. However, when observing the reconstructions they generate, it is possible to notice that the α value of 0. 6 reconstructs certain images unsatisfactorily. Although the problem of reconstruction was mentioned when drawing conclusions, the need to limit the possible combinations for the next section was what pushed the authors to delete 0. 6 from the selection.

6.3. Architecture Filter

One of the main shortcomings of the parameter analysis was the randomness in the selection of images to analyze the reconstruction. Studying 200 images manually is a tedious task, which results in the last reviewed images not having a similar criterion to the first.

It was decided to choose six random images by *autoencoder* for analysis. At the time it did not seem a mistake to select these images at random, but then it became clear that those *autoencoders* that had images with exaggerated expressions in the selection had the worst reconstructions.

The reason why a uniform selection was not used was because the trained models had not been stored, which was a learning experience for the next stages of the project. It was decided to continue with the *autoencoders* corresponding to the selected values, taking into account that this should not be taken as a conclusive test, as mentioned several times in section **6.2.** A quick review of the 200 images of the reduced group described below revealed good reconstructions for most of the images. However, it is important to keep in mind that a more suitable *autoencoder* for this dataset might have been discarded.

As previously mentioned, the selected parameter values were the result of the analysis of *Loss, Gap Loss* and image reconstructions. As a summary, the final elections will be listed here:

- Optimizer learning rate: 10^{-3}, 10^{-4} y 10^{-5}.
- Number of convolutional layers: 3.
- Amount of data for gradient update: 8 and 32.
- Dimensions of the latent layer: 64 and 128.
- Activation function: 0. 9 and 1. 3.

One of the options considered was to perform the combinatorics of these parameters and analyze all the corresponding models. This resulted in 24 models but it was decided not to take this path for the following reasons: first, the number of training times of the previous *autoencoders* was 75 and, as mentioned, certain parameter values needed more iterations to achieve a reconstruction comparable to the architectures that could converge. Secondly, the training-test separation of the dataset remained fixed at all times. It was of interest to perform cross-validation to analyze how this separation was affecting the performance of the *autoencoders*.

It was taken as an alternative to perform a kind of “filter” of the previous values, reducing the number of architectures to be studied in the third stage to 8. On the one hand there would be the autoencoders whose training times would remain similar to the previous step. This results in the selection presented in ***Table 1***.

Nombre	Épocas	Tamaño batch	Filtros	Leaky ReLU Alpha	Adam LR	Capa Latente	Núcleos
ID1	80	8	[16, 32, 64]	0,2	1,00E-03	64	[(3, 3), (5, 5), (7, 7)]
ID2	80	8	[16, 32, 64]	0,2	1,00E-04	64	[(3, 3), (5, 5), (7, 7)]
ID3	80	8	[16, 32, 64]	0,9	1,00E-03	64	[(3, 3), (5, 5), (7, 7)]
ID4	80	8	[16, 32, 64]	1,3	1,00E-03	64	[(3, 3), (5, 5), (7, 7)]

Table 1: *Selection of autoencoders, keeping number of times.*

The initial values observed in the section Appendix, Initial parameters of the ABC were maintained and, for each, the corresponding variation has been modified for two parameters: learning rate (Adam LR in **Table 1**) and parameter α of the Leaky ReLU activation function (Leaky ReLU Alpha in **Table 1**).

Note that the use of *batch* 32 and 10^{-5} learning rate has been discarded as these require more training iterations. On the other hand, it was decided not to use the latent layer of

dimension 128 because the results obtained when the default value 64 was used were satisfactory..

Nombre	Épocas	Tamaño batch	Filtros	Leaky ReLU Alpha	Adam LR	Capa Latente	Núcleos
ID5	1300	32	[16, 32, 64]	1,3	1,00E-05	64	[(3, 3), (5, 5), (7, 7)]
ID6	1300	32	[16, 32, 64]	1,3	1,00E-05	128	[(3, 3), (5, 5), (7, 7)]
ID7	1300	32	[16, 32, 64]	0.9	1,00E-05	64	[(3, 3), (5, 5), (7, 7)]
ID8	1300	32	[16, 32, 64]	0.9	1,00E-05	128	[(3, 3), (5, 5), (7, 7)]

***Table 2**: selection of autoencoders, increasing number of times.*

Secondly, there are those autoencoders that required more training times, which are listed in **Table 2.** Here you can see that the values for optimizer learning rate and *batch* size that need more iterations are used. On the other hand, the combinatorics of the selected values for the dimension of the latent layer and the parameter α of the Leaky ReLU activation function were added.

For the autoencoders in **Table 1**, cross validation was performed with parameter k=5, where k is the value that determines the number of fragmentation blocks. This was not done for the *autoencoders* in **Table 2** because of the amount of time it took to train them: 14.5 hours for each *autoencoder,* which would give a total (including cross-validation) of 232 hours, approximately 10 days.

This decision resulted in a total of 24 models: 20 as a result of cross-validation experiments from ID_1 to ID_4 and the last four are those of **Table 2**, with 1300 training periods each.

6.4. Comparison methodology with Rekognition

The goal of the last step, mentioned above, is to choose the autoencoder that will be used for the experiments. For each of the models resulting from the previous step, two folders were generated, which will be named `/recon` and `/data`. Images of the unmodified dataset are stored in the `/data` folder. Note that the `/data` folder will have the same information for each of the models On the other hand, in the `/recon` folder older you will find the reconstructed images of the model in question. Each image in the recon folder will have its corresponding image in the `/data` folder.

To evaluate the performance of the autoencoders, it was decided to use the tool that, ultimately, is targeted: Amazon Rekognition. From the functionalities mentioned in Section **5. 2. 2.** , to obtain the following metrics, with respect to the images in the `/recon` folder:

- ***Similarity***: To get this value, the two-image comparison feature is used: the reconstructed image stored in `/recon` against its respective original image in `/data`. This similarity is defined by the Amazon Rekognition model and its calculation works as a black box for authors.
- ***Confidence***: this parameter is found in the response obtained from the detection of an individual and indicates the confidence with which Amazon Rekognition determines that it found the face in the image.

- ***Quality Sharpness***: this value comes from the face detection functionality. The goal is to analyze this metric for the reconstructed image, so that it can be evaluated which model takes higher values in terms of quality.
- ***Quality Brightness***: this value has a similar objective to the one mentioned above with regard to the analysis of image quality. It is obtained with the same functionality and is made on the image reconstructed by the *autoencoder.*

For each model, the mean and standard deviation of the obtained metrics will be taken. The results are shown below (the table with the corresponding data can be found in the Appendix, Data obtained with Rekognition for *Autoencoder* selection). It is important to note that the priority metric for model selection is *Similarity*. The search was conducted for the highest values that the models could achieve, alongside the confidence with which Amazon Rekognition identifies a face in the image reconstructed by the *autoencoder*. Another aspect complementary to the decision was the quality analysis. Although quality did not determine the choice, it was necessary to confirm that the images of the model chosen by the *Similarity* metric did not lose quality significantly with reference to the other models.

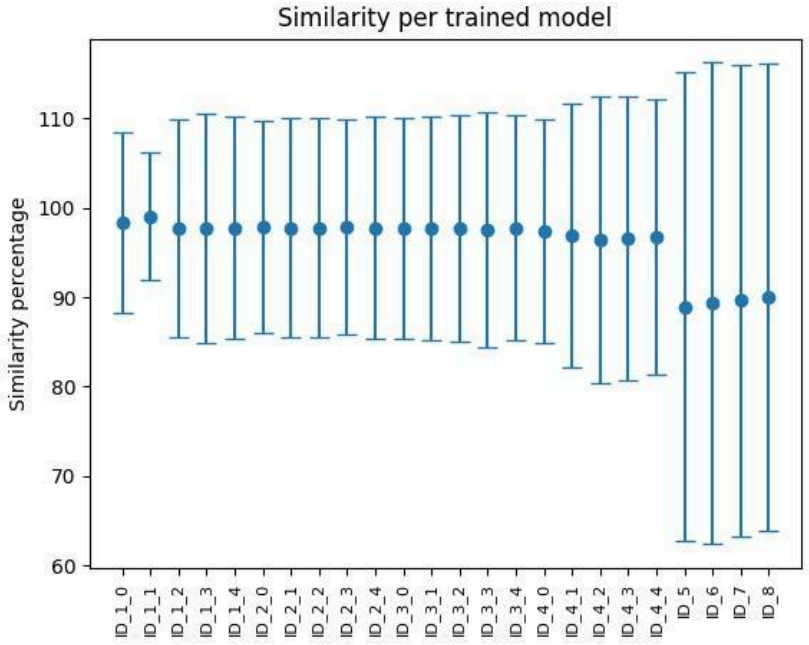


***Figure 26**: results obtained with respect to the Similarity metric for the trained models. Those whose tag is of the form ID_n_m correspond to models with cross-validation.*

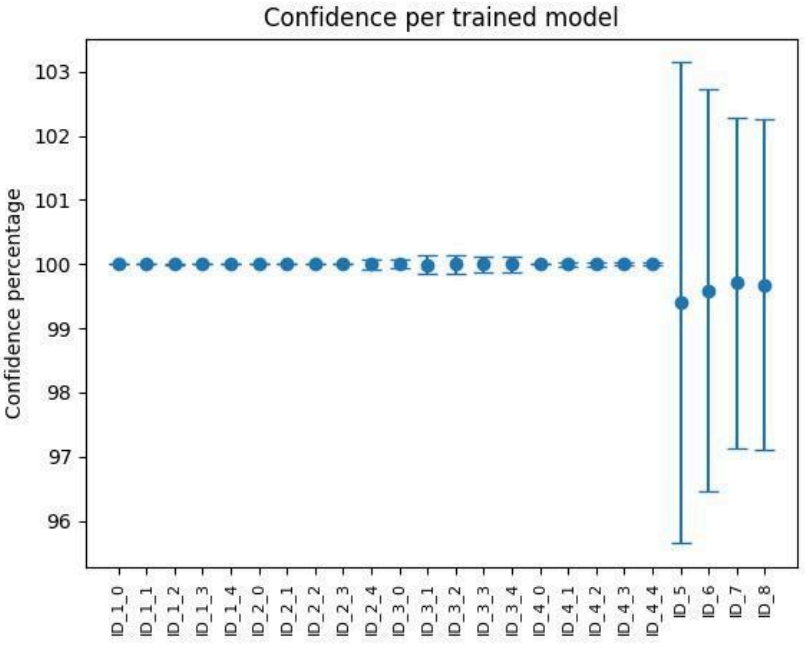


***Figure 27**: results obtained with respect to the Confidence metric for the trained models. Those whose tag is of the form ID_n_m correspond to models with cross-validation.*

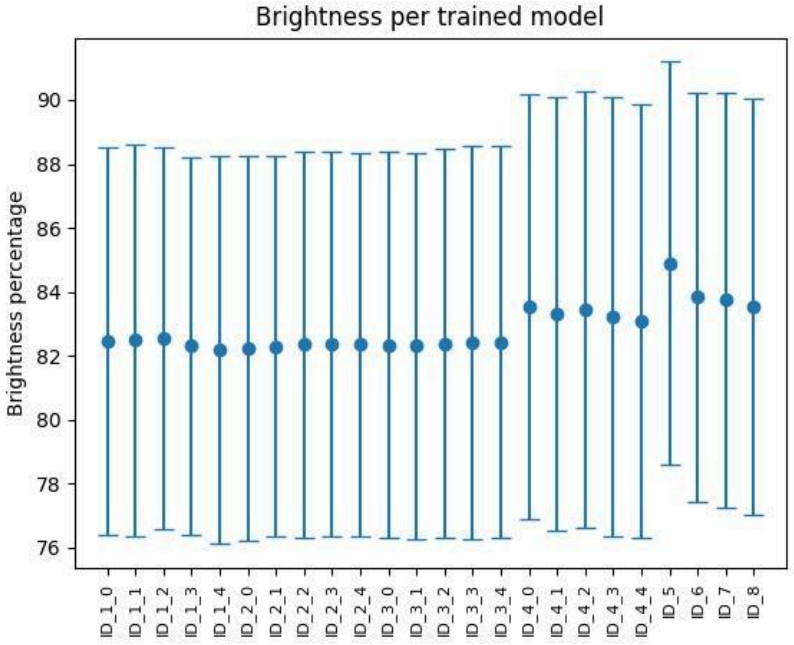


***Figure 28**: results obtained with respect to the Quality Brightness metric for the trained models. Those whose tag is of the form ID_n_m correspond to models with cross-validation.*

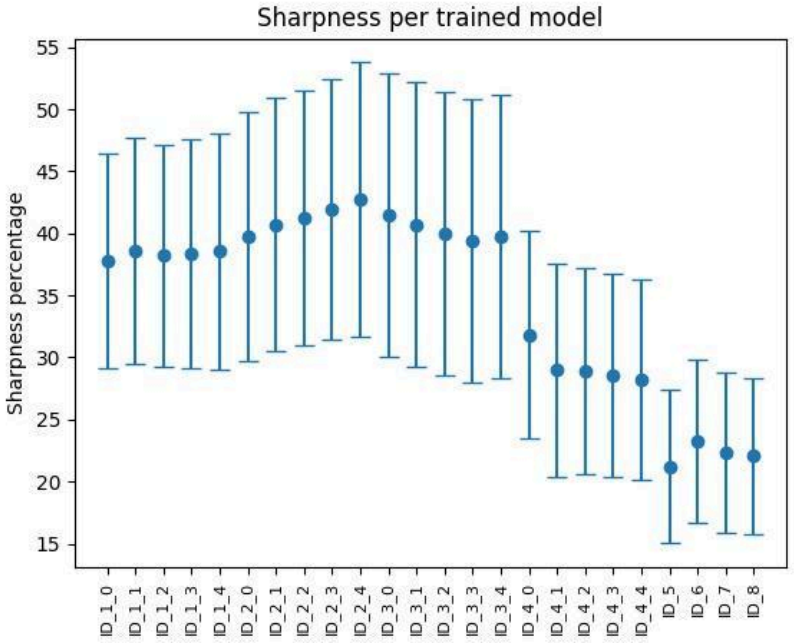


***Figure 29**: results obtained with respect to the Quality Sharpness metric for trained models. Those whose tag is of the form ID_n_m correspond to models with cross-validation.*

In **Fig. 26** it can be seen that the model ID_1_1 has the highest *Similarity* metric at the same time that its standard deviation is the smallest. Therefore, it was decided to take this model for experimentation. See that in **Fig. 27** the confidence with which faces are recognized is centered at 100% with a very small standard deviation. On the other hand, although the percentage of quality for both the *Brightness* and *Sharpness* metrics is low, the results are not very different from the other trained models. From the next sections of the paper, when talking about autoencoder, reference will be made to ID_1_1.

It is important to note that models ID_5 to ID_8 behave differently from the others. The metric values, except for the brightness of the images, are noticeably lower than the ID_1 to ID_4 models. An inspection of the images reconstructed by these models (of which some examples can be seen in **Fig. 30**) showed that they were still blurred, a possible indication of the need for more periods. It was decided not to continue with the analysis for the selection of *autoencoder* in order to allow more time for experimentation.

***Figure 30**: reconstruction of three images using the ID_8 model.*

A question that may arise when studying the complete methodology (from parameter analysis to this same section) is the fact that in the parameter analysis and subsequent filtering of architectures different criteria are used to those just described. This may be seen as a mistake or a disadvantage, as using the same metrics would allow us to maintain a unified approach to analyzing whether the results improved or worsened by combining features of the original *autoencoders.*

When defining an *autoencoder* from among the 24 available models, questions began to arise about the simplification of the analysis performed by taking 6 images out of 200 as a subset of the study, in addition to the problems of quality and reliability in the recognition of a face. The methodology used at the beginning has a level of subjectivity considered very high according to the authors, since the metrics mentioned here were analyzed manually, all at the same time.

When looking for alternatives, the potential of Rekognition as a tool to increase the objectivity of the choice of the *autoencoder* was recognized. The Rekognition result provided information that was implicitly analyzed manually. An important advantage in the change of methodology is the obtaining of values analyzed in a consistent way, regardless of the number of images to be studied (which is not possible for a subjective human subject).

7. Analysis of Principal Components

This section explains in detail the use of Principal Component Analysis in the context of the work. As mentioned in section **4. 2.**, attempts will be made to generate adversarial examples that allow evasive and/or impersonation attacks to be carried out. PCA is used on the representations in the latent space of the set of images and, through the manipulation of the resulting projections, seeks to obtain potential adversarial examples.

7.1. Generation of Models

This first part describes the procedure that makes it possible to acquire the necessary tools to obtain projections on main components from images and vice versa. The following steps will be performed only once, since the used libraries allow to store the models (with their respective parameters) for later reuse.

1. Representations in latent space of the dataset are obtained using the *encoder* of the autoencoder. This results in 200 vectors.
2. The set of representations obtained in 1 is standardized, using an instance of the `StandardScaler`, class from the `sklearn.preprocessing` library.
3. Principal component analysis is applied on the set obtained in 2, using an instance of the PCA class from the library `sklearn.decomposition`.
4. The models corresponding to the `StandardScaler` and `PCA`. are stored in two *.joblib* files.

Since each representation in latent space is 64x1 and there are 200 representations, the number of eigenvalues (with their corresponding eigenvectors) resulting from applying PCA is 64. **Table 3** shows the ratio of variance explained for the eigenvalues. The list of eigenvalues can be found in the Appendix, under the Principal Component Analysis section.

Eigenvalues, Explained Variance Ratio							
A1	0,16794744	A2	0,1064457	A3	0,064212084	A4	0,0529059
A5	0,04945347	A6	0,047460143	A7	0,04176735	A8	0,032769542
A9	0,03206953	A10	0,027543474	A11	0,02566085	A12	0,024135994
A13	0,023233654	A14	0,021846538	A15	0,01997548	A16	0,019750385
A17	0,017400328	A18	0,016722362	A19	0,015929572	A20	0,014889501
A21	0,012427767	A22	0,0115133375	A23	0,011165145	A24	0,010974049
A25	0,010307819	A26	0,009446151	A27	0,008347811	A28	0,0075190715
A29	0,0072452654	A30	0,0067230794	A31	0,0065918486	A32	0,006465569
A33	0,005721348	A34	0,0054692533	A35	0,004821858	A36	0,004697566
A37	0,0045182505	A38	0,003935131	A39	0,003792166	A40	0,0035469485
A41	0,0029739758	A42	0,0028897158	A43	0,0027488023	A44	0,0026673602
A45	0,0024066838	A46	0,0021420482	A47	0,0019885378	A48	0,0017795902
A49	0,0016003761	A50	0,0013683785	A51	0,0013173097	A52	0,0011832417
A53	0,0011094314	A54	0,0010149184	A55	0,0008864566	A56	0,00082319847
A57	0,00077462324	A58	0,00061722647	A59	0,0005423088	A60	0,0005210481
A61	0,0004198526	A62	0,00038504144	A63	0,00036332934	A64	0,000127693

***Table 3**: ratio of variance explained by eigenvalue.*

7.2. Functionalities

Once the models have been obtained, they shall be used to:

- Get the projection into main components of an image i:
    a. The image i is sent to the autoencoder encoder and the latent space representation z is obtained.

    b. Standardise z using the `StandardScaler` model.
    c. Obtain the principal component projection using the `transform` function of the PCA model.
- Obtain an image from the principal component projection:
    a. The `inverse_transform` function of the PCA model is used on the vector in question.
    b. The inverse operation is applied to the standardisation by obtaining a vector z, using the `inverse_transform` function of the `StandardScaler` model.
    c. Send the vector z to the *autoencoder* decoder to obtain the image.

8. Experimentation

As can be recalled from section **4.3**., the objectives for experimentation include achieving, firstly, adversarial examples that allow an evasion attack and, secondly, an impersonation attack.

One problem with regular *autoencoders* is that they do not have a structure for the latent space. This results in difficulties when generating images from the latent space, as there is no relation between proximity (e.g. according to Euclidean distance) and decoding result, as is the case with variational *autoencoders*.

Another challenge concerns the way the latent space is traversed. A naive and computationally expensive solution would be to traverse the space by brute force (with a defined step), analysing the existence of adversarial examples. This methodology is not very attractive, given the *hardware* limitations outlined in section **4.4.1.**

The use of principal component analysis seeks to provide a hierarchical ordering to the latent space. The hierarchy is given by the decreasing order of variability represented by each eigenvector of the principal component analysis. Of interest is the possibility to relate certain components to identity allowing the construction of an adversarial example generation strategy oriented to them.

In the following, the experiments will be presented in chronological order. They are divided into the following sections:

- **Background information**: optional section containing information necessary for the conduct of the experiment.
- **Hypothesis**: includes not only the propositions to be tested but also the explanation why the hypothesis was chosen.
- **Procedure**: details the steps for conducting the experiment.
- **Analysis**: discusses whether or not the evidence supports the previously formulated hypothesis and includes additional comments on new avenues of exploration.
- **Additional Information**: optional section containing information necessary but not pertaining to the procedure and/or results.

8.1. Experiment 1

8.1.1. Hypotesis

It is possible to find a relationship between the first two principal components and the identity of each individual. Although the explained variance summed by these components is 26%, which is not particularly high, it is important to note that the eigenvalues corresponding to these components are an order of magnitude larger than the rest.

Modifying these components will not modify the label assigned by the oracle O (see section **2.1.**), as the other 62 components (with 74% of the variance explained) will remain fixed, but it will generate substantial changes in Rekognition as they are the two most important ones.

8.1.2. Procedure

1. Principal component projections are obtained for all images in the dataset.
2. The average projection of all principal components is obtained for Marina and Ignacio separately. The images corresponding to these components are obtained and the "average" images of each individual are stored.
3. The projections of the first two principal components are plotted and it is analysed whether a separation can be made between the points corresponding to Marina and Ignacio. If such a separation is not possible, the experiment is terminated.
4. The limits of Ignacio's group for each component are taken from the graph in 3. From the average image of Marina, the 0 and 1 components are modified according to the limits obtained previously. A step of 0.2 will be used to traverse these components. The image corresponding to the result is obtained and stored.
5. Repeat 4, taking the limits of Marina's group and using Ignacio's average image.

8.1.3. Analysis

When analysing the graph made in step 3, a clear separation of the data between the two individuals can be observed. This provides evidence in favour of the first part of the hypothesis, as the first two principal components seem to be related to the identity of the person in question.

***Figure 31**: On the left, average image of Ignacio. On the right, average image of Marina..*

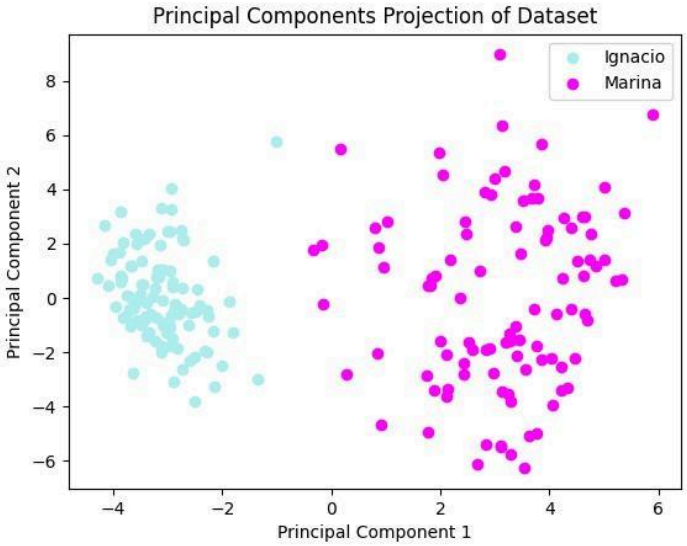


***Figure 32**: projection of the first two components of the dataset.*

As can be seen in **Fig. 32**, the data set corresponding to Marina is more spread out than that of Ignacio. This difference could be explained by observing that Marina's images have more varied expressions and focus angles than those of Ignacio..

***Figure 33**: Manipulation results from Ignacio's average image.*

Figure 34*Manipulation results from Marina's average image..*

Both **Fig. 33** and **Fig. 34** show the original image in the top row, with the corresponding manipulation in the bottom row. In Fig. 33 it is possible to observe that, starting from an image whose label is Ignacio (top row), the images resulting from the manipulation verify that the label assigned by O is Marina (bottom row). Then, in **Fig. 34** the same results occur for the reverse situation: the modification of the first two principal components causes a change of label from Marina to Ignacio. The changes made to the values of the first two components are detailed in the Appendix, in the section Detail of Modifications for Experiment 1.

The experiment showed that the second part of the hypothesis is not verified since by modifying the projections of the first two principal components of one individual with those of the other, the classification given by the oracle O changes.

8.2. Experiment 2

8.2.1. Hypothesis

The first principal component is the only direction in which a clear separation between Marina's and Ignacio's data is observed.

As can be seen in **Fig. 32** the separation into groups only occurs along the X-axis, corresponding to the first principal component. The behaviour of the second component is expected to be repeated for the remaining 62 components

8.2.2. Procedure

1. Principal component projections are obtained for all images in the data set
2. The projections obtained in 1 are plotted for each principal component. In each projection, Marina's and Ignacio's points are distinguished with different colours to analyse if there is a separation.

8.2.3. Analysis

The experimentation supports the hypothesis. It can be observed that the separation into groups occurs only in the first principal component. In the rest, Marina's and Ignacio's values

are overlapping and, in general, Marina's values have a larger range of variability than Ignacio's values.

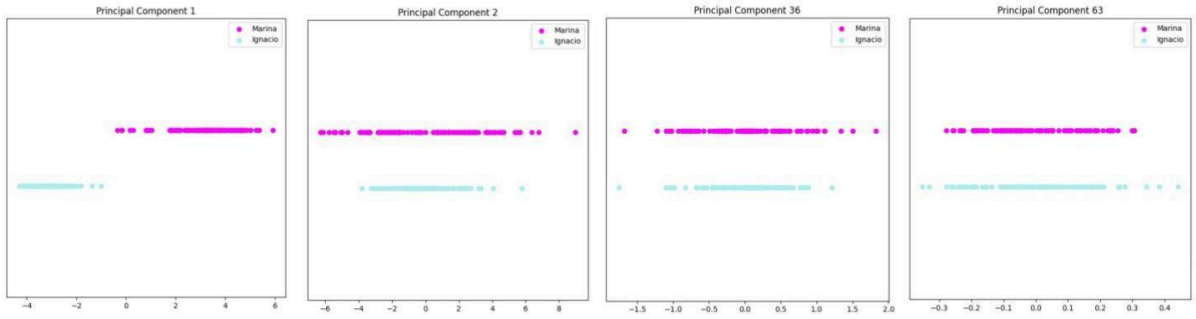


***Figure 35**: Representation of the elements of the data set according to the i-th principal component*[7].

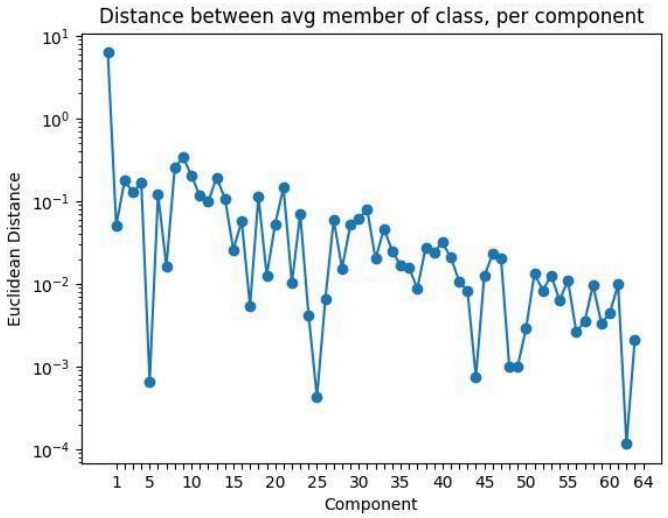


***Figure 36**: Euclidean distance between Ignacio's and Marina's averages, by principal component.*

Once the principal component projections of the data have been obtained, the separation between the average value of Marina's and Ignacio's data, by principal component, can be analysed in Fig. 36. This graph provides greater clarity when analysing the hypothesis, as it is clear that the first principal component shows a significantly greater Euclidean distance between averages than the rest of the components. There even seems to be a downward trend with respect to the distance between the averages.

8.2.4. Additional information

During the course of the experiment it was decided to replicate the procedure for the latent space representations of the data. The aim was to analyse whether such a separation is observed in any dimension of this space.

It could be observed that, although there are no dimensions with a disjunct separation between Marina and Ignacio, there are directions that seem to have a "predominance" of one individual or the other. In **Fig. 37**, dimension 53 has a predominance of Ignacio, with

[7] Graphics for all major components can be found in this folder.

Marina's data clustered together, with little variability. The reverse is true for dimension 3, with Marina's predominance.

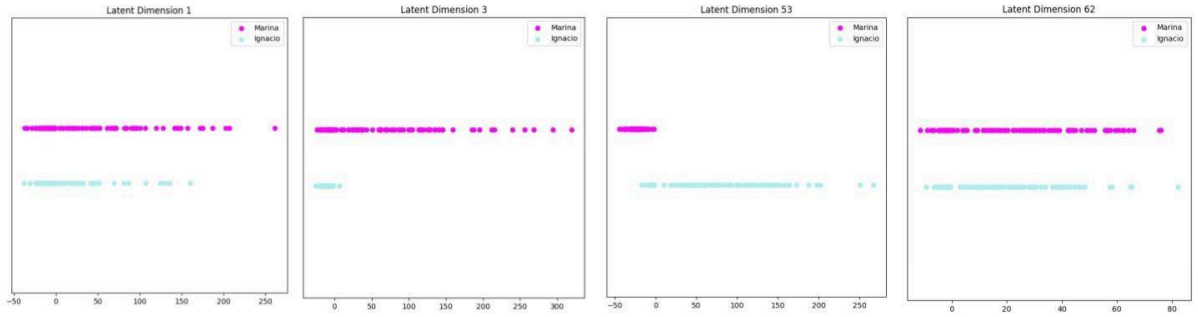


***Figure 37**: Representation of the elements of the dataset according to the i-th dimension of the latent space[8].*

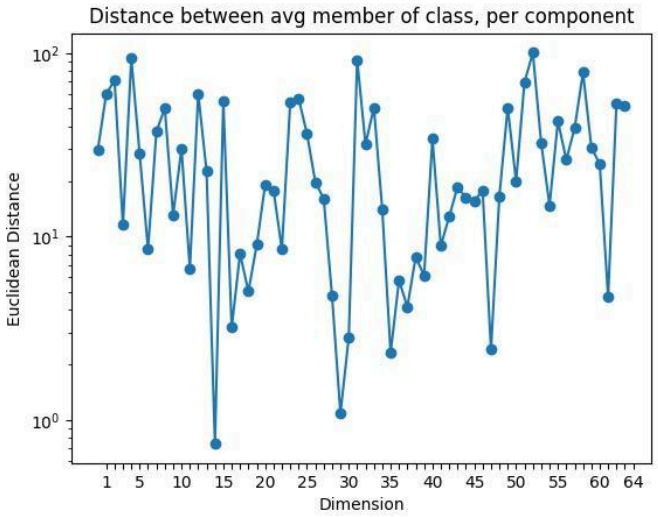


***Figura 38**: Euclidean distance between the average of Ignacio and Marina, by dimension of the latent space..*

For each dimension of the latent space, as in **Fig. 36**, the average was calculated for Ignacio's data and, separately, for Marina's data. Then, the Euclidean distance between the averages for each individual was measured. As can be seen in **Fig. 38**, there does not seem to be a clear trend in the distances between the averages.

In conclusion, it is believed that the behaviour of the latent space directions provides the possibility to build a new strategy for generating adversarial examples, in addition to the one that is leveraged on the use of principal components. This will be discussed further in section **9.1.**

8.3. Experiment 3

8.3.1. Hypothesis

From the average of the principal component projections of the dataset, the modification of the first principal component allows to control the label assigned by the oracle O.

[8] Graphs corresponding to all dimensions of latent space can be found in this folder.

8.3.2. Procedure

1. The principal component projections of all the images in the dataset are obtained.
2. The average of the projections obtained in 1 is obtained. The image corresponding to the result is obtained and stored.
3. The minimum and maximum value that the first principal component can take according to **Fig. 32** is taken. From the average obtained in 2, the first component is varied in the acquired range with a step of 0.2.

8.3.3. Analysis

***Figura 39**: decoded images after modifying the first principal component from the average image..*

The results obtained support the hypothesis. As can be seen in **Fig. 39**, depending on the value of the first principal component, the label assigned by . It is important to note that this variation is performed on the average image, which has an empty expression (see **Fig. 40**).

***Figura 40**: Average image*

8.4. Experiment 4

8.4.1. Hypothesis

Principal components 2 to 64 control the expression and/or angle of the image. Modifying these components allows a face to change its expression without modifying the label assigned by the oracle O.

8.4.2. Procedure

1. Nine images are manually selected in such a way that they are diverse in their expressions. The chosen images can be seen in **Fig. 41.**
2. Combinations are made from the images in Fig. 41. Each combination will have an original image and a reference image. From the original one, the aim is to keep the label assigned by O and from the reference one, to obtain the expression of the face.

Oiriginal	Referencia
marina0	ignacio40
marina69	ignacio40
marina41	ignacio40
ignacio37	marina69
test-ignacio9	marina69
ignacio50	marina69
ignacio40	marina64
marina69	ignacio22

***Table 4**: Image combinations used in the experiment.*

3. The following steps are performed on each combination in ***Table 4***:
4. The principal components of the original image, which we will call y_o are obtained from the reference image y_r.
    a. A new vector is created $y*$ using the value of the first component of y_o and for the rest of the components, the values of y_r.
    b. The image corresponding to $y*$ is obtained and the result is stored.
    c. The images corresponding toy_o, y_r and $y*$ are plotted to observe the changes.

***Figure 41**: Selected images for experiment 4..*

8.4.3. Analysis

The results obtained provide evidence in favour of the proposed hypothesis. For different y_o, it was decided to use the same reference image y_r as it was expected that the expression of the result $y*$ would be similar to y_r that of and independent of the expression of y_o. The characteristic that was sought to be maintained from y_o was the label assigned by O. It can be seen in ***Fig. 42*** that all images $y*$ result in the same expression keeping $O(y_o)$.

***Figure 42**: original images of marina with reference from ignacio*

The same results are achieved by reversing the situation: y_o corresponds to different images of Ignacio and corresponds to an image of Marina. In ***Fig. 43*** we can see that the images $y*$ behave in a similar way to what was analysed above. The difference is that the resulting decoding generates images with significant noise that distorts the features of the face.

Figure 43*: Original images of Ignacio with reference from Marina.*

The appearance of noise in $y*$ does not necessarily imply that the change of expression has failed. For this reason it was decided to use an image y_r with a more distinctive expression to make his presence apparent $y*$ in spite of the noise. In ***Fig. 44*** it can be seen that the smile of y_r appears in y_* despite the distortion introduced.

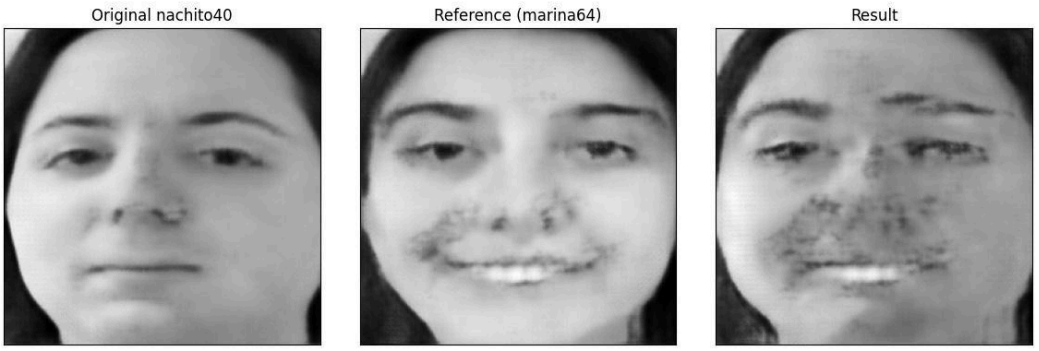


***Figure 44**: Original image of Ignacio with reference to Marina with distinctive expression*

In the examples presented in **Fig. 42** and **Fig. 43**, it can be seen that the reference image has a serious expression, similar to the average image. It was of interest to analyse whether the reverse situation was possible: going from a serious expression to a more exaggerated one. The results of the combinations corresponding to **Fig. 44** and **Fig. 45** show that it radically changes its expression, maintaining $O(y_o)$.

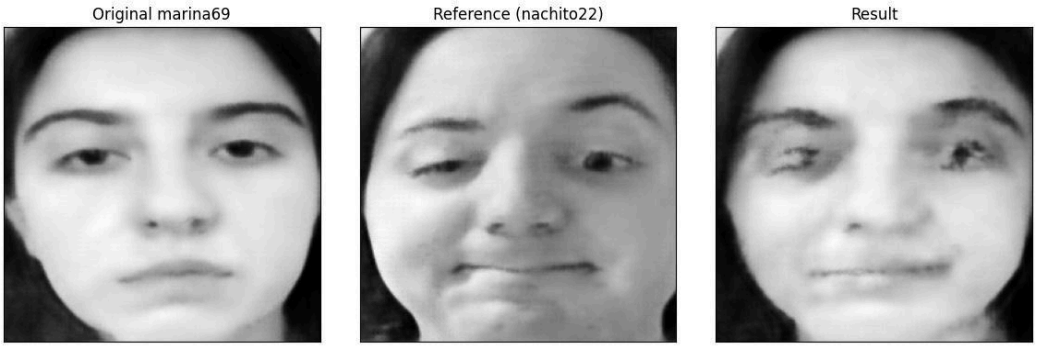


***Figure 45**: Original image of Marina with reference to Ignacio with distinctive expression*

In conclusion, the results increase the evidence in favour of the hypothesis put forward in the previous experiment: the first principal component controls the oracle label assigned to the image. On the other hand, the combination of the remaining 63 manipulates the expression of the image. There is no knowledge about the semantic meaning of each component separately, and it could even happen that such meaning does not exist and only arises from the combination of the components.

Finally, the presence of distortions when going from Ignacio to Marina or from simple to complex expressions, as in **Fig. 45**, shows the need to study the quality of the images resulting from the experimentation process. It is from this reflection that the approach of an objective quality analysis proposed in section **4.** and described in section **8.6.1**. arises.

8.5. Experiment 5

8.5.1. Hypothesis

While the last 63 principal components control facial expression, each separately has semantic meaning. For example, one principal component controls the position of the eyes, another the shape of the mouth, and so on.

8.5.2. Procedure

1. The principal component projections of all images in the dataset are obtained.
2. The average of the projections of 1 is calculated.
3. The average vector is taken and, for each component, the following procedure is carried out:
4. We take the minimum and maximum that the component i takes for the projections obtained in 1. Then we divide the range into eight values..
5. For each one, all the principal components are kept fixed, except for the i-th component, which takes the value in question..
    a. The vector of principal components is decoded into an image and stored[9].

8.5.3. Analysis

The results obtained in this experiment do not allow us to define whether the evidence supports or contradicts the hypothesis. By modifying certain components, visual changes are generated to which meaning can be assigned (such as the angle of rotation of the face). The problem is that such meaning is often not unique to a single feature on the face and, moreover, noise often appears as the component is modified.

***Figure 46**: Average image decoding as principal component 1 is varied.*

[9] The computation of the 512 (64 dimensions x 8 values) resulting images had to be split into small batches of 40 due to *hardware* constraints.

***Figure 47**: Average image decoding when varying principal component 5.*

***Figure 48**: Average image decoding when varying principal component 3.*

***Figure 49**: Average image decoding when varying principal component 9.*

***Figure 50**: Average picture decoding when varying principal component 41.*

For example, in **Fig. 47**, a rotation of the face can be observed. However, it also appears that the expression changes to one of anger as the maximum values of component five are reached. The rotation of the face not only appears in **Fig. 47** but can also be found in component three as seen in **Fig. 48**. Unlike the previous component, in this case a considerable amount of noise is produced, as in **Fig. 49**. Finally, it happens that for the last principal components the changes become more subtle to the human eye, as evidenced in **Fig. 50**, which shows the results for component 41.

8.5.4. Additional information

It is decided to replicate experiment 5 on the images of the dataset, encoded in latent space. Since PCA failed to provide evidence to support the hypothesis, we seek to analyse whether the disorganised latent space allows modifying point features through movements in specific directions in the latent space.

The procedure remains the same, with the exception of point 1 in which the latent space representations are obtained instead of the projections in principal components.

***Figura 51**: Average image decoding by varying dimension 4 of the latent space.*

***Figura 52**: Average image decoding when varying latent space dimension 10.*

Figure 53: *Average image decoding when varying dimension 20 of latent space*

Figura 54: *average image decoding when varying dimension 34 of latent space*

Based on the results obtained, observations similar to the original experiment can be made: there are dimensions that do not produce significant changes and dimensions that do but do not generate a single change and that often add noise.

In conclusion, if the goal is to modify specific features of the face, it would be difficult to do so through particular directions both in latent space and in the directions of the main components.

Attempts could be made to add more individuals to the dataset to study how it impacts the separation of characteristics along different directions. It is believed that because it has little variability of faces, the *autoencoder* does not generalize enough about elements of a face..

Another alternative, since the previous one might be insufficient, is to observe what happens when different main components or latent space directions are combined. It may be necessary to study the directions together to find ways to make punctual changes to the faces.

8.6. Experiment 6

8.6.1. Background information

The aim of this section is to obtain metrics on how images of both Marina and Ignacio are recognized in Rekognition. It is important to remember that Rekognition has a collection of images against which it will perform comparisons in case of using the “Compare Similarity Against a Collection” feature described in Section **5.2.2.** This collection consists of the characteristics of the images of the dataset after passing through the *autoencoder*, for the reasons explained in Section **5.2.**

The aim is to obtain “base” measurements of the confidence in facial recognition, similarity and quality of the collection stored by Rekognition, and then analyze potential adversarial examples based on these initial values. It was explained earlier that the adversarial examples would be generated by the *autoencoder*, with which it is considered that the best values of quality and similarity that can be achieved are those corresponding to the images used to train and evaluate the autoencoder. An adversarial example will only be considered as such if its *Quality Brightness* and *Quality Sharpness* values are around the average obtained for the unmanipulated images.

Rekognition allows you to determine a parameter called `face_threshold`, between 0% and 100%, allowing you to define which results are considered: if the similarity with an image is greater than or equal to `face_threshold`, el “*match*” the “match” will be included in the result. Otherwise, it will be discarded. It was decided to use a `face_threshold` of 0% to analyze all comparisons against the collection. The procedure can be defined as follows:

1. The images of the dataset are passed through the *autoencoder* and stored. Each image
    a. It is sent to Rekognition and the “Compare similarity against a collection” functionality is used. This allows you to get reliable results in face recognition and similarity against the collection that Rekognition has stored.
    b. It is sent to Rekognition and the “Analyze Image Quality” functionality described in Section **5.2.2** is used. The goal is to get the quality metrics, *Quality Sharpness* and *Quality Brightness*, for each image.
2. Each result obtained in 1a, corresponding to an individual (Marina or Ignacio), has a list of “*matches*” between the image in question and those stored in Rekognition. For this image, we averaged the similarity corresponding to Marina's images and, then, to Ignacio's images. This average is stored with the label of the individual in the image.
3. Separate the values obtained in those corresponding to the label Marina (there will be 93 averages since there are 93 images of Marina) and to the label Ignacio (same number of averages). From this separation, the average and standard deviation of the 93 values are calculated for each label.

4. The mean and standard deviation metrics for confidence in facial recognition and quality are computed from the results obtained in 1b..

Target	Marina Mean	Marina Standard Deviation	Ignacio Mean	Ignacio Standard Deviation
Marina	99,72605991	0,83894468	1,845080529	2,18497707
Ignacio	1,842878439	2,00393929	98,94514483	3,01514133

Table 5: Rekognition similarity analysis

Metric	Mean	Standard Deviation
Confidence	99,99999224	1,48E-05
Quality Brightness	83,02473108	5,978395951
Quality Sharpness	40,98942438	8,595760332

Table 6: *Rekognition quality analysis*

Table 5 shows the results of item 2 of the procedure and **Table 6** shows the results of item 3.

8.6.2. Hypothesis

Facial recognition systems are not based solely on the overall appearance of an image for the recognition of the individual. As explained in Experiment 3, it is considered that the label assigned to an image by the oracle O can be controlled by modifying the value of the first major component. When manipulating an image to modify its original label, the similarity percentages resulting from analyzing the image with Rekognition will not necessarily be close to those in **Table 5**.

For example, if a manipulated image labeled as O “Marina” is sent to Rekognition, it may happen that the average of “matches” with Ignacio takes a value greater than 1. 84508052% and, on the other hand, that of Marina takes a value less than 99. 72605991%. The same would happen with an image labeled by the oracle as “Ignacio”.

8.6.3. Procedure

1. The main component projections of the data set images are obtained.
2. Two ranges are defined to change the value of the projection in the first main component
    a. To make the identity match Marina, use the interval [1, 6. 5] with a step of 0. 2.
    b. To make the identity match Ignacio, use the interval [-5, 1. 5] with a step of 0. 2.
    c. For each vector obtained in 1 new projections are generated by modifying the first component according to the range defined in 2 according to the label assigned by O the corresponding vector image.
    d. If the oracle label is Ignacio, the range defined in 2a will be used.
    e. If the oracle label is Marina, the range defined in 2b will be used.
3. The resulting vectors are decoded into images and the results are stored.
4. 18 images are selected whose original label has changed and whose quality seems to be maintained, according to the human eye. Since this selection was manual from

4600 images, the subjectivity of this step[10] must be taken into account. Each of the images was sent to Rekognition to obtain

a. Similarity metrics with respect to images stored in Rekognition.
b. Confidence in facial recognition.
c. Quality metrics: *Quality Sharpness* and *Quality Brightness*.

8.6.4. Analysis

After the experiment, the 18 images selected in item 5 of the procedure were analyzed, which had a change in the label assigned by O. The results of the Rekognition analysis on quality are shown in **Table 7** and those corresponding to the similarity analysis in **Table 8**.

Potential Adversarial Examples, Quality Analysis			
Source Image	Confidence	Quality Brightness	Quality Sharpness
ignacio0	100	87,70657349	32,20803452
ignacio11	99,99998474	89,30258179	26,17736816
ignacio14	100	85,84529114	26,17736816
ignacio21	99,99998474	90,79774475	32,20803452
ignacio29	99,99998474	88,12119293	26,17736816
ignacio49	99,99998474	89,27024841	32,20803452
ignacio51	99,99998474	84,99997711	38,89601135
ignacio54	99,99998474	90,41576385	32,20803452
ignacio76	100	87,24095917	32,20803452
ignacio87	100	83,67180634	46,02980042
ignacio88	99,99998474	89,55757904	32,20803452
test-ignacio0	99,99998474	90,46258545	32,20803452
marina17	99,99990845	82,61359406	32,20803452
marina21	99,99987793	83,24923706	32,20803452
marina29	100	82,19504547	46,02980042
marina33	99,99993846	85,35240936	38,89601135
marina50	99,99998474	77,64185333	32,20803452
test-marina7	99,99997711	77,64492035	38,89601135

Table 7: *Rekognition quality analysis on potential adversarial examples*

Potential Adversarial Examples, Similarity Analysis				
Source Image	Mean Marina Similarity	Stdev Marina Similarity	Mean Ignacio Similarity	Stdev Ignacio Similarity
ignacio0	98,07464083	3,76530556	3,42836851	3,28099627
ignacio11	99,84451138	0,44299136	2,67762129	2,72107566
ignacio14	98,04071176	3,12131098	9,27779568	9,02577925

[10] From the set of images obtained, they could have been sorted according to the quality determined by Rekognition and, once sorted, selected the 18 best ones to verify that the original label assigned by the oracle changed. The problem that was presented in this experiment, which had not been presented so far, was the limitation of Rekognition's *Free Tier*, mentioned in section 5. 2. 1. For both quality and similarity analysis, it was necessary to restrict the selected images due to the limit of 5000 images per month imposed by Amazon's free package.

ignacio21	99,55909073	1,05749549	3,92870466	4,17965053
ignacio29	99,94787491	0,18264075	2,00511547	2,19638218
ignacio49	99,75134868	0,70359387	1,96552379	2,94280761
ignacio51	99,42177319	1,47528882	2,15327145	2,34993002
ignacio54	99,26387926	1,56206441	2,06201546	1,97096455
ignacio76	96,17442076	7,39962654	6,47935669	7,31280913
ignacio87	99,58807677	1,34167541	2,65176981	2,70953122
ignacio88	99,47252073	2,23569500	3,59192174	4,39843860
test-ignacio0	99,75705399	0,91449670	2,81552409	2,62947759
marina17	1,08416005	0,86099082	91,84194524	11,20301741
marina21	3,83634580	3,71051312	46,84344557	22,6654669
marina29	35,37624416	18,83294737	62,60839676	23,33784421
marina33	64,07814949	17,81382552	69,9908113	21,38174483
marina50	26,38905084	16,75245342	81,39943349	17,91575476
test-marina7	17,73802099	12,09566476	66,42649591	22,02305054

Table 8: *Rekognition similarity analysis on potential adversarial examples*

The results are not conclusive with respect to the hypothesis put forward. First, it was analyzed whether Ignacio's images manipulated to look like Marina took different values from those in **Table 5** with respect to Marina's original images.

Fig. 55 shows the behaviour of the modified Ignacio images. On the axis we have the source image which, after manipulation, O assigns the label Marina. On the other hand, on the axis y is plotted for each image the average (with its standard deviation) of similarity with respect to Marina and Ignacio separately.

On the other hand, two lighter colored lines represent the behavior of an original Marina image when sent to Rekognition. For each individual, the mean similarity values are plotted together with their standard deviation (in a clearer band).

Fig. 55 shows that the manipulated images behave similarly to the original images, providing evidence against the hypothesis. This is not the case for Marina's images that are manipulated O to have them labeled as Ignacio.

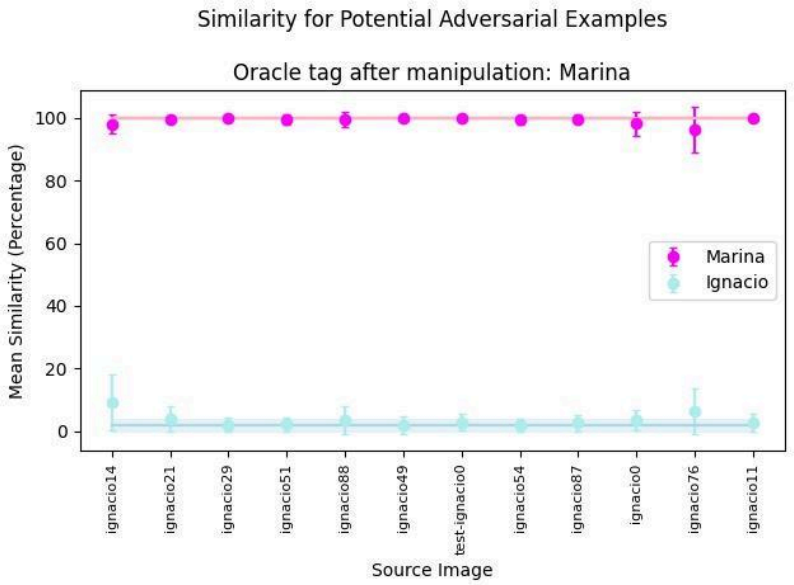


***Figure 55**: Mean similarity for manipulated Ignacio images.*

Fig. 56 shows a figure similar to the one explained above, with some differences. The source images are Marina images and its first major component has been manipulated in such a way that O assigns the Ignacio label. On the other hand, the straight lines corresponding to the mean and standard deviation of similarity refer to the expected behavior of an original Ignacio image.

In this case, there is evidence to support the hypothesis, since the behavior of potential adversaries differs markedly from that of the original images. Of particular interest is the case of the manipulation of marina33, which can be found in **Fig. 57**, since the similarity values for both individuals are practically overlapped.

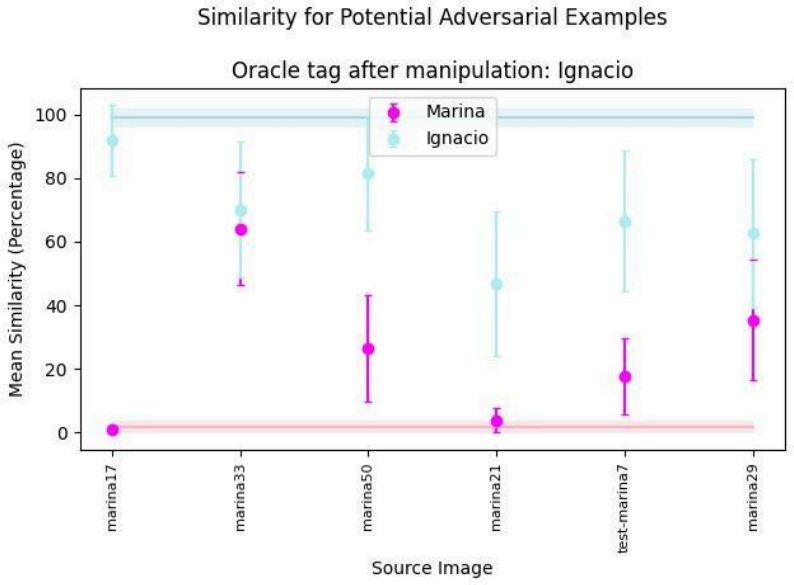


Figure 56: average similarity for manipulated Marina images.

***Figura 57**: Result of the handling of marina33.*

When generating the 4600 images of item 4, an unexpected phenomenon was observed: not all manipulated images resulted in a change of label. It was decided to study this situation separately and to add a sixth step to the procedure, since the existence of these images calls into question the findings of experiment 3.

In the same way as in item 5, 8 images were selected from the 4600 whose original label did not change (despite having changed the value of the first component) and whose quality seems to be maintained. In this step, apply the same restriction on quality analysis explained above.

Each of the images was sent to Rekognition to obtain the metrics obtained for the images of item 5 of the procedure. **Table 9** shows the quality analysis of these images and **Table 10** shows the similarity analysis.

Images with same Oracle's Tag, Quality Analysis			
Source Image	Confidence	Quality Brightness	Quality Sharpness
marina2	100	88,01103210	46,02980042
marina3	100	83,47825623	38,89601135
marina12	100	91,77685547	53,33004761
marina13	100	85,69889069	53,33004761
marina18	100	82,27753448	53,33004761
marina20	99,99811554	85,59648132	53,33004761
test-marina6	100	75,19393158	60,49041748

***Table 9:** Rekognition quality analysis on modified images who kept the label assigned by the oracle.*

Images with same Oracle's Tag, Similarity Analysis				
Source Image	Mean Marina Similarity	Stdev Marina Similarity	Mean Ignacio Similarity	Stdev Ignacio Similarity
marina2	92,61780532	10,9613069	2,96117127	2,91782896
marina3	83,49153317	16,35864252	5,24387771	5,4755931
marina12	97,38220371	5,39847509	8,23557135	7,09698961
marina13	99,14090499	2,37094935	15,19867614	13,65096206
marina18	96,13959216	6,84118404	0,27803256	0,23591804
marina20	68,91432025	22,96477827	3,34106165	5,63893883
test-marina6	99,70805014	1,21200391	6,47128051	8,14745347

Tabla 10: *Rekognition similarity analysis on modified images who kept the label assigned by the oracle.*

Again, a graph was made to study the behavior of these modified images. In **Fig. 58** it can be seen that, although the behaviour did not change as drastically as in **Fig. 56**, it also does not behave in the same way as the original Marina images.

It is interesting to observe the cases of marina20 and marina3, since the similarity with Marina decreases, while with Ignacio it remains within the original values. Although it is possible to notice some noise in the image, the quality results of **Table 9** do not show a significant decrease in the values of *Quality Brightness* and *Quality Sharpness* with respect to the mean of the unmanipulated images, presented in **Table 6**.

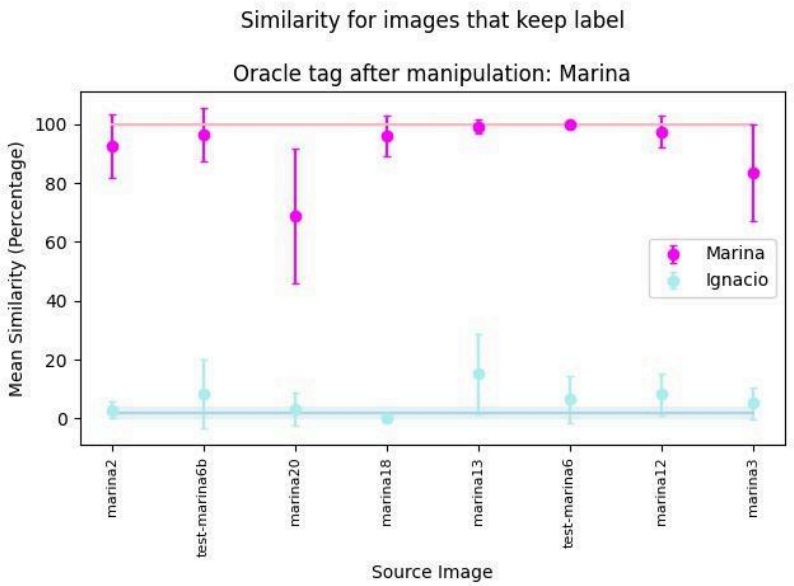


Figura 58*: Average similarity for manipulated Marina images, which retain the label* O*.*

Figura 59*: Result of the manipulation of marina12 and marina20, left and right respectively..*

In conclusion, the existence of these examples shows that the first main component does not allow you to control the label assigned by O in all cases. It seems that there are more factors at play within what is identity, apart from the first component.

On the other hand, to alter the averages of similarity obtained by Rekognition with this methodology, the manipulation need not necessarily include a change of label by O.

8.6.5. Additional information

By modifying the value of the first major component corresponding to the Marina image33 a similarity of 69% was achieved for Ignacio and 64% for Marina, approximately. Although the resulting image is an adversarial example, since sending it to Rekognition meets the definition of evasion attack, it was decided to investigate the environment of this image to look for an adversarial example that would not only produce evasion attacks but also impersonation attacks.

With regard to the first main component, it was decided to maintain an environment close to the value with which the Marina image was manipulated33 during the experiment. On the other hand, components two and three were modified. The aim was to analyse whether, using other components apart from the first, Rekognition could be made to tilt the scales even further towards Marina's recognition, without affecting the label assigned to the image (Ignacio). The procedure is as follows:

1. We get the projections on major components of Marina33.
2. Starting from the original projection vector, the following manipulations are performed:
    a. The first component is modified according to the range [-1. 7, -1. 4] with a step of 0. 1. During the experiment the modification value was -1. 6.
    b. The second component is modified according to the range [-6, 8] with a step of 0. 5.
    c. The third component is modified according to the range [-6, 6] with a step of 0. 5.
3. Of the 2688 resulting images, 36 images[11] are selected to perform a quality and similarity analysis, as in item 5 of the procedure.

[11] A manual selection was made, due to the same restrictions that were mentioned in the previous footer.

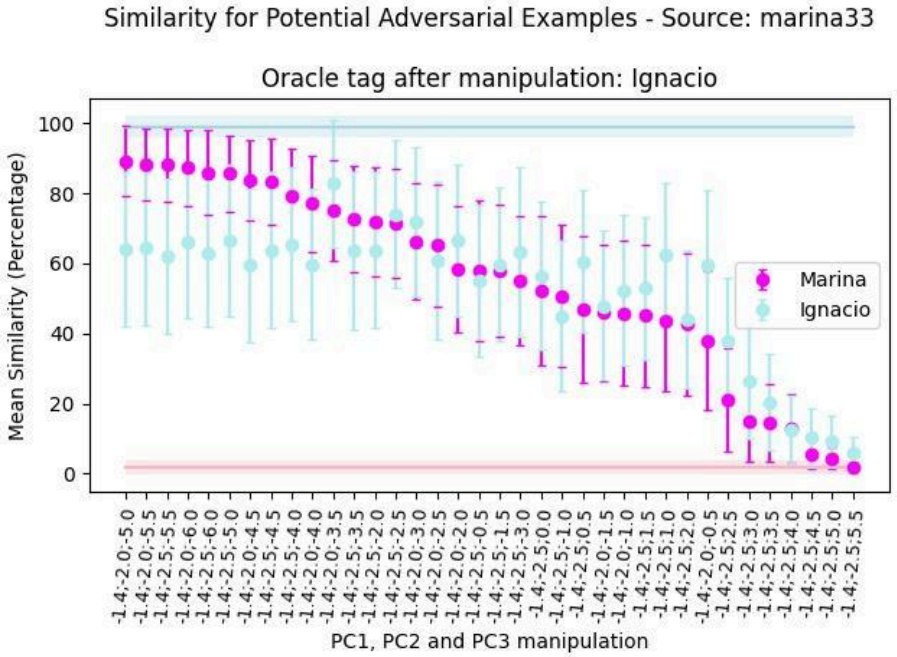


***Figure 60**:* Average similarity for manipulated Marina images.

Fig. 60 shows the similarity results of the new manipulations, ordered in descending order by the mean similarity with respect to Marina. It seeks to analyze the behavior of the same ones in reference to the values of an original image of Ignacio, when sending it to Rekognition. It can be observed that several images have exceeded the 80% threshold for Marina recognition, so that, if the quality is within the expected parameters, such images would allow impersonation attacks to be carried out.

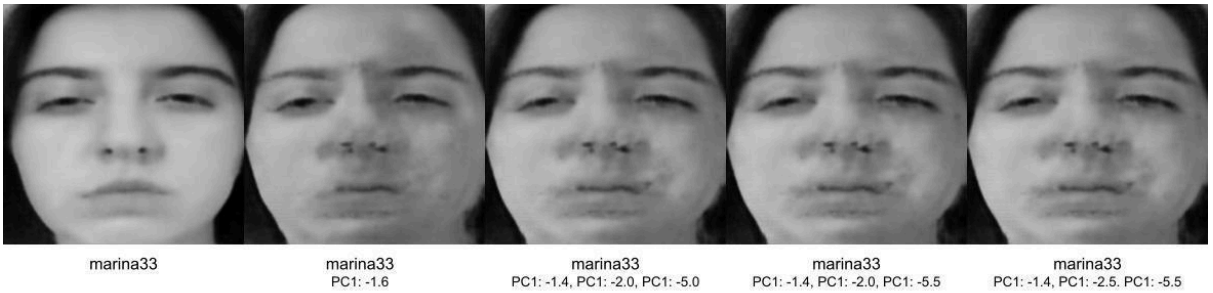


***Figure 61**: manipulations of marina33 in relation to the original image.*

Potential Adversarial Examples, Quality Analysis					
PC1	PC2	PC3	Confidence	Quality Brightness	Quality Sharpness
-1,4	-2,0	-5,0	99,99998474	85,28433228	46,02980042
-1,4	-2,0	-5,5	99,99998474	84,90164948	53,33004761
-1,4	-2,5	-5,5	99,99998474	84,76454926	53,33004761

***Table 11:** Rekognition quality analysis on manipulations from the Marina image33.*

Potential Adversarial Examples, Similarity Analysis						
PC1	PC2	PC3	Mean Marina Similarity	Stdev Marina Similarity	Mean Ignacio Similarity	Stdev Ignacio Similarity
-1,4	-2,0	-5,0	89,14839300	9,908472096	64,15828444	22,19039942
-1,4	-2,0	-5,5	88,14728263	10,24077961	64,30149917	21,93384265
-1,4	-2,5	-5,5	88,06548047	10,51835873	62,03947989	22,24786561

Table 12: *Rekognition similarity analysis on manipulations from the Marina image33.*

Table 11 shows the quality characteristics of the three manipulated images that achieved greater recognition for Marina. There we can see that both *Quality Brightness* and *Quality Sharpness* are within the parameters of average quality, obtained in **Table 6**.

Although the manipulated images are similar to the human eye with respect to the potential adversarial example of the experiment 6, as can be seen in **Fig. 61**, movements in the environment of the experiment allowed to significantly increase recognition for Marina, exceeding the threshold defined for Rekognition and allowing to obtain adversarial examples that provoke evasion and impersonation attacks.

8.7. Experiment 7

8.7.1. Hypothesis

The previous experiment questions the hypothesis raised in experiment 3 about the relationship between identity and the first major component. This leads to reflections on the proposed methodology for generating adversarial examples. Previously, the aim was to define a methodology using the first main component as a basis, with the possibility of adding minor modifications. Once the hypothesis of experiment 3 was questioned, the strategy was proposed to find adversarial examples through an exhaustive search of the space of projections in main components. As a result, the following hypothesis arises.

It is possible to find adversarial examples through the manipulation of main components by brute force. Modifying a main component, not necessarily the first, is enough to generate these examples.

8.7.2. Procedure

1. Major component projections of all images in the dataset are obtained [12].
2. 2 images are selected at random, one with the tag “Marina” and the other with the tag “Ignacio”. Featured pictures: **ignacio74** and **marina19**
3. Calculate the projections on main components of the selected images in 2.
4. For each of the vectors generated in the previous step, the following procedure is performed for each component i[13]:
    a. The minimum and maximum that i takes for the projections obtained in 1. Then the range is divided into eight values.
    b. For each, all the principal components are kept fixed, except for the i-th component, which takes the value in question.

[12] The generated dataset can be found in the following folder.
[13] Limited to the first 20 major components due to AWS *Free Tier* restrictions.

5. The images generated from **ignacio74** are sent to Rekognition and the similarity results are saved in a `.csv`.
6. Item 5 is repeated with the images generated from **marina19**.
7. Searches for impersonation or evasion attacks in the files of 5 and 6, manually observing the similarities. Once the opposing examples are selected, the quality of the same is verified. If the parameters are within the expected range according to **Table 6**, it is taken into account.
8. Of the possible adversarial examples found in 7, 2 of each label are selected.

8.7.3. Analysis

The results of the quality analysis can be seen in **Table 13** and the corresponding similarity analysis in **Table 14** for the adversarial examples obtained in item 8. Finally, **Fig. 62** illustrates the respective opposing images.

Source Image	Confidence	Quality Brightness	Quality Sharpness
PC17: -2.0368	99,99999893	93,24709320	46,02980042
PC15: 4.0341	99,99999893	91,68709564	53,33004761
PC8: -2.968	99,99999893	97,41493225	46,02980042
PC3: 6.9605	99,99999893	96,46032715	53,33004761

Table 13: Quality analysis of possible adversarial examples

Source Image	Mean Marina Similarity	Stdev Marina Similarity	Mean Ignacio Similarity	Stdev Ignacio Similarity	Oracle Result	Attack Type
PC17: -2.0368	0,75797176	0,78659789	72,94214247	24,83002875	Ignacio	Dodging
PC15: 4.0341	0,22895857	0,19176988	60,44409301	26,26751010		Dodging
PC8: -2.968	78,95486565	21,02921946	2,15884985	2,19278473	Marina	Dodging
PC3: 6.9605	42,57506853	20,30322823	6,54301130	8,16298712		Dodging

Table 14: Similarity Analysis of Possible Adversarial Examples

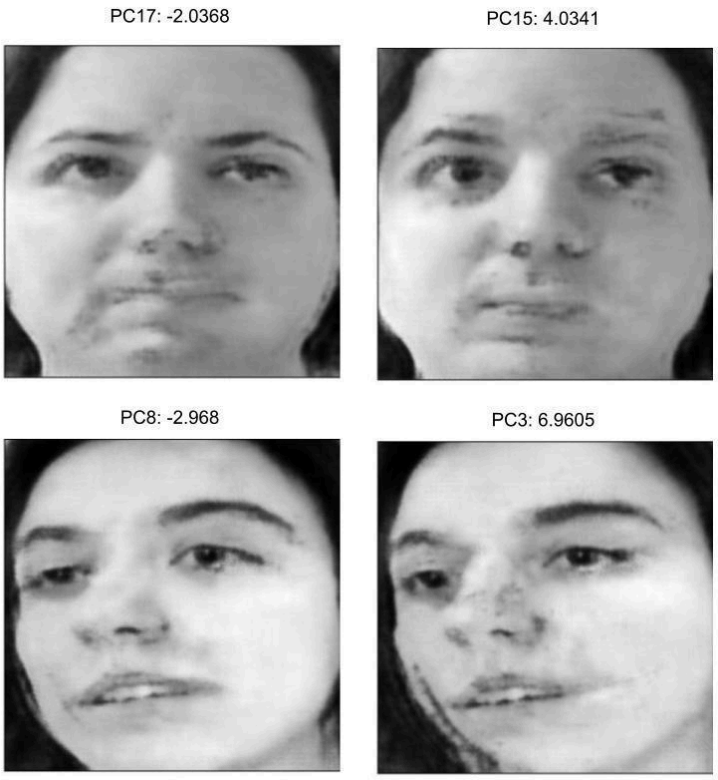


Figure 62: Selected adversarial examples.

In the selected cases, the mean similarity is less than 80, thus meeting the definition of evasion provided in Section 2. 3. However, although it is observed in **Table 13** that the values of *Quality Brightness* and *Quality Sharpness* are within the expected parameters (and even slightly above the mean), it is clear that the manipulation generated a degradation in the image through direct observation in **Fig. 62**

It was also mentioned in step 7 of the procedure that impersonation attacks would be sought, but when reviewing the information generated, no examples were found that would allow such an attack to be carried out. Since an adversarial example could be found in experiment 6 that allows such an attack, it is believed that the modification of a single main component is insufficient to generate such adversarial examples.

9. Discussion

9.1. On decisions made during the investigation

One of the main decisions taken in the work concerns the use of principal component analysis as a way of organizing the latent space to generate adversarial examples. Although the variational *autoencoder* has been successful in generative models, a state-of-the-art *autoencoder* with generative capacity was not thought necessary because the goal was to achieve the construction of adversarial examples from pre-existing images. On the other hand, the analysis of main components provides a hierarchy in the organization, which was of great interest to the authors and, in addition, there was familiarity with the tool. In summary, it was found to be a better alternative.

Another important aspect of the project was the decision on the pattern of attack. Section 3 defined the principle of transferability: there are robust adversaries to different classifiers. This could be an indication that, for the generation of these, it may be advisable to perform a *white box* attack scheme, due to the availability of information from the attacker's perspective, and then to analyze whether this principle is met. This alternative required an accessible facial recognition system. It was decided not to follow this path, opting for a *black box* scheme, which allowed to dedicate more time to the *autoencoder* and the generation of adversarial images. On the other hand, an improvement on the proposed methodology involves adding more than one facial recognition system to validate the results. The choice of Amazon Rekognition as the target system is a consequence of the authors' pre-existing knowledge of the AWS infrastructure.

Additionally, it would be advisable to use specialized *hardware* since it allows to expand possibilities regarding what kind of architectures and with what set of data it is possible to experiment. In this project, we sought to keep a small dataset to speed up training despite the cost that this decision entails in terms of variability within the dataset..

With respect to the architecture of the *autoencoder*, it was decided to take a base *autoencoder*, which decreases the possibilities at the time of assembling the architecture, saving time dedicated to the selection of the same. Although building the network from scratch has many advantages on a personal level due to the learning that it entails, it does not seem advisable for academic work whose goal is not to study neural network architectures. On the other hand, if it exists, it is recommended to use a pre-existing network for the time spent analyzing different architectures, even starting from a base *autoencoder*.

Although a methodology was proposed for choosing the architecture of the *autoencoder,* difficulties were encountered in analysing the data since there was an important component of manual decisions that brought subjectivity to the selection. Step by step, the model was improved until arrival at a decision-making process based on the results of Amazon Rekognition, described in section 6.4. However, it is believed the decision not to start from scratch was correct because a lack of objectivity was found in some of the steps. It is reasonable, due to constraints related to the time invested in selecting the architecture of the *autoencoder*, which should not be excessive. As a recommendation, a more orderly selection methodology should be considered.

Choosing which parameters would be modified during architecture analysis was a complex process. A great lesson during this stage is the attention to be paid when studying the influence of different variables. When the distinction was made between those *autoencoders* that needed more time to train, it was decided to combine two parameters that, when studied separately, required an increase of iterations to achieve convergence: a small optimizer learning rate and a larger *batch* size to update the gradient. The problem is that this combination is exactly the opposite of what should have been done, as shown by an analysis of the effect of *batch* size on neural networks [39]. At the time, it did not seem relevant to analyse in depth what effect the two variables could have together and, later, it was discovered that it was not the desired effect.

Moving on to the process of experimentation, an unexplored alternative path was the manipulation of data projections into latent space. When analyzing how the data of each class (Marina and Ignacio) are distributed in these dimensions, “predominant” directions are observed for each individual. For example, in **Fig. 37**, dimension 3 has Marina as the predominant individual, since its data are scattered along this direction and Ignacio's data are compressed to the left. Investigating the autovector corresponding to the first eigenvalue, it was found that the highest coefficients (in modulus) referred to these dimensions and that the sign of the coefficient corresponded to the predominant individual. An alternative strategy could be based on the use of these characteristic addresses, rather than focusing on the manipulation of the first major component.

The preceding paragraph mentions the importance of using the first major component during experimentation. However, it is important not to lose sight of the fact that the ratio of variance explained is not significantly higher than the rest, especially with respect to the second component, as shown in **Table 3**. Perhaps the weight given to the first component as the only one showing a complete separation of classes should have been more cautious.

Finally, it is considered that one of the most important and necessary changes concerns quality analysis. The aim, and one of the hallmarks of the project, was to obtain adversarial examples of similar quality to that of unmanipulated images. Although the presented cases have quality values similar to the images of the original dataset, using the human eye it is possible to notice that these examples are the result of some form of manipulation. It would be necessary to study alternatives to the *Quality Sharpness* and *Quality Brightness* metrics provided by Amazon Rekognition, since the lack of an objective study of the quality of the adversarial examples is one of the central criticisms made to other proposals for generating the same ones.

9.2. On the relevance of the work in the context of the problem

When reflecting on the possible applications of work, the question arises about the risks of adversarial examples in everyday life situations. Within the mentioned bibliography, the only method of attacking facial recognition systems that seems reasonable is presented in an article whose objective is the feasibility of transferring the adversarial examples to real life [29]: patches or lenses with different patterns are printed on the face causing errors in recognition, and it remains to be studied how this technique performs on video. In the case of the methodology proposed in **4.2**., the attack is restricted to very specific conditions, such as the characteristics of the photo, and would require non-trivial efforts to be applied in the real world.

It is questioned whether the problem of adversarial examples requires the attention it receives or whether, on the contrary, it should be a concern in specific situations. For example, a case of using adversarial examples is the manipulation of content to avoid restrictions of ad-blocking software. On the other hand, neural networks do not yet generalize with the same precision those data with low probability of occurrence. In self-driving cars, a wind-bent sign is still a valid sign because it is not an adversarial example, but an atypical case of real life. So, shouldn't there be more emphasis on other neural network issues?

At present, there is no consensus on the reason for the existence of adversarial examples. A paper presented in 2019 analyzes the characteristics of the data, classifying them into robust and non-robust [30]. Non-robust are a possible cause of adversarial examples but they are of great importance for classifiers because, by removing these characteristics, the accuracy of the same goes down. In general, non-robust characteristics represent properties of little (or nothing) useful for classification by a human actor. Based on what has been discussed, there seems to be a *trade-off* between precision and security in classification networks. Perhaps it would be beneficial to look to other disciplines in search of complementary tools to solve the problem of adversarial examples.

The proposed work, although its main objective was to expand knowledge on adversarial examples, ended as a study of the subject. From the academic point of view, it is considered that there was a lack of perspective: alternative paths were taken and unproductive aspects deepened in order to understand, at an early stage, that the proposed methodology did not support the proposed hypothesis.

10. Conclusion

Although a methodical process to carry out a realistic attack, this being understood as a feasible scenario in real life, was not found, the methodology of experimentation (hypothesis, procedure and subsequent analysis) was very useful to explore the theme and it is recommended to reuse it for the study of other paths.

Regarding the use of principal component analysis as a way of organizing the latent space, despite finding specific examples of evasion and impersonation attacks, no conclusive evidence was found about its usefulness for the proposed objectives. Other techniques, such as Kohonen nets or a variational *autoencoder*, could be used for better results.

Another issue to consider is how to go through the directions in major components. During this project, it was limited to a constant way of movement, that is, with a fixed step. Perhaps other strategies for tracing are more useful for the goals set. In this case, research on displacement in $\mathbb{R}^n$ should be carried out as alternatives are unknown by the authors.

On the other hand, the constraints defined must be taken into account when delimiting the boundaries of the problem. Experimentation could yield very different results if we had a different set of data, such as CelebA (*CelebFaces Attributes*). [40] In addition, training a neural network with specialized hardware would make it possible to obtain a more realistic image generator.

In order to emphasize one of the items discussed in section **9.1**, it is interesting to highlight the importance of performing a quality analysis regardless of the path chosen to organize the latent space, circumnavigate it, etc. The study conducted must be superior to that presented in this report, as the metrics showed that poor resolution images (according to the human eye) had quality values similar to unmodified images.

As a final assessment, from the perspective of novice researchers, it is considered important to carry out a deep analysis of the certainties that exist about the field of study and it is

believed that this applies not only to the theme of adversarial examples but also to those fields where information abounds but a structured scheme to incorporate knowledge is lacking. Throughout the work, it was challenging to decide when to set limits on the objectives, as well as to discriminate the validity of the analyzed contents.

Bibliography

[1] Y. Song, R. Shu, N. Kushman and S. Ermon. Constructing Unrestricted Adversarial Examples with Generative Models. *arXiv:1805.07894 [cs.LG], 2018*.

[2] A. Rosenbrock. Autoencoders with Keras, TensorFlow and Deep Learning. Retrieved from https://www.pyimagesearch.com/2020/02/17/autoencoders-with-keras-tensorflow-and-deep-learning/

[3] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs.LG], 2017.*

[4] S. Park. A 2021 Guide to improving CNNs-Optimizers: Adam vs SGD. Retrieved from https://medium.com/geekculture/a-2021-guide-to-improving-cnns-optimizers-adam-vs-sgd-495848ac6008

[5] C. Versloot. Leaky ReLU: improving traditional ReLU. Retrieved from https://www.machinecurve.com/index.php/2019/10/15/leaky-relu-improving-traditional-relu/

[6] A. Mehra. Facial Recognition Market worth $8.5 billion by 2025. Retrieved from https://www.marketsandmarkets.com/PressReleases/facial-recognition.asp

[7] Facial Recognition: top 7 trends (tech, vendors, markets, use cases & lateset news). Retrieved from https://www.thalesgroup.com/en/markets/digital-identity-and-security/government/biometrics/facial-recognition

[8] N. Carlini. A Complete List of All (arXiv) Adversarial Example Papers. Retrieved from https://nicholas.carlini.com/writing/2019/all-adversarial-example-papers.html

[9] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik and A. Swami. Practical Black-Box Attacks against Machine Learning. *arXiv:1602.02697 [cs.CR], 2017*.

[10] Adversarial Image Translation: Unrestricted Adversarial Examples in Face Recognition Systems. K. Kakizaki and K. Yoshida. Retrieved from http://ceur-ws.org/Vol-2560/paper4.pdf

[11] M.P. Deisenroth, A. A. Faisa and C.S. Ong, “Dimensionality Reduction with Principal Component Analysis” in *Mathematics For Machine Learning* 1st Ed, Cambridge, Reino Unido: Cambridge University Press, cap. 10

[12] I. Jolliffe and J. Cadima Principal component analysis: a review and recent developments. Retrieved from https://royalsocietypublishing.org/doi/10.1098/rsta.2015.0202

[13] M. Stewart. A Comprehensive Introduction to Autoencoders. Retrieved from https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368

[14] T. Hastie, R. Tibshirani, J. Friedman “Principal Components, Curves and Surfaces” in *The Elements of Statistical Learning: Data mining, Inference, and Prediction* 2nd Ed, Springer.

[15] I. J. Goodfellow, J. Shlens and C. Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv:1412.6572 [stat.ML], 2015*.

[16] I. Jolliffe and J. Cadima. Principal Component Analysis: a review and recent developments. Retrieved from https://royalsocietypublishing.org/doi/10.1098/rsta.2015.0202

[17] MTCNN Documentation. Retrieved from https://pypi.org/project/mtcnn/

[18] AWS SDK for Python (Boto3). Retrieved from https://aws.amazon.com/sdk-for-python/

[19] Amazon Rekognition Documentation for its API (used with Boto3). Retrieved from https://docs.aws.amazon.com/rekognition/latest/dg/API_Reference.html

[20] Matplotlib Documentation. Retrieved from https://matplotlib.org/stable/index.html

[21] W. Clumper. How Accurate are Facial Recognition Systems. Retrieved from https://www.csis.org/blogs/technology-policy-blog/how-accurate-are-facial-recognition-systems-%E2%80%93-and-why-does-it-matter
[22] DeepAI Machine Learning Glossary. What is the manifold hypothesis? Retrieved from https://deepai.org/machine-learning-glossary-and-terms/manifold-hypothesis
[23] Scikit Learn. Python Principal Components Analysis Documentation. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
[24] Y. Zhang, X. Tian, Y. Li, X. Wang and D. Tao. Principal Component Adversarial Example. Retrieved from http://staff.ustc.edu.cn/~xinmei/publications_pdf/2020/09018372.pdf
[25] D. Hendrycks and K. Gimpel. Early Method for Detecting Adversarial Examples. *arXiv:1608.00530 [cs.LG], 2017*.
[26] N. Carlini and D. Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. Retrieved from https://dl.acm.org/doi/abs/10.1145/3128572.3140444
[27] N. Carlini, D. Wagner, F. Tramèr. Adversarial Examples, Machine Learning Street Talk. Retrieved from https://www.youtube.com/watch?v=2PenK06tvE4
[28] C. Szegedy et al. Intriguing Properties of Neural Networks. *arXiv:1312.6199 [cs.CV], 2014*.
[29] M. Pautov, G. Melnikov, E. Kaziakhmedov, K. Kireev and A. Petiushko. On adversarial patches: real-world attack on ArcFace-100 face recognition system. *arXiv:1910.07067 [cs.CV], 2020*.
[30] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran and A. Madry. Adversarial Examples are not Bugs, they are Features. *arXiv:1905.02175 [stat.ML], 2019*.
[31] U. Hwang, J. Park, H. Jang, S. Yoon, N. Ik Cho. PuVAE: A Variational Autoencoder to Purify Adversarial Examples. *arXiv:1903.00585 [cs.LG]*.
[32] B. Wójcik, P. Morawiecki, M. Śmieja, T. Krzyżek, P. Spurek, J. Tabor. Adversarial Examples Detection and Analysis with Layer-wise Autoencoders. *arXiv:2006.10013 [cs.LG]*.
[33] F. Tramèr, N. Carlini, W. Brendel, A. Madry. On Adaptative Attacks to Adversarial Example Defenses. *arXiv:2002.08347v2 [cs.LG]*.
[34] R. Bird. The Post-Pandemic Workforce Requires Greater Identity Security To Achieve The New Normal. Retrieved from https://www.forbes.com/sites/forbestechcouncil/2021/08/11/the-post-pandemic-workforce-requires-greater-identity-security-to-achieve-the-new-normal/?sh=42364cba370d
[35] A. Wang. The importance of secure remote authentication during lockdown (and beyond) Retrieved from https://dis-blog.thalesgroup.com/corporate/2020/06/19/the-importance-of-secure-remote-authentication-during-lockdown-and-beyond/
[36] McKinsey Global Institute. The future of work after COVID-19. Retrieved from https://www.mckinsey.com/featured-insights/future-of-work/the-future-of-work-after-covid-19
[37] M. Dubie. Another Layer of Security for Your Remote Workforce, On Us Retrieved from https://blog.lastpass.com/2020/05/another-layer-of-security-for-your-remote-workforce-on-us/
[38] N. Carlini and C. Wagner. Towards Evaluating the Robustness of Neural Network. *arXiv:1608.04644v2 [cs.CR]*.
[39] D. Chang and A. Pathak. Effect of Batch Size on Neural Net Training. Retrieved from https://medium.com/deep-learning-experiments/effect-of-batch-size-on-neural-net-training-c5ae8516e57
[40] CelebA Dataset. Retrieved from https://www.kaggle.com/jessicali9530/celeba-dataset

Appendix

Arquitectura del autoencoder base

```
Model: "encoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 14, 14, 32)        320
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 14, 14, 32)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 14, 14, 32)        128
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 7, 7, 64)          18496
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 7, 7, 64)          0
_________________________________________________________________
batch_normalization_1 (Batch (None, 7, 7, 64)          256
_________________________________________________________________
flatten (Flatten)            (None, 3136)              0
_________________________________________________________________
dense (Dense)                (None, 16)                50192
=================================================================
Total params: 69,392
Trainable params: 69,200
Non-trainable params: 192
```

```
Model: "decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 16)]              0
_________________________________________________________________
dense_1 (Dense)              (None, 3136)              53312
_________________________________________________________________
reshape (Reshape)            (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 14, 14, 64)        36928
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 14, 14, 64)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 14, 14, 64)        256
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 28, 28, 32)        18464
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 28, 28, 32)        0
_________________________________________________________________
batch_normalization_3 (Batch (None, 28, 28, 32)        128
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 28, 28, 1)         289
_________________________________________________________________
```

```
activation (Activation)      (None, 28, 28, 1)         0
=================================================================
Total params: 109,377
Trainable params: 109,185
Non-trainable params: 192
```

ABC initial parameters

Parámetro	Valor
Dimensiones	
Alto	256
Ancho	256
Canales	1
Dimensión capa latente	64
Capas Convolucionales	
Padding	*Same*
Strides	2
Filtros Capa Convolucional 1	16
Filtros Capa Convolucional 2	32
Filtros Capa Convolucional 3	64
Núcleo Capa Convolucional 1	3
Núcleo Capa Convolucional 2	5
Núcleo Capa Convolucional 3	7
Funciones de Activación	
Activación para codificador	Leaky ReLu
Valor alpha para codificador	0,2
Activación para decodificador	Leaky ReLu
Valor alpha para decodificador	0,2
Entrenamiento	
Optimizador	Adam
Tasa de aprendizaje	1,00E-03
Beta 1	0,9
Beta 2	0,9999
Epsilon	1,00E-07
amsgrad	Falso

Reconstruction for architectural selection

Optimiser learning rate

Tasa de aprendizaje 10^{-1} | Tasa de aprendizaje 10^{-2} | Tasa de aprendizaje 10^{-3} | Tasa de aprendizaje 10^{-4} | Tasa de aprendizaje 10^{-5}

Number of convolutional layers

3 capas convolucionales | 4 capas convolucionales | 5 capas convolucionales

Amount of data for gradient update

Batch Size 4 | Batch Size 8 | Batch Size 12 | Batch Size 32

Latent Layer Dimension

Dimensión capa latente 3 | Dimensión capa latente 9 | Dimensión capa latente 16

Dimensión capa latente 32 Dimensión capa latente 64 Dimensión capa latente 128

Activation function

Leaky ReLU Alpha 0,01 | Leaky ReLU Alpha 0,05 | Leaky ReLU Alpha 0,1 | Leaky ReLU Alpha 0,3

Leaky ReLU Alpha 0,6 | Leaky ReLU Alpha 0,9 | Leaky ReLU Alpha 1,3

Data obtained with Rekognition for autoencoder selection

The values presented in the table have been rounded to 4 decimal places after the comma.

Dataset	Mean Confidence	SD Confidence	Mean Similarity	SD Similarity	Mean Brightness	SD Brightness	Mean Sharpness	SD Sharpness
ID_1_0	100,0000	0,0003	98,3428	10,0982	82,4791	6,0569	37,7729	8,6112
ID_1_1	100,0000	0,0002	99,0623	7,1970	82,4895	6,1173	38,6177	9,1183
ID_1_2	99,9997	0,0044	97,6836	12,1530	82,5429	5,9556	38,1909	8,9645
ID_1_3	99,9998	0,0036	97,6715	12,8152	82,3019	5,8837	38,3637	9,2180
ID_1_4	99,9998	0,0033	97,7735	12,4828	82,1778	6,0637	38,5492	9,4988
ID_2_0	99,9998	0,0030	97,8291	11,8882	82,2344	6,0022	39,7759	10,0466
ID_2_1	99,9998	0,0038	97,7302	12,2530	82,2973	5,9347	40,7180	10,1970
ID_2_2	99,9998	0,0035	97,7605	12,2355	82,3485	6,0196	41,2278	10,2701
ID_2_3	99,9998	0,0033	97,8329	12,0351	82,3689	6,0049	41,9026	10,4844
ID_2_4	99,9976	0,0709	97,7380	12,4298	82,3611	5,9931	42,7289	11,0332
ID_3_0	99,9978	0,0672	97,7560	12,3442	82,3425	6,0406	41,4821	11,4005
ID_3_1	99,9954	0,1403	97,7328	12,4784	82,3171	6,0302	40,6911	11,4524
ID_3_2	99,9957	0,1344	97,6895	12,6126	82,3890	6,0788	39,9569	11,4458
ID_3_3	99,9961	0,1291	97,5304	13,1587	82,4201	6,1352	39,3538	11,4106
ID_3_4	99,9958	0,1271	97,7629	12,6137	82,4201	6,1188	39,7518	11,4493
ID_4_0	99,9999	0,0007	97,4381	12,5033	83,5325	6,6583	31,7963	8,3458
ID_4_1	99,9983	0,0323	96,9212	14,7676	83,3255	6,7770	28,9734	8,5647
ID_4_2	99,9987	0,0266	96,4541	16,0063	83,4483	6,8107	28,9305	8,3102
ID_4_3	99,9990	0,0231	96,5180	15,8799	83,2203	6,8739	28,5142	8,2005
ID_4_4	99,9991	0,0207	96,7389	15,3207	83,0847	6,7891	28,1898	8,1105
ID_5	99,3956	3,7466	88,9110	26,2089	84,8908	6,3000	21,2068	6,1944
ID_6	99,5937	3,1381	89,3062	26,9383	83,8281	6,4139	23,2503	6,6237
ID_7	99,7056	2,5816	89,6576	26,3961	83,7425	6,4889	22,3525	6,4552
ID_8	99,6801	2,5723	90,0398	26,1199	83,5294	6,4926	22,0458	6,2983

Principal Component Analysis

The values in the table correspond to the magnitude of the eigenvalues resulting from applying PCA.

Eigenvalues							
A1	10,802648	A2	6,846757	A3	4,1302238	A4	3,4029918
A5	3,180926	A6	3,052712	A7	2,6865425	A8	2,1077893
A9	2,0627632	A10	1,7716402	A11	1,6505468	A12	1,5524656
A13	1,4944257	A14	1,4052042	A15	1,2848547	A16	1,2703762
A17	1,1192168	A18	1,075609	A19	1,0246155	A20	0,95771646

A21	0,79937375	A22	0,74055624	A23	0,7181599	A24	0,7058683
A25	0,6630153	A26	0,60759145	A27	0,5369445	A28	0,48363867
A29	0,46602702	A30	0,43243918	A31	0,4239982	A32	0,4158757
A33	0,36800623	A34	0,35179108	A35	0,31014958	A36	0,30215493
A37	0,29062107	A38	0,2531139	A39	0,24391817	A40	0,22814539
A41	0,19129087	A42	0,18587112	A43	0,17680734	A44	0,17156887
A45	0,15480174	A46	0,13777995	A47	0,12790592	A48	0,11446608
A49	0,10293874	A50	0,08801629	A51	0,08473146	A52	0,07610799
A53	0,071360394	A54	0,065281175	A55	0,0570183	A56	0,052949436
A57	0,049825	A58	0,03970099	A59	0,034882165	A60	0,033514645
A61	0,027005587	A62	0,024766479	A63	0,023369921	A64	0,008213418

Detail of Modifications for Experiment 1

The values in the table correspond to the modifications of the first and second main components made on the images of Marina and Ignacio during experiment 1. The images corresponding to the manipulations are found in **Fig. 33** and **Fig. 34** respectively.

Potential Adversarial Examples, Modifications				
Source Image	PC1 Old Value	PC1 New Value	PC2 Old Value	PC2 New Value
ignacio average	-3,4602	0,5000	-0,3558	-3,5000
ignacio average	-3,4602	2,0000	-0,3558	-0,5000
ignacio average	-3,4602	4,5000	-0,3558	1,0000
ignacio average	-3,4602	5,5000	-0,3558	-3,0000
marina average	3,5583	-1,5000	-1,1882	-3,0000
marina average	3,5583	-3,0000	-1,1882	-1,5000
marina average	3,5583	-3,5000	-1,1882	-3,5000
marina average	3,5583	-4,0000	-1,1882	-4,0000

Detail of Modifications for Experiment 6

The values in the table correspond to the modifications of the first main component made on the potential adversarial examples of item 5 of the procedure of experiment 6. Below this table are the images resulting from modifying that component.

Potential Adversarial Examples, Modifications		
Source Image	PC1 Old Value	PC1 New Value
ignacio0	-3,551283	5,00
ignacio11	-3,813199	6,40
ignacio14	-3,199674	4,40
ignacio21	-3,987424	6,40
ignacio29	-3,389351	6,20
ignacio49	-2,824564	5,40

ignacio51	-2,916997	6,40
ignacio54	-3,654767	6,40
ignacio76	-3,037767	6,20
ignacio87	-2,562436	6,00
ignacio88	-3,223754	4,20
test-ignacio0	-3,774843	6,00
marina17	3,251978	-4,20
marina21	3,875732	-1,60
marina29	3,099534	-2,80
marina33	3,993810	-1,60
marina50	2,776832	-4,20
test-marina7	3,143551	-4,00

Below is a second table, corresponding to the modifications of the first main component of those images that maintained the label assigned by the oracle (despite such manipulation). Below this table are the images resulting from the decoding of the manipulated main components.

Images with same Oracle's Tag, Modifications		
Source Image	PC1 Old Value	PC1 New Value
marina2	1,8581791	-1,60
marina3	1,8956704	-1,60
marina12	1,0469455	-1,60
marina13	-0,1853556	-1,80
marina18	0,2062798	-1,60

marina20	-0,2197045	-3,00
test-marina6	-0,2140693	-1,60

Annex A

As the end of the work approached, the tutor proposed a series of questions to reflect on the final project. Below are the questions with the respective answers of the authors.

What problem does it solve? This project does not seek to solve a particular problem but to expand the frontier of knowledge regarding the area of study of adversarial examples. As we mentioned during the state of the art, adversarial examples were first presented in 2014 and even today there is no consensus on their cause. Through the creation of a new methodology for generating adversarial examples, we try to find relevant data for the problem.

What is the state of the art? It was presented in the State of the Art section of the report.

What knowledge of the degree was useful to you? Of the many subjects taken during the degree, those that proved to be most useful are:

- Artificial Intelligence Systems, since both the area of study and the main tools for the attack methodology were presented in this course.
- Software Engineering II, mainly during the proposal of an alternative path analyzed in March 2021: a web application to implement the experimentation environment. It allowed us to diagram functionalities, assumptions, *trade-offs*, etc.
- Machine Learning, because in this area special attention is paid to the correct evaluation of the results obtained when implementing different solutions.
- Cloud Computing, since cloud computing tools were used, including the target of attack: Amazon Rekognition.
- Cryptography and security, as we both had a special interest in choosing a topic relevant to computer security due to the curiosity present during the subject. It was also useful to analyze the feasibility of considering attacks in a real context.
- Algebra, as it allowed us to understand the theory behind principal component analysis.

What is unique about this thesis/FP that you haven't found implemented or researched elsewhere? Primarily, there are two elements of this final project that have not been found during the research, which are:

- The use of an *autoencoder* combined with principal component analysis in its latent space for the generation of adversarial examples.
- The quality analysis of potential adversarial examples with respect to original images. Although in this case it was of great relevance, given that the autoencoder did not correspond to the state of the art, we did not find any mention of quality in any of the papers in the bibliography or of the possibility that it influences as a factor during the recognition or classification of an object.

Why did you choose this issue, and how does it relate to the team? We selected this problem because we both shared an interest in issues related to security and, during the search within this area, the question arose about possible vulnerabilities in facial recognition systems. Over the years of working as a team, it always happened that one of us showed more interest in the theoretical aspect of a subject and the other in the technical one. It

turned out to be an excellent combination to introduce us to the research, which neither of us had previously engaged in.

Is there a product or service that can put the work they did to use? No, the current work has a purely academic purpose and, in addition, we concluded that the methodology was not an ideal research alternative, much less to apply in a real-life scenario.

What would they do if they had more time? We agree that a more in-depth analysis of the state of the art would be carried out, especially delving into hypotheses and research on possible explanations of the adversarial examples. On the other hand, from the technical side, we believe it is important to validate that the results obtained for the individuals in the dataset are replicated when these individuals are changed.

What important decisions did they have to make? The decisions that we consider most relevant were analyzed in the Discussion section and in two subsequent questions:

- What mistakes do you think you've made along the way?
- What successes do you think you have made along the way?

What did they learn in the process? From a technical and methodological point of view, our main learnings were:

- Research Methodology: How to Approach a Research Problem.
- The Operation of *Autoencoders* and Principal Component Analysis.
- How Amazon Rekognition works.

On the other hand, from a process management perspective, our main learnings were:

- If possible, have people who are active in the area of research you are approaching.
- When making estimates, keep in mind the possibility of underestimating the task to be performed. It happened to us with the web application we tried to develop (see **Appendix B**).
- Delimit the boundaries of objectives and tasks as work can be extended indefinitely.
- Clearly outline the short, medium and long term. Then, carry out regular checks as it is easy to lose sight of the main objective by having so many possibilities. It happened that certain aspects of the work, such as the training of the *autoencoder*, took a long time in relation to their contribution to the problem we sought to study.
- Honest and early communication between team members. Talking about the concerns we had regarding the chosen course and the methodology with which we worked was essential to discuss the advantages and disadvantages of the different options available to us.

What is the difference between what they planned to do in the first month and what they ended up doing? We had hoped to find a clearer relationship between the projections of the first principal components and the results obtained with Amazon Rekognition. We also hoped to obtain a generic manipulation methodology, whereas in the end no definite link was found and the experiments turned out to be much more manual.

What mistakes do you think you've made along the way? While several mistakes were discussed in the Discussion section, we would like to highlight three decisions that ultimately harmed development:

- At the beginning of the *autoencoder* selection, we were not diligent in storing all the results obtained. This brought temporary delays in the selection of the same.
- The lack of an objective methodology for the choice of the autoencoder. Many decisions were based on subjective observations.
- Finally, what we consider to be the biggest mistake is the retour to the web application. We thought it was a good way to finalize the final project with a deliverable, but we underestimated the time needed to develop it and overestimated the relevance it had in our work.

What successes do you think you have made along the way? Two successes that we think are important to mention are, firstly, to approach the research without fear that the hypothesis proposed is false. Second, to have returned to the original research after spending two months developing the Dexter app. While it was a difficult decision, we believe it was the right one to bring the work to a satisfactory end.

Why do you think what you did is worthy of approval? An in-depth study was carried out on the research topic and, although the hypotheses raised were not tested, the research and decision-making process was carried out seeking the greatest clarity and objectivity that we could achieve. Testing the hypothesis was not the focus of the project, but rather carrying out an exhaustive and critical research work of the proposal.

What kind of audience would be interested in what they did? The work is aimed at people interested in the subject of deep learning, not only from a practical perspective, but also from an academic and perhaps philosophical perspective. This project is a particular example of a curious phenomenon that occurs with algorithms, which, although under investigation, raises many questions about the suitability of deep learning tools in everyday life.

What profile of students could continue with their work? First of all, we think it is an interesting job for students looking to enter the world of research, since we make, in our opinion, constructive criticisms of the approach to research from the perspective of novices. Since the main subject of the work is artificial intelligence, these students would be expected to be curious about the subject and knowledge of the basic concepts.

Annex B

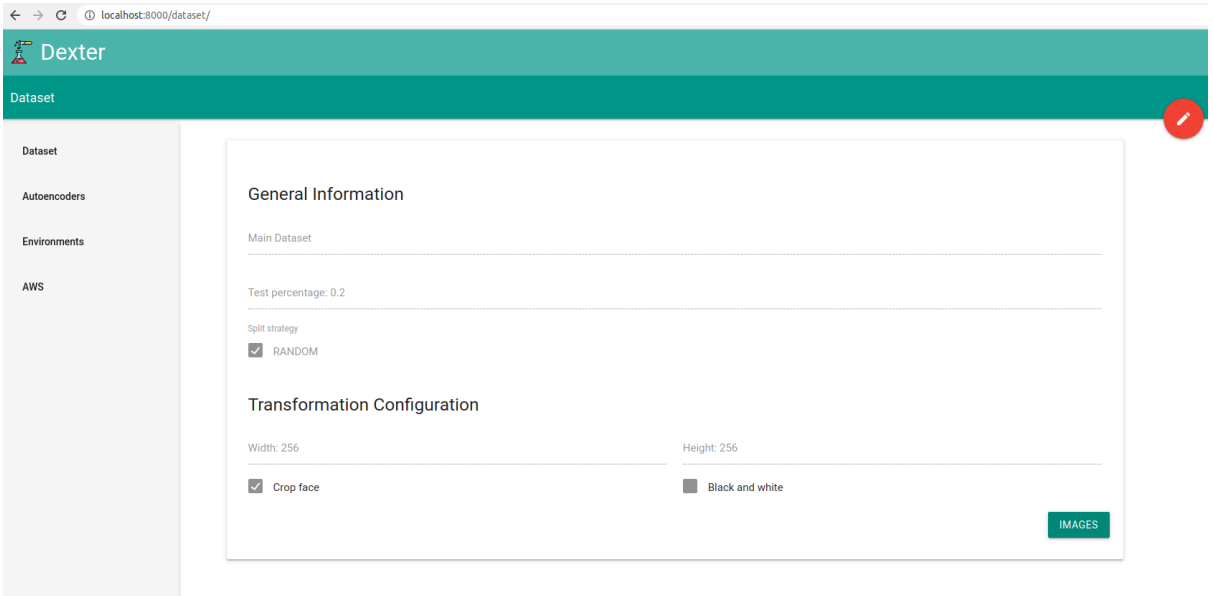
Dexter is the name given to a web application whose development began in March, 2021. The goal was to create a platform that would allow to carry out experiments in a simple way through the development of a GUI, including

- Creation and training of *autoencoders* with different datasets and architectures.
- Creation of different experimentation environments including a dedicated *autoencoder* and facial recognition system.
- Setting up experiments and obtaining results for further analysis.
- Integration with facial recognition systems to test the principle of transferability. Initially it was intended to add Amazon Rekognition.

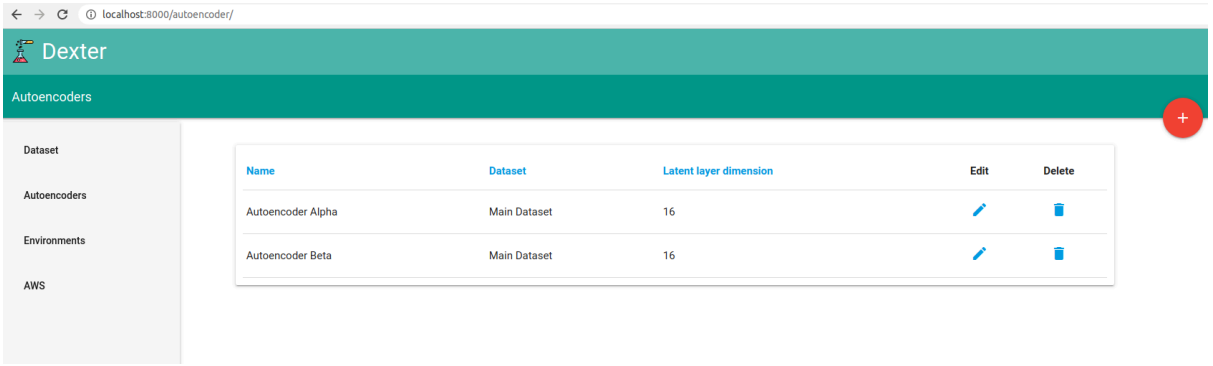
This project was developed for two months but it was during June that it was decided to return to the original path since the implementation was a very ambitious project as a complementary element of the research. Considering that a lot of time had already been devoted to complementary but not central elements of the attack methodology (such as the selection of the *autoencoder*) it was decided to go back to the original path and leave this project on indefinite pause.

Here are some views of the application:

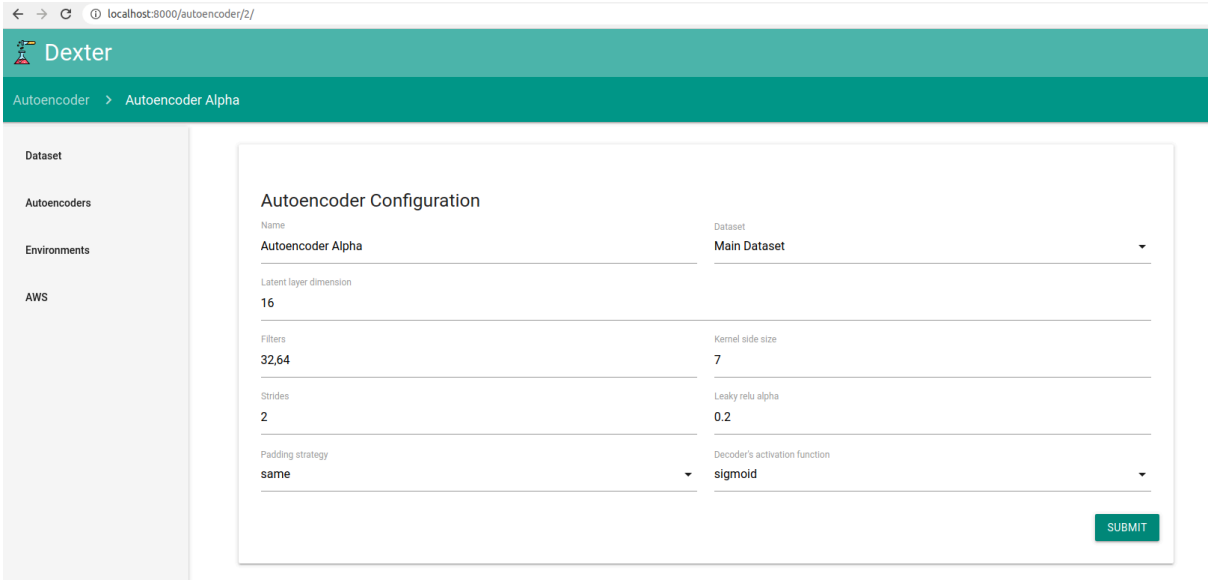
Detailed view of the options for creating or editing the dataset. In the case of the screenshot, the editing of an already created dataset is observed.



Detailed view of the list of *autoencoders* created. In the case of the screenshot you have two *autoencoders*.

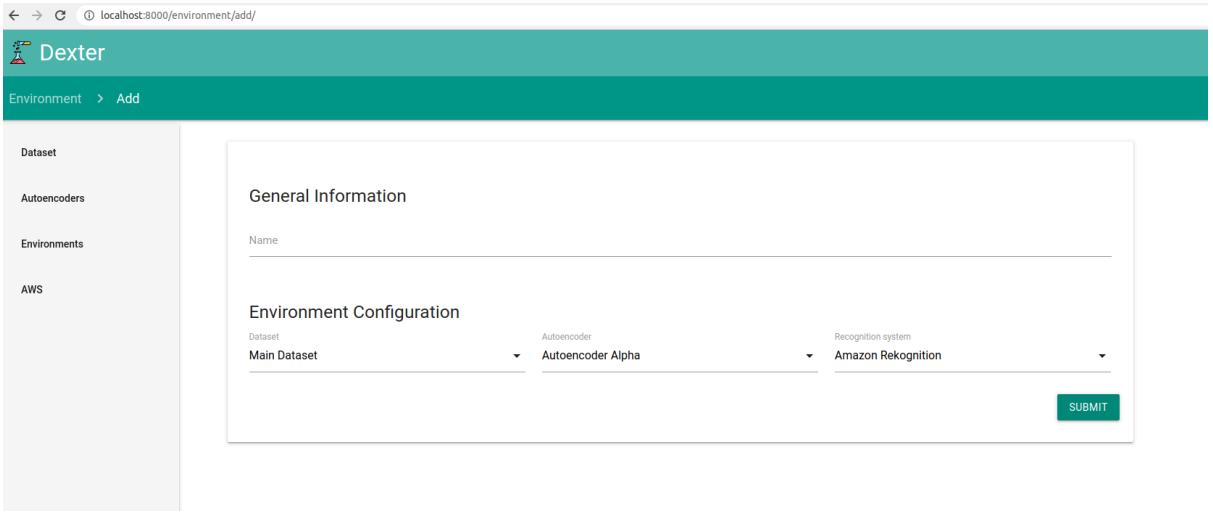


Detailed view of the options for creating or editing an *autoencoder*.



Detailed view of the list of experimental environments.

Detailed view of the options for creating or editing an experimentation environment.



Detailed view of the list of experiments for a particular environment, in this case, Env0.