T.C.

ALTINBAS UNIVERSITY

Institute of Graduate Studies

Information Technologies

DETECTION OF FACIAL FEATURES USINGSTATISTICAL MACHINE LEARNING AND MODIFIED ADABOOST CLASSIFIER

Melak Thamer NASSRULLAH

Master of Science

Supervisor

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM

Istanbul, 2021

DETECTION OF FACIAL FEATURES USING STATISTICAL MACHINE LEARNING AND MODIFIED ADABOOST CLASSIFIER

by

Melak Thamer NASSRULLAH

Information Technologies

Submitted to the Graduate School of Science and Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

ALTINBAŞ UNIVERSITY

2021

The thesis titled "DETECTION OF FACIAL FEATURES USING STATISTICAL MACHINE LEARNING AND MODIFIED ADABOOST CLASSIFIER" prepared and presented by "Melak Thamer NASSRULLAH" was accepted as a Master of Science Thesis in Information Technologies.

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM

Supervisor

Thesis Defense Jury Members:

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM	School of Engineering and Architecture, Altinbas University	_____
Asst. Prof. Dr. Mesut ÇEVIK	School of Engineering and Architecture, Altinbas University	_____
Asst. Prof. Dr. Sewale Musadaq TAHA	Faculty of Communication, Beykent University	_____

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Approval Date of Institute of Graduate studies:

____/____/____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Melak Thamer NASSRULLAH

DEDICATION

I would like to thank Allah Almighty for the power of mind, health, strength, guidance, knowledge and skills to complete this study.

This thesis is wholeheartedly dedicated to my parents. There are no words to describe what you mean to me; there is nothing that I can repay for what you have done to me. I will continue to do my best to achieve your expectations.

Lastly, I dedicated this to my parents, sisters and brother, relatives, and friends who have been encouraging me during this study.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Asst. prof. Dr. Abdullahi Abdu IBRAHIM for all support during my study. It's great pleasure to express my deepest gratitude to my friends who have shared with me best moments during my study for the Master degree.

ABSTRACT

DETECTION OF FACIAL FEATURES USING STATISTICAL MACHINE LEARNING AND MODIFIED ADABOOST CLASSIFIER

NASSRULLAH Melak Thamer

M.Sc., Information Technologies, Altınbaş University,

Supervisor: Asst. Prof. Dr. Abdullahi Abdu IBRAHIM

Date: 08/2021

Pages: 53

Classification modules can be used in multiple and diverse applications. The general objective of this work is the classification of facial features using statistical learning algorithms, this means being able to detect and classify the largest possible number of features that can be found on a face using only examples of images of these, without using a priori no information of the given characteristics. In the present work, detectors and classifiers of the characteristics that are considered most significant were developed, Excellent results were reported in mouth detection, with detection rates greater than 99% and errors comparable to the error from manual mouth marking. The lens sorter also obtained excellent results, with detection rates of 95% for databases with controlled environments and of the order of 90% for databases with uncontrolled environments. Beard and mustache classifiers after using the mouth detector obtained very good results, with a detection rate of over 95% in databases with uncontrolled environments.

Keywords: Face recognition, Statistical learning, Mouth, Detectors, Facial features.

TABLE OF CONTENTS

Pages

LIST OF FIGURES

Pages

1. INTRODUCTION

1.1 BACKGROUND

Today, society lives in the midst of a constant flow of data where privacy plays an important role. In the XXI century, the majority of society has a close link with Social Networks, which we use as a means to share with our environment day to day. But, are we really aware of the impact that this has on our private lives? Many of the data that we share cease to be completely ours from the moment it is uploaded to the network. One of the main axes of this project, apart from its technical preparation, is to raise awareness and awareness of the scope that our data may have, since with a single photo and with the proper data analysis, it is possible to get to extract more information from ourselves than we think. In this case, I will stick to the data extracted from the social network Instagram, to later perform a facial recognition of the user through convolutional neural networks and an extraction of information from the data obtained, such as obtaining trends or hobbies according to the photos of the recognized user. Once again, the end result of this project, like most Deep Learning processes that use personal data, can have a multitude of applications in the real world that we will analyze throughout the project, although many of them have commercial purposes. That is why the main objective of this project is to create a facial recognition system, based on Deep Learning techniques and from data obtained from Instagram. To achieve it, we have divided this objective into two particular objectives: - Construction of a facial recognition system, where we include all the classification and vision techniques in face recognition through neural networks. - Obtaining the database, and all the techniques that we will use to extract information from it. All the information and knowledge about the neural networks that we are going to use has been acquired after completing the Deep Learning.ai course from Coursera, therefore, we will use materials already worked on in said course. [2]:

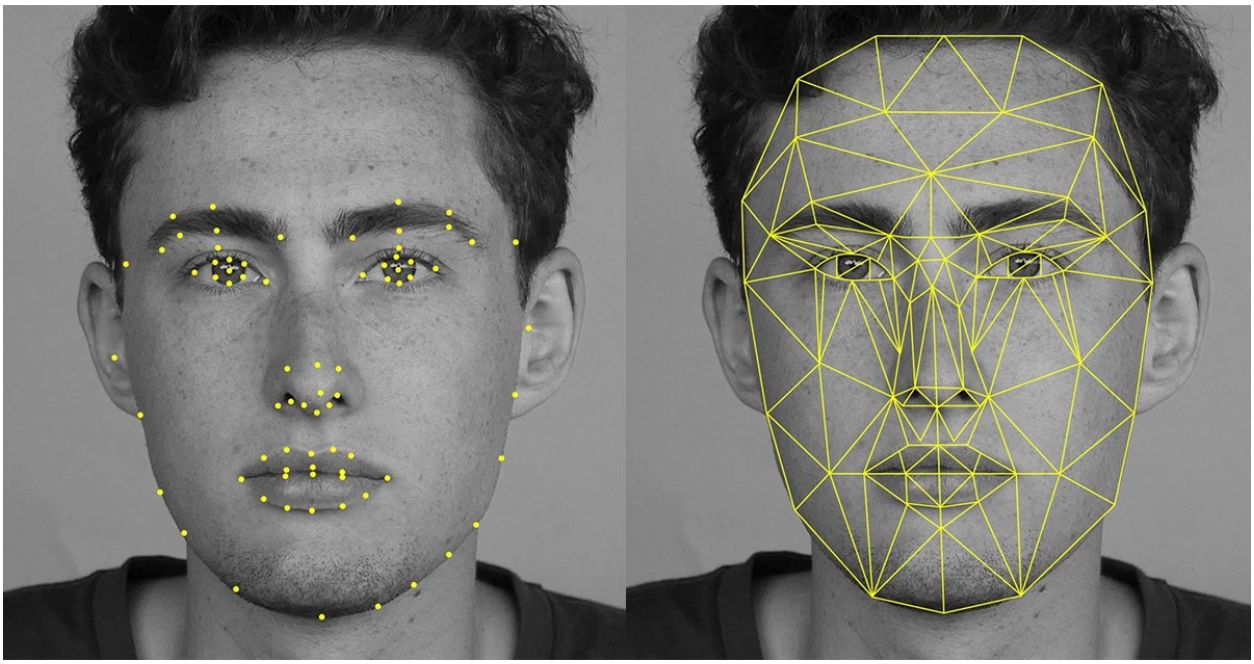

Figure 1.1: Simple face tracking system.

1.2 MOTIVATION

Classification modules can be used in multiple and diverse applications. The general objective of this work is the classification of facial features using statistical learning algorithms, this means being able to detect and classify the largest possible number of features that can be found on a face using only examples of images of these, without using a priori no information of the given characteristics. In the present work, detectors and classifiers of the characteristics that are considered most significant were developed, Excellent results were reported in mouth detection, with detection rates greater than 99% and errors comparable to the error from manual mouth marking. The lens sorter also obtained excellent results, with detection rates of 95% for databases with controlled environments and of the order of 90% for databases with uncontrolled environments. Beard and mustache classifiers after using the mouth detector obtained very good results, with a detection rate of over 95% in databases with uncontrolled environments.

1.3 PROBLEM STATEMENT

In today's world, human-computer interfaces, such as those previously presented, are becoming more common and necessary, and imagining a system that a few years ago was only seen in a Hollywood movie is no longer so difficult. Due to various factors, the use of automatic interfaces, in which the person has to interact quickly and unobstructed, are becoming more and

more common. The main motivation of this memory work is the study of facial characteristics, with the purpose of using them for example in human-computer interfaces. Of course, there are many other possible uses for the study of facial characteristics, such as security, automatic indexing or also as an aid to the always difficult goal of facial recognition. The characteristics that we seek to detect and classify in this memory work correspond to the most common characteristics that we can find in a face, and many times, the characteristics that as humans, we tend to memorize when we see a face for the first time. These characteristics can help to discern between different characteristics of the person, which can then be used for different purposes. There are, in the aforementioned classifiers and detectors, multiple difficulties when it comes to discerning, the most complex being the high degree of variety they present. For example, lenses, there are infinite types and styles of lenses, which suggests that it will be difficult to make an automatic classification of these. Why else mention beards, which today have gone from forming religious or political characteristics to a kind of accessory on men's faces. Although the Royal Spanish Academy defines them as "hair located under the area of the mouth", there are different types and positions of these. [12]:

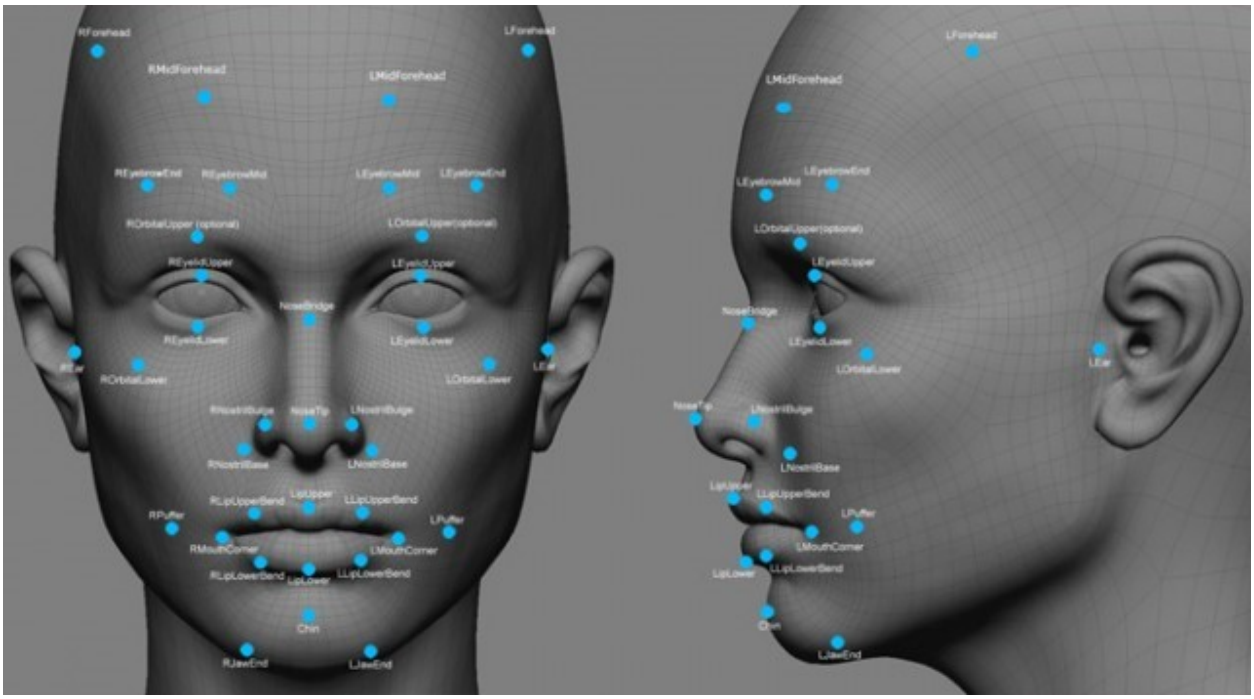

Figure 1.2: Basic principle of facial features

1.4 CONTRIBUTION

In the present work, detectors and classifiers of the characteristics that are considered most significant were developed, Excellent results were reported in mouth detection, with detection rates greater than 99% and errors comparable to the error from manual mouth marking. The lens sorter also obtained excellent results, with detection rates of 95% for databases with controlled environments and of the order of 90% for databases with uncontrolled environments. Beard and mustache classifiers after using the mouth detector obtained very good results, with a detection rate of over 95% in databases with uncontrolled environments.

1.5 THESIS ORGANIZATION

The thesis is structured as follows: Section 2 is where we review some of the previous work and implementation on smart grids. Section 3 is where we give a background about all the components. Section 4 is where we explain our model in details and implementation of that model in details, Section 5 is where we simulate our model and record the results obtained. Section 6 is where we conclude our work and put a future scope into perspective.

2. STATE OF THE ART

In order to be able to identify the individual previously, it is necessary to be able to identify a person's face, that is, it is necessary to perform facial detection, through the image received, of all the individuals found in the image. To proceed with the detection and facial recognition it is necessary to perform the following steps [19 Biometric recognition software plays an increasingly significant role in modern security, administration and business systems. Biometrics include fingerprint, retinal scanning, voice identification and facial recognition. Facial recognition has attracted particular interest, as it provides a discreet, non-intrusive means of detection, identification and verification, without the need for the subject's knowledge or consent. It is now commonplace in applications such as airport security, and has been widely embraced by law enforcement agencies, due to the improving accuracy and deploy ability of systems, and the growing size of face databases. A recent example of its success occurred in China; face recognition technologies were successfully used to identify and track a wanted fugitive at a concert attended by over 60000 people, resulting in his arrest [2]. As a consequence of the proven ability of deep neural network based systems to outperform human performance in face verification tasks, the international government sector is projected to be the largest user of face biometrics systems, including face recognition [3], over the next 10 years. Face recognition technologies are also at the forefront of a wide range of consumer applications and devices, from user verification tasks enabling account access, to digital camera applications, and social media tagging. Consumers and industry require rapid, affordable and efficient applications, to meet demand in business, employment and education functions, which extend to employee monitoring, roll call and security, reducing administrative costs, and procedural efficiency. Despite significant progress in recent years, and widespread usage, many shortcomings still exist. This survey will provide a comprehensive perspective on the current state of face detection, verification and identification technologies, highlighting limitations which must be rectified in order to progress to efficient, dynamic and versatile systems capable of meeting the needs of modern day usage. Since the 1990s, significant progress has been made in the realm of face detection and recognition [4]. Current research in both face detection and recognition algorithms is focused on Deep Convolutional Neural Networks (DCNN), which have demonstrated impressive accuracy on highly challenging databases such as the WIDER FACE dataset [5] and the Mega Face Challenge [6], as well as on older databases such as Labeled Faces in the Wild (LFW) [7]. Rapid advancements have been triggered due to the increasing affordability of

powerful GPUs [8], and improvements in CNN architecture design which is focused on real world applications [9]. Furthermore, large annotated datasets, and a better understanding of the non-linear mapping between input images and class labels has also contributed to the increase in research interest in DCNNs. DCNNs are very effective due to their strong ability to learn non-linear features, however they are inhibited by intensive convolution, and non-linear operations, which result in high computational cost [10]. Nevertheless, DCCNs are predicted to encompass future research and industry application, and are currently being deployed by large corporations such as Google, Facebook, and Microsoft [11]. No comprehensive survey of face detection, recognition and verification methods has been conducted since 2003, although several quality reviews have been conducted on specific aspects of facial recognition solutions. However, they lack complete coverage of all existing research areas and processes necessary to the development of effective solutions. To address this shortage, this paper will review all relevant literature for the period from 2003-2018 focusing on the contribution of deep neural networks in drastically improving accuracy. Furthermore, it will provide a comprehensive analysis of the state-of-the-art approaches in face detection, verification and identification, alongside a thorough coverage of the most recently developed databases and benchmarks, and preprocessing methods. Specific applications, such as video-based face recognition, and infrared recognition technologies will also be considered. Throughout the review, shortcomings and limitations are highlighted, and indicators of areas requiring future research and improvement are emphasized. Causes of inaccuracy and computational bottlenecks are explored, and recently proposed solutions evaluated in terms of real life applications. Furthermore, a comparative analysis of state of the art methods will be provided, as well as a brief overview of traditionally used methods which have recently been outperformed, but which provide an alternative to computationally expensive deep methods. In particular, this survey will identify areas of improvement in processing time, efficiency, cost and accuracy and analyses the means by which effectiveness is measured. Our contributions are outlined as follows; • Provide visually appealing comparative analysis of modern face recognition systems and databases using tables, graphs and figures revealing performance and key features, • Offer a critical analysis of the benefits and limitations of the state of the art methods, and a broad range of solutions designed to address shortcomings, • Summaries the role of current research in the context of traditional methodologies, outlining the development of technologies over time, • Highlight major shortcomings, and key areas requiring improvements in light of the latest research undertaken in specific areas of facial recognition.

3. MATERIALS AND METHODS

In this chapter we review some of the essential components that we used to implement our model, we give a brief introduction to the component.

3.1 MACHINE LEARNING

In recent years and due to the power that data have acquired in the so-called Big Data effect, Machine Learning has gained great importance. Machine Learning is defined as machine learning, included within the range of fields covered by Artificial Intelligence. Due to the large amount of data that is generated every day and given the amount of information that can be extracted from it, the need to process it arises. There is so much information generated that it is practically impossible for human beings to be able to manually process this data. Hence the idea of Machine Learning was born. Machines that automatically and massively perform the work that we cannot do, and that, based on a multitude of techniques, they manage to learn by themselves to facilitate the next information extraction tasks. In more precise terms, all Machine Learning systems are based on generating a model, which is in charge of processing all the information and making decisions for itself. This model will work obtaining an input of data, processing it, obtaining an output on it. All Artificial Intelligence techniques have in common that they try to emulate the intelligence that a human would provide to carry out this task. What differentiates Machine Learning from other AI techniques is that it performs automatic learning and updating of the parameters of its model. Today Machine Learning is one of the most widespread techniques in the world, since thanks to all the work that it can carry out quickly and precisely, it is present in most services, companies and generally in fields where it is work with large amounts of information. [11].

The problems or models that Machine Learning addresses can be differentiated into three types, based on the way in which they learn: - Unsupervised Learning: The data with which the system learns is not previously labeled, that is, we do not distinguish previously, no characteristic of them but rather this type of model is in charge of processing the data, and later recognizing certain patterns within the data set itself and classifying them according to what has been learned. In this project we will use one of the most widespread unsupervised techniques, clustering. - Supervised Learning: From a set of previously labeled data, the model learns by looking at them. Thus, once its parameters have been learned and adjusted, the model will be able to label new data. Within supervised learning, classification problems (establishing which

class an object belongs to, for example in our case, given images, classifying people's faces) and regression problems (predicting real values, such as the value of a property). In most of the project we will use this type of Machine Learning, since the neural networks used are based on these principles. - Reinforcement Learning: The model learns based on previous predictions, that is, it will take into account whether the prediction has been correct or not to modify its parameters. This type of learning is said to be based on experience, and does not require a large amount of information as is the case with the other two types. This type of technique has not been used in.

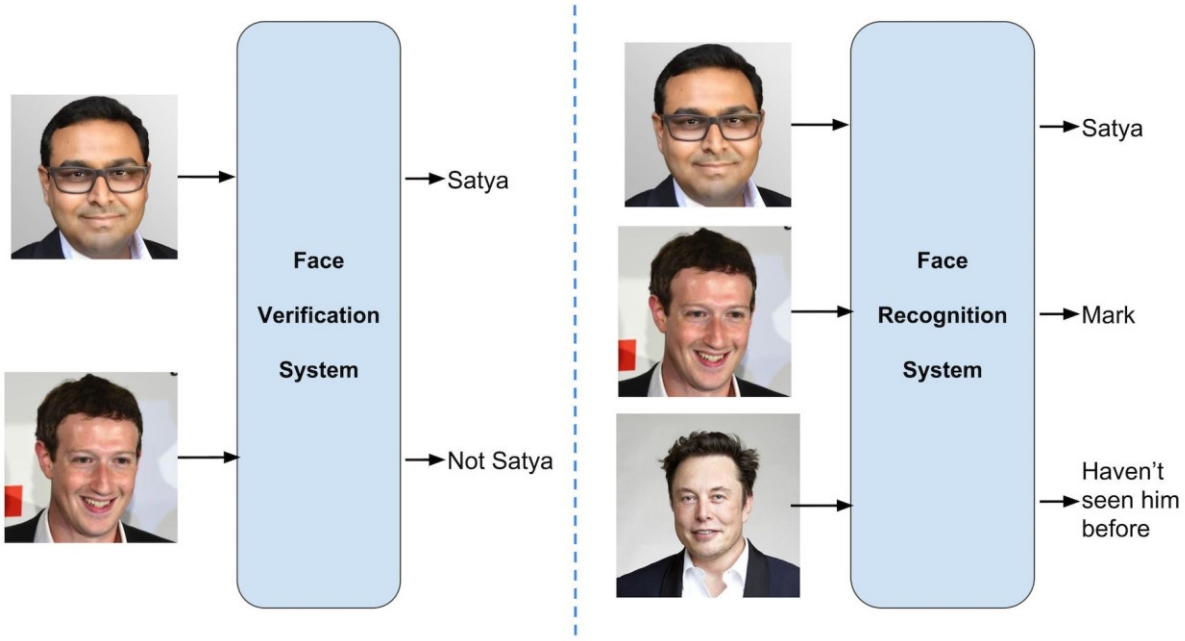

Figure 3.1 Facial features analysis using deep learning

3.1.1 Deep Learning

Although by definition Machine Learning is considered to encompass fields such as Deep Learning, many scientists go further and claim that it is the opposite, since Deep Learning is deeper and uses more complex algorithms. Machine Learning, on the other hand, only focuses on parsing data and providing a solution to it. In any case, we will not enter into debates, being objective and explaining what Deep Learning is based on. Deep Learning, or Deep Learning, tries to use techniques where data is passed through numerous layers (hence the name of deep learning) to carry out the learning process. One of the basic foundations in the creation of Deep

Learning was the analogy with the biological structure of neurons, and how the learning process is carried out in human beings. In fact, many of the questions that we will discuss below bear a great deal of resemblance to the disposition of human intelligence. One of the most widespread applications, in fact, is neural networks [19].

As we have said before, the amount of data that concerns us day by day is very large, and every year it grows. As time goes by, we have more accessibility to devices such as cameras, sensors ... All this leads to a constant growth in the number of data generated. Deep Learning is not something new, since it has been invented for many years, only that when we have data sets of a reduced size, the performance of classic classification or regression algorithms is the same as using a neural network, even better, since sometimes the implementation and the computational cost is lower. In the past and because of what we have mentioned above, the amount of data was not so large, which led us to use simple or traditional algorithms, obtaining similar results. Nowadays, with the amounts of data we work with, these algorithms have become stagnant and do not generate good performance, which forces us in most cases to use a neural network. The larger (or deeper) the neural networks are, the more difficult they will be to implement and train, but in turn, the more data they will offer better performance. Due to the performance offered by neural networks, and also because companies increasingly need to process their data more efficiently, it can be said that neural networks are in full commercial boom (the only thing necessary to make an efficient neural network is a large set of previously labeled data, and an extensive neural network).

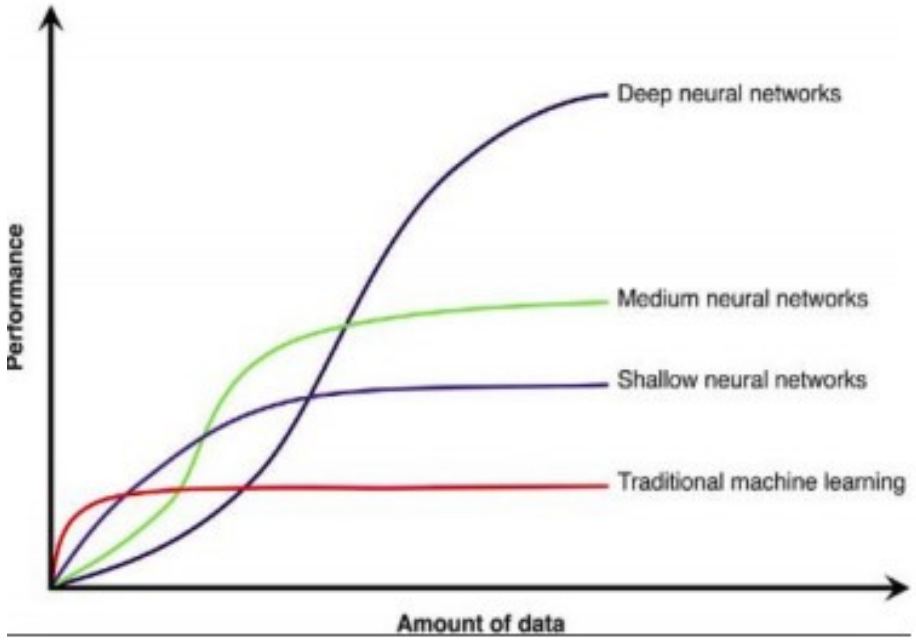

Figure 3.2: amount of facial data increase by year

3.1.2 Iterative Process

Every neural network requires resources and a process, so that before using it, it can give good results and be productive. But how quickly this iterative process is carried out will depend on several factors: - Data: We need to have a data source (or commonly called datasets), and that these have been previously labeled with what we want our model to learn. Perhaps the labeling process is one of the simplest, but at the same time more expensive of the whole general process, since it requires that someone has previously classified it correctly, and given the large size of the data sets, this entails a lot weather. There are techniques to save human labor in this labeling process, but it has not yet been achieved that this can be done automatically. - Computing: The computing process is one that is carried out by the machine, that is, all the time in which the machine processes data and updates values or performs calculations. This factor will depend on the material, that is, if the resources used are good, this will make this process be carried out more effectively. - Algorithms: The internal algorithms that we use to calculate values, using one or the other according to the types of data or results in general will also affect the process, since all of them do not have the same complexity of execution. Having said that, we can reflect the

iterative learning process, in the form of a wheel (Figure 2), since it is a process that never ends. In other words, a neural network is not always as efficient, just as the data changes over the years. That is why it will retrain and adapt thanks to this iterative process.

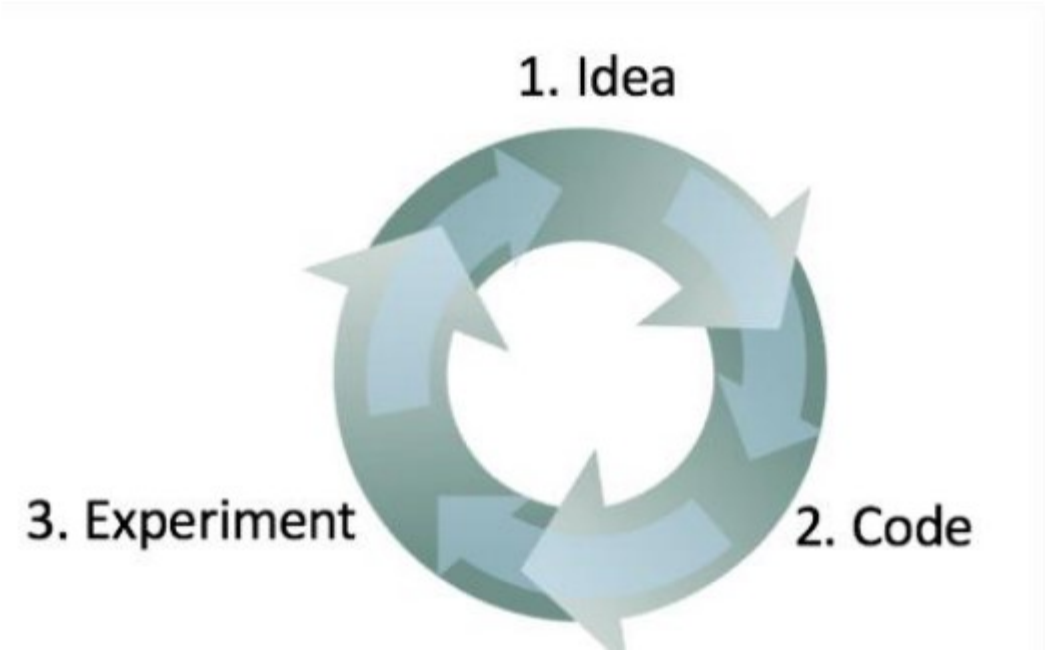

Figure 3.3: Iterative process in facial recognition

3.2 NEURAL NETWORKS

Once we know the scope and the way in which Machine Learning solves your problems, we will be more precise and we will focus on understanding the concept of neural networks. Neural networks are nothing more than a type of model used for solving Machine Learning problems. The objective of these, like any learning algorithm, is to give a prediction about a specific input based on previously learned data. These data are processed through neurons, which interconnected with each other form the so-called neural network. The more neurons connected, the deeper the network will be. Neurons are simply a human analogy, they are trying to model the behavior that a natural neuron would have and they are an essential unit of neural networks. - Input: This is the information that we will use to predict values. Usually these data belong to the set X, and each data is assigned the name of X (i), and it will be known as the i-th example. Each object or example will be distinguished by its characteristics (which will be the ones that we pass to the network), and N will be denoted as the total number of characteristics $(x1\ldots\ xn)$.

Therefore, we can say that xj (i) refers to characteristic j of the i-th example. The training set will be composed of M examples, and all of them will be stored in a dimension matrix (NxM). - Output: It is the result that the network will return once it has processed the information. The output will be denoted as Y, and its dimensions will depend on the structure of the network that we have used. - Weights: These are the values acquired by the weights of each neuron. These are updated in each iteration and their function is to store the values that will act in the linear application of the data. That is, the output will depend on the value of the weights. These refer to the link between an input characteristic and the neuron itself, where wi is the weight between the ith characteristic and the neuron. - Neuron: Also known as a perceptron, it is the smallest element that is responsible for processing information. It receives input information (input variables or characteristics), processes the information (applying the weights) and finally gives an output or prediction. It encompasses all the concepts that we have just explained. This what it does mainly is perform the weighted sum of the characteristics and weights, applies an activation function (we will see them later) and returns an output.

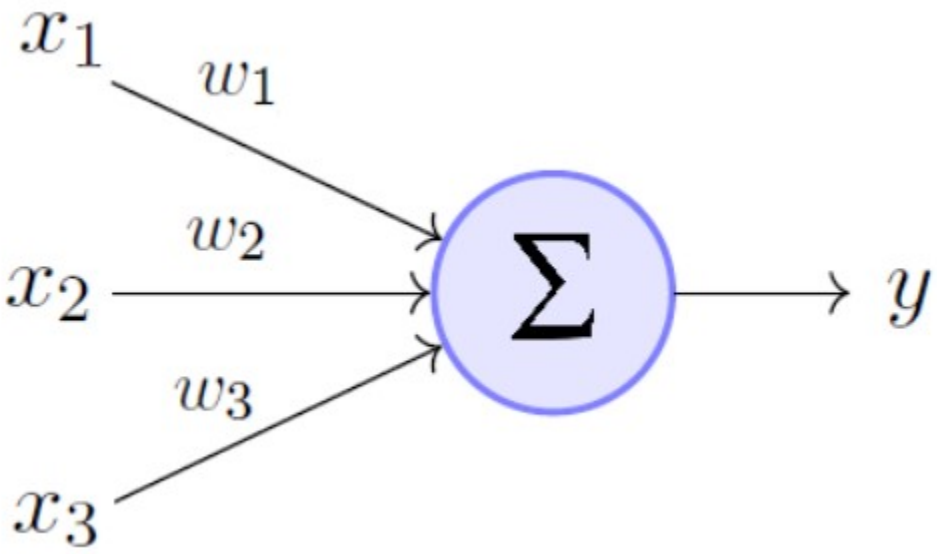

Figure 3.4: ANN structure

In the graphical representation of Figure 3, we can see the input characteristics ($x1$, $x2$, $x3$), the applied weights ($w1$, $w2$, $w3$) and the output variable (y). If this output variable is to be used as the input to a new linked neuron, we will call it activation. To calculate the output, we will apply the function: $\boldsymbol{y} = \boldsymbol{a}\ (\boldsymbol{z})\ \boldsymbol{z} = \boldsymbol{w0x0} + \cdots + \boldsymbol{wnxn} + \boldsymbol{b}$ These equations perform the prediction of

the output of a neuron. If you look closely, this looks a lot like the equation used in logistic regression. With the use of a single neuron like this we can only deal with linearly separable problems. In order to tackle more complex problems, we need to link more neurons like this, thus creating multilayer networks. In the part of Deep Networks, we will analyze this. Function a refers to a trigger function. These are used for the final calculation of the prediction. In binary classification, for example, applying this function we were able to obtain the prediction in the determined interval of values between zero and one. Trigger functions There are many trigger functions, many of them non-linear, and there is no protocol to determine which one works best. The choice of one or the other will essentially depend on the range in which the sample data falls. Here are some examples.

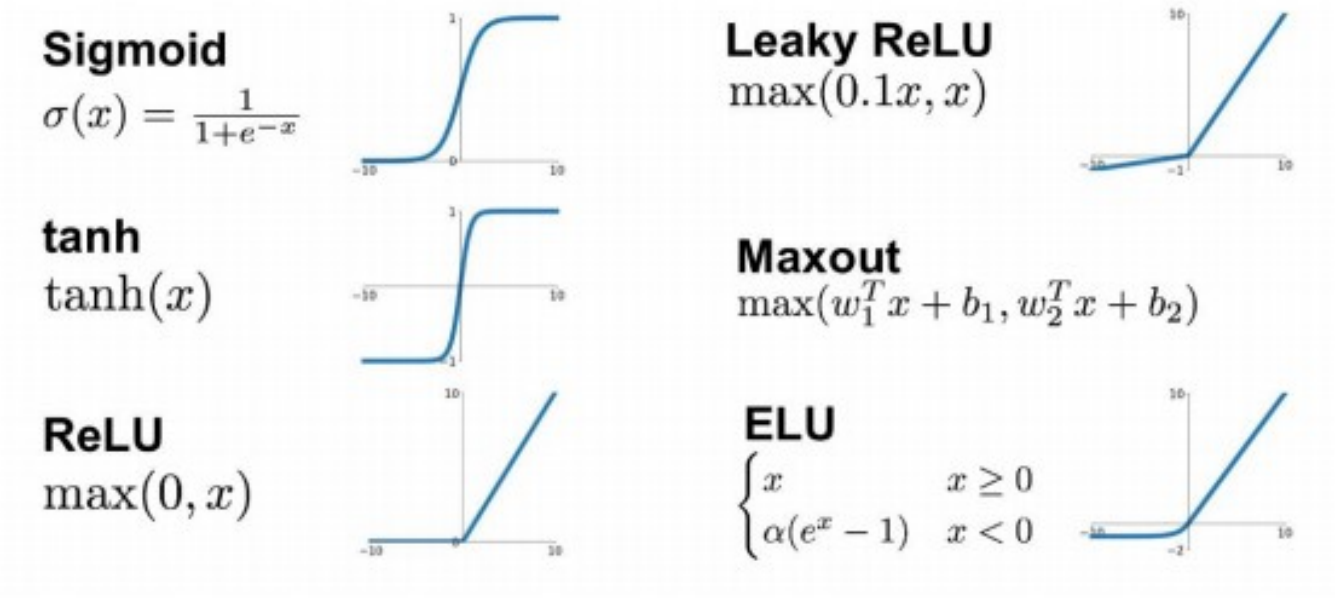

Figure 3.5 Functions in CNN

The most used activation function and the one that works best is ReLu, although in each layer of the neural network we are free to choose which one to use. In fact, it is recommended to use the Sigmoid function for the last output layer (as long as the classification is binary, that is, there are two classes of output). In case the output is multiclass, as we will see later, we will use softmax. Gradient descent and update of weights One of the most important objectives of neural networks is to find a good adjustment of the weights, in such a way that it generalizes the problem in the best possible way, achieving the least deviation when it comes to calculate new predictions, or, in other words, make the model as effective and efficient as possible. The error produced when

making predictions is measured through a cost function, or in English, loss function (in the case of regression). Like the trigger functions, different cost functions can be used. The most popular and the one that we will use will be the mean square error (MSE). This is given by the expression:

$$MSE = \frac{1}{M}\sum_{i=1}^{M}(real_i - estimado_i)^2$$

Obviously, the error is quite intuitive, since it is simply the sum of the times that the model has wrongly predicted the output in the entire data set. How do we reduce this error? All we have to do is update the weights iteratively, so that in each iteration the error decreases. This, like most algorithms that seek the optimal solution to a problem, is based on finding the solution through the gradient. Next, we will briefly explain the algorithm that we have worked so many times throughout the race.

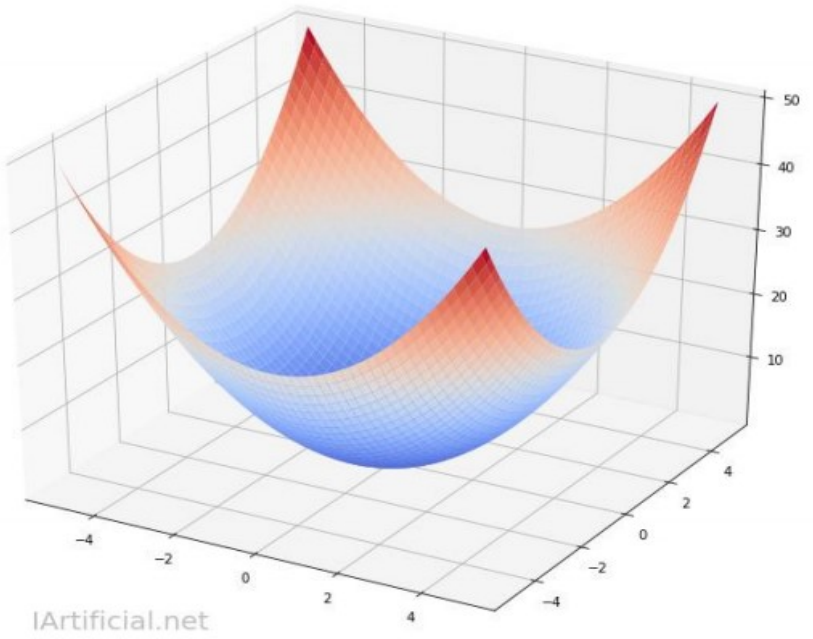

Figure 3.6: 3D artificial net

The gradient, given a function, aims to find the minimum value of that function iteratively. The function, in this case, is the error function that we mentioned earlier. That is, in each iteration of

the algorithm, it will try to update the values of the weights in such a way that each time we get closer and closer to the minimum of the function.

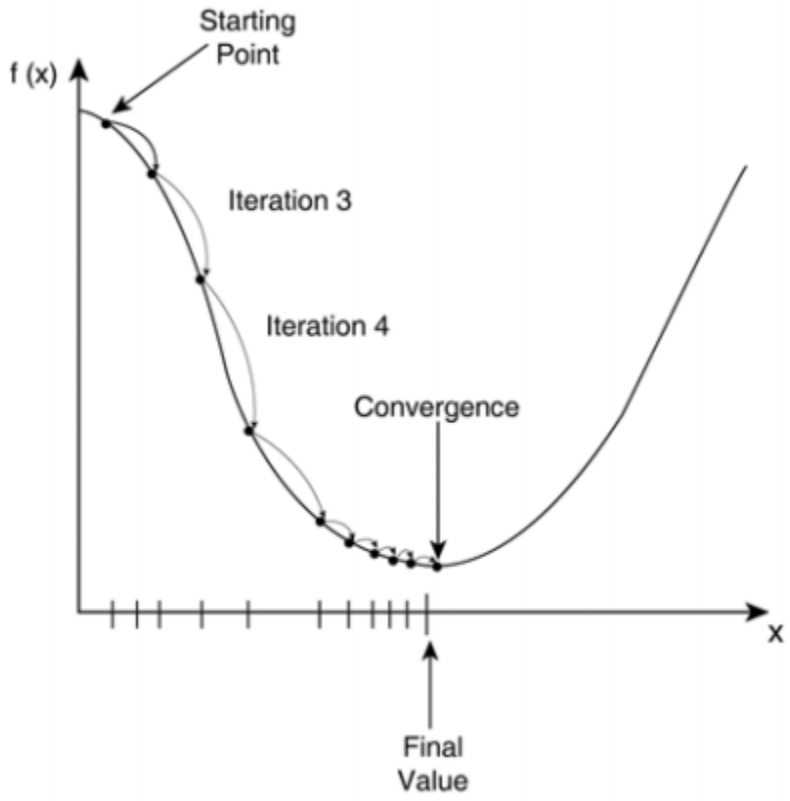

Figure 3.7: convergence in AI

we show the iterations performed in plane until reaching the minimum of the function. Each jump represents one iteration of the algorithm. The length of these jumps, or, rather, the speed of convergence is given by the parameter α. This will set the pace of convergence when updating the weights. We need to find an optimal setting for this parameter, to prevent the algorithm from making very small jumps (slowing down the algorithm), or against it, very large jumps, which can cause us to never find the minimum. To do this, we resort to the use of derivatives to update these weights. As we already know, the cost function will always converge on the only minimum that the function has, thus avoiding possible loops or entering local minimums. The derivatives will mark the direction of the maximum slope, or what is the same, the direction in which we want to descend if we want to find the minimum. These derivatives are made from the weight to be updated on the cost function J:

$$w_i^{new} = w_i^{old} - \alpha \frac{\partial J}{\partial w_i}$$

$$b_i^{new} = b_i^{old} - \alpha \frac{\partial J}{\partial b_i}$$

3.3 NORMALIZATION OF DATA

Normalization is a concept that has been covered throughout the career, either in set theories or in machine learning concepts. This consists of adapting the numerical data in such a way that they all belong to the same range of values. But why should we normalize the data? The characteristics of the examples often belong to ranges of values that are very different from each other, so when training the weights, some could have very high values referring to high-range values, and others, however, very small values. This if we transfer it to the theoretical foundation, causes us to obtain a cost function with a more elongated bowl shape (see comparison below, Figure 17), which makes the execution of the descent by gradient more expensive, having more iterations, while with the normalized data making the descent is easier, since the steps to be carried out are uniform and the search is more direct:

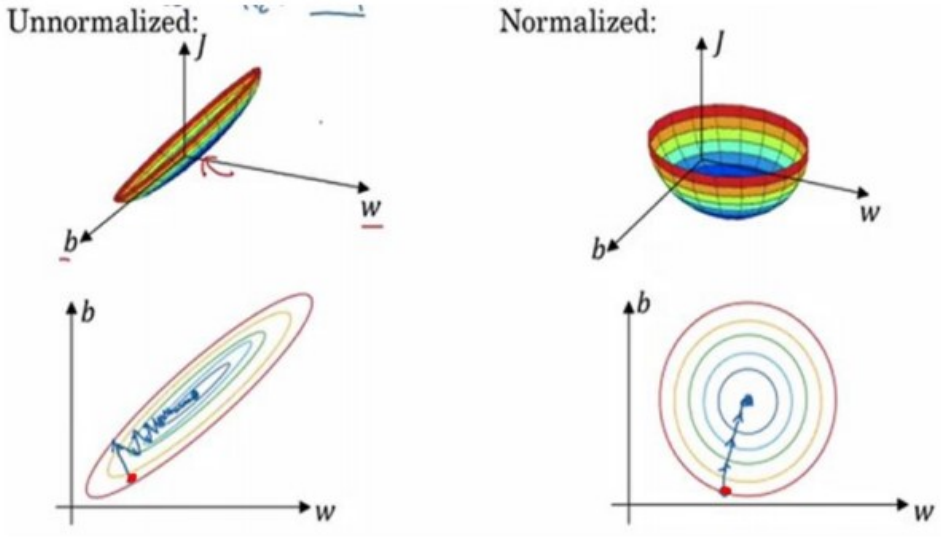

Figure 3.8: 3D convergence of AI particles

As we can see in the flat representations, we need many more iterations to find the minimum of the function, which implies a higher value of α in order to reduce the total number of iterations, running the risk that the function does not minimize correctly.

When working with a large amount of data such as deep neural networks, it is important to use optimization algorithms to improve the performance of the model, and in general, of the equipment. To do this and just as we did with the regularization techniques, we will review the optimization algorithms most used in Deep Learning. - Batch gradient descent (Mini-batch gradient descent): This optimization technique is used to reduce the computational cost of the original gradient descent algorithm (or Batch gradient desdent) and its main objective is the use of vectorization for the execution time savings. Let's imagine that we need to train a neural network of about fifty million pieces of data. In each run of the normal gradient we need to cycle through the entire dataset, which would slow down the training. The batch gradient offers a solution to this slowdown problem. The objective of this is to divide the complete data set into subsets (minibatches), so that each subset is processed in each iteration and the weights are updated, instead of having to use all the training examples in each iteration. If we illustrate the cost function with respect to the number of subsets, we see that, in the case of the gradient per mini batch, a greater noise is obtained in the results referring to cost, but it also descends towards a solution in a more efficient way.

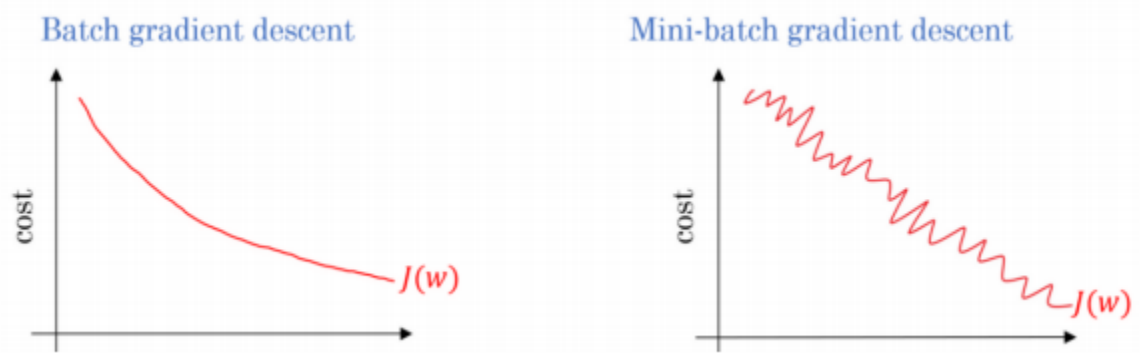

Figure 3.9: Badge gradients in facial detections

When using this technique, the efficiency will depend in part on the value that we assign to the number of mini-batches into which we want to split the original data set. If this value is very close to the size of the data set, that is, there is only a single batch, we will be applying the original gradient. On the other hand, if each batch contains exactly one example, we will have as

many minibatches as there are examples in the set, which gives rise to the so-called stochastic gradient, this being more random and can in turn be more:

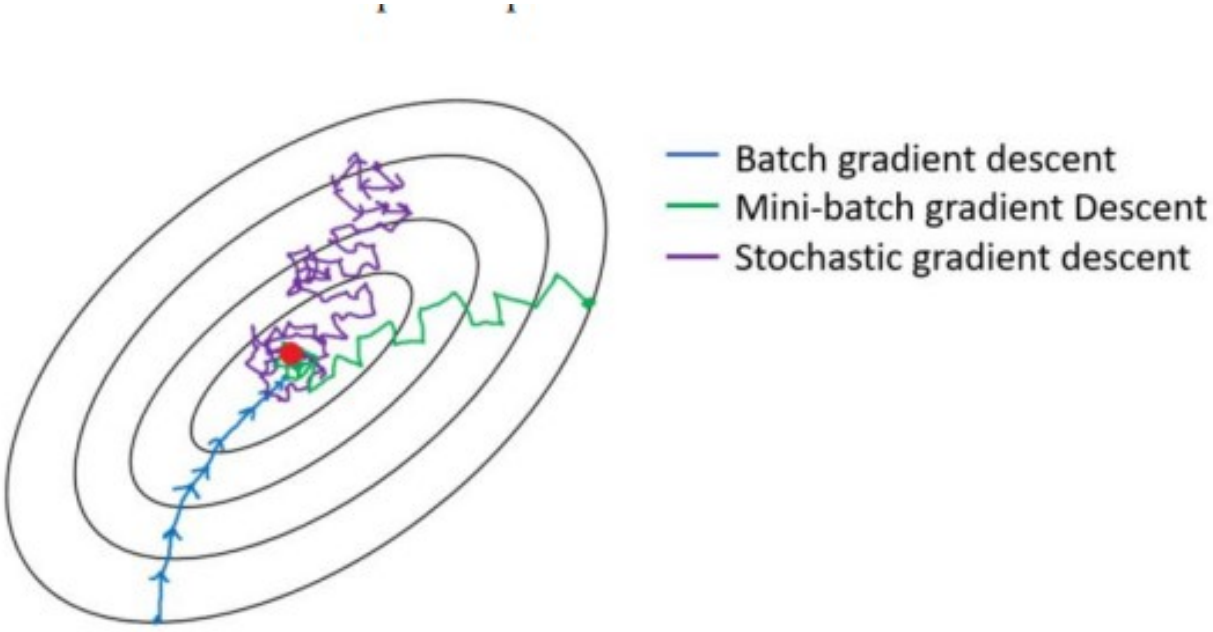

Figure 3.10: Match gradients of descent

3.4 SOFTMAX LAYERS

So far we have only seen how to perform a binary classification. Using the Softmax classifier, we can classify an example to multiclass. The last output layer will have as many nodes as there are classes, and each node will refer to each of these classes. As an output, each node will return a probability, which will represent the probability that the example has of belonging to that class. When classifying an example, we will award the class with the highest probability in its output. Therefore, normally, the Softmax layer will be the last output layer of our model. To better understand how it works, we will use an example: in the example, we want to make a classifier, which, based on photos of animals, we can obtain which animal belongs to being the classes "dog", "cat", "chicken" or "other". The neural network has previously been trained with photos of these types of animals. The Softmax layer, as we will see in the following diagram, returns four probabilities each referring to belonging to each class. Once we know a multitude of ways to make our model more efficient, as a general process a recipe applicable to every machine

learning model is established, which describes the complete optimization process, to find the best fit with respect to bias and variance. This, as a guide, summarizes the steps we have to follow to systematically improve the performance of the model. In a schematic and iterative way, we will use it once the initial model has been trained, to first reduce the bias, and then the variance.Once we have a general idea about the problems that neural networks are capable of addressing, we can go further into detail and analyze the different types of networks that exist and their specific functionalities. Introduction Within neural networks, there are various types of networks. Each of these types is focused on solving certain problems, or we will say that it is focused on working with a specific type of data. When we talk about a neural network that works with images, we enter into the universe of convolutional neural networks, or as they are also known, CNNs. These, unlike conventional neural networks, and like the human brain, are capable of detecting and differentiating characteristics within photos and working with each of these characteristics separately. That is, each neuron or CNN subnet will be responsible for treating the characteristics separately. In this way, the first layers of the network will be able to treat very specific characteristics such as edges or changes in pixel intensity, and as we go into the hidden layers of the network, we will go from recognizing edges to treating other characteristics more general such as the eyes, mouth, hair ... So until we reach the last layer, where we will get the full face. Colloquially, the CNN network will begin by distinguishing complex details, joining them, each time managing to differentiate more general aspects and objects.

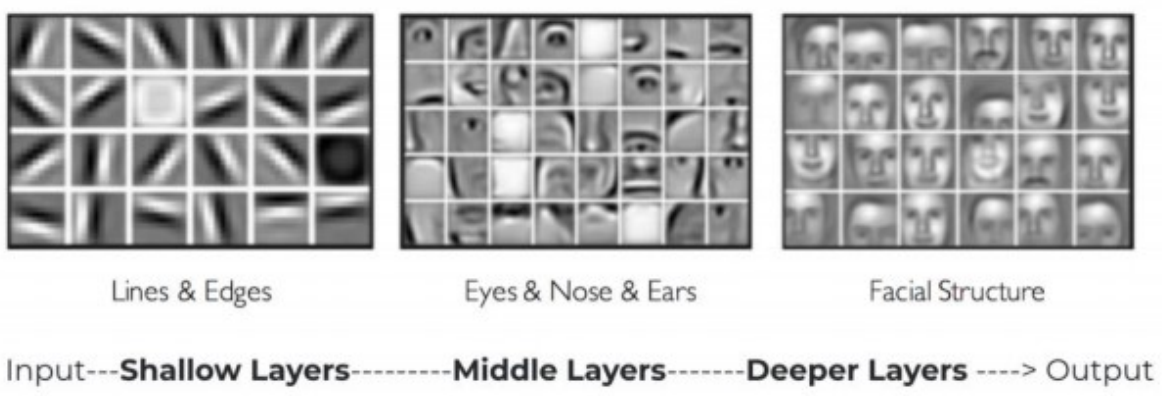

Figure 3.11: Layers of image features

For each pixel we do not process a weight, but we will work by blocks. This, in terms of efficiency, saves a lot of computational cost. That said, we will say that the main objective, and that is why CNNs are used in image processing, is that as we go deeper into the layers of the

network, the characteristics of the data are reduced, without losing any type of information, making the model more efficient. Next, we will see how a convolutional network is structured and how it works. Structure and operation of the CNN The input data for this type of network, being images, will have the form of a matrix. Generally, and as we already know, photographic files are represented by three matrices, one for each color channel. Each array will contain the values of the pixels in the image. If each pixel is treated as a characteristic, as we have said before, the amount of data to be manipulated for each photo is very high (and the more resolution it has, the more it is). But starting from this point, how do we manage to reduce the features without losing useful information along the way? It all comes down to using different filters across the layers of the network. These are applied on the data structure of the previous layer (following the previous example and initially, we will have a 28x28x3 structure), to achieve a "prediction" in the form of a block, in such a way that in each layer we obtain a different structure. In the Figure on the right, we get a visual idea of how we apply these filters by walking through the data structure:

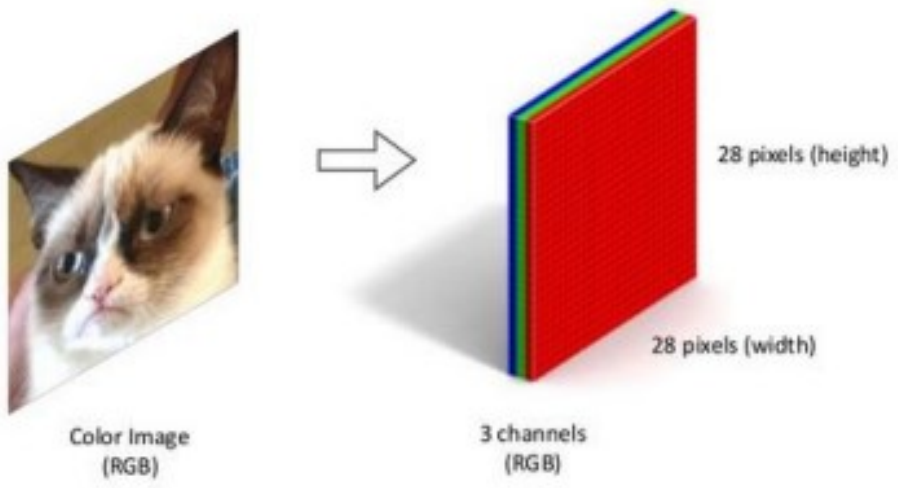

Figure 3.12: image to 3d channels

Mainly, there are two types of filters (also known as kernel or masks) present in convolutional networks. Each layer of the network will be named according to the filter that we have used in it: - Convolutional layers: They apply convolution filters. These filters are used to extract specific information from a structure. The convolutional filters of the first layers will detect very specific details progressively until they detect more general objects. In the field of image processing,

there are already many predefined filters, below we show some examples. - Pooling filters (pooling layers): They apply grouping filters. The objective of these filters is only to reduce the dimensionality of the input data, to later allow to apply a convolutional mask in the most efficient way, keeping all the possible information. Two types of pooling are the most used and those that give the best results: Max Pooling (they keep the maximum value of a subset of the input structure) and Average Pooling (they return the average value of the analyzed characteristics).

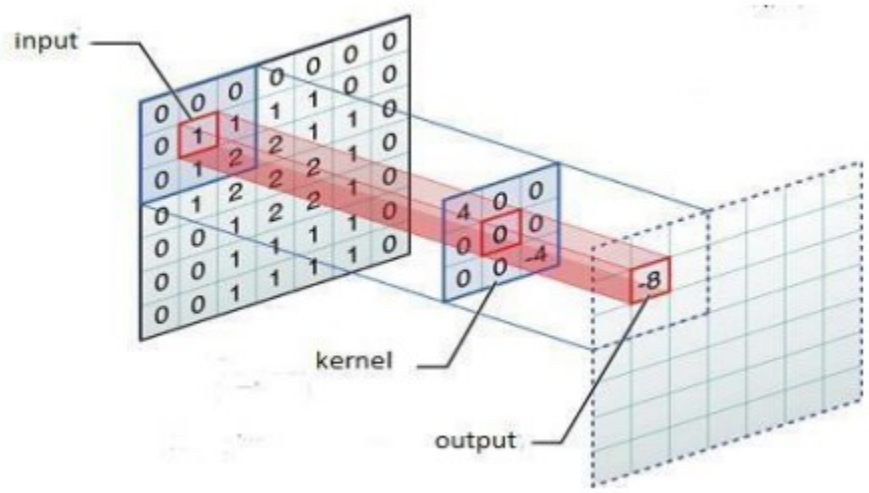

Figure 3.13: Image features in pixel values

we can observe a simple architecture of a CNN that performs an image classification. We can observe the structure of the data and the results of applying the convolution and pooling layers. Finally, layers known as Fully Connected Layers (FC) are applied to the final CNN layers. Its operation is similar to that of conventional neural networks, connecting all the characteristics resulting from the application of the filters, and thus carry out the subsequent processing of the data. In the case of classifiers, the previously mentioned Softmax layers will be used. In our case, we will be interested in storing the vector of characteristics that the network returns (these vectors are called embeddings), since for each image we will obtain a vector, preserving the most important characteristics, and so to speak, more malleable and differentiable. We will not go into detail, but we also have to mention that, when applying the filters, padding processes can be included. These are done when there are conflicts between the dimensions of the filter and the data structure. It consists of applying a fill on the data structure so that the dimensions allow the mathematical operation to be carried out. Likewise, there is also the possibility of applying a

sliding or stride technique. This is based on skipping some positions of the structure when applying the filter. The other functionalities, such as training or network evaluation, will be carried out in the same way as we did for normal neural networks. One of the advantages offered by the use of CNN networks is that there are many predefined architectures that have been updated over time, improving their performance. We will name the three best known and most used, to get a general idea of how they are structured in a real way: - LeNet-5: It was one of the first functional CNN architectures. It was used with around 60,000 data points. Today and with the amount of data that we are used to handling, it would be insufficient.

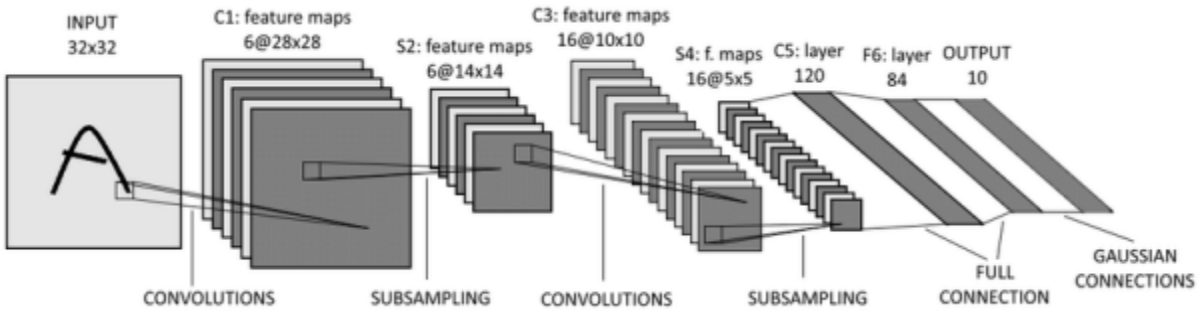

Figure 3.14: CNN layers in sampling

3.5 FACIAL FEATURES

Facial recognition is a very wide range within digital recognition, since a priori it seems a simple problem, but in reality it is deeper. Facial recognition, or face recognition, by definition is a technique in which a face detection is performed followed by a life detection (to ensure that a photograph cannot be used for identity theft). Within facial recognition, they differ: - Facial verification: Starting from an input image, a database and an identification, check if that person really is who they say they are (problem 1: 1). See the person with that Id in the database and say if she is or not. - Facial recognition: This problem, more complex than the previous one (1: k), tries to give an image, relate it to a person. That is, go through the entire database and assign that person to a class. We will face a recognition problem, since we have to link the user with those that we have stored in the database. And, if we want to include a new person in the database so that she can be recognized, do we need to go through the network training process every time? Obviously, if this were so, in concepts of complexity it would be unfeasible and not very functional. That is why the classification form will be different. One of the most important problems that exist when making a recognition system is that, most of the time, the data of each

person with which we form the database is very scarce, which generates a loss of precision. Throughout the project we will see how to deal with it. At this point in the project, we can say that we have a general view and the basis of all the concepts and tools with which this work is going to be carried out. As we progress through it, new concepts will appear that will be detailed in each part, but most of them will be directly related to the topics we have covered so far. Likewise, it should also be noted that the explanations of many of the theoretical concepts that we have seen in this section will be completed as we use them and progress in carrying out the project.

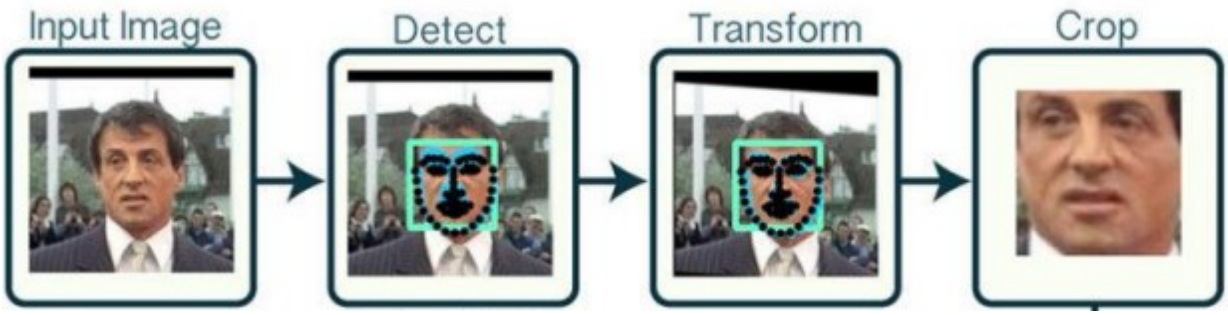

Figure 3.15: Facial detection by features

At this stage of the project, we will screen the images with which we are going to work. On the one hand, we will use only the faces of the images for subsequent facial recognition, and, on the other hand, we will use the complete images for the generation of descriptions (in the next chapter). Once we have cut out the faces, we will only keep those that are useful to us.

top of the diagram, identifying the photos that contain faces and even cropping the face, leaving the deal with the neural network for later. That said, we proceed to process the images, in order to detect as many faces as possible. The more faces of the user we identify; the greater precision we will obtain when it comes to recognizing it. To carry out this process we will base ourselves on using cascade Haar filters, whose implementation is provided by OpenCV, but first, we will explain the fundamentals of the algorithm on which said implementation is based.

OpenCV, in addition to providing the necessary functions to perform object detection, also makes available "XML" files containing previously trained models to be able to detect certain objects simply. We will take advantage of the implementation of the model that recognizes faces for our own implementation, but, in addition, we can make use of models that detect eyes, smile, sunglasses.

For this, we simply download the open source file “haarcascade_frontalface_default.xml. Which contains all the necessary tools for the process and we include it in the general program route. Once the OpenCV (cv2) libraries have been imported, we will instantiate the classifier object using Cascade Classifier, passing the downloaded xml file as an argument. Once this is done, we will have a face detector in images. To detect faces, all you have to do is call the detect Multiscale function and pass it the image to be detected (previously passed to grayscale to make it more efficient and accurate), along with some previously defined flags. The result of the call to the function is a list with the coordinates of the rectangles that enclose the faces. To observe the correct operation, we have drawn on some of our own images the faces detected by the matplotlib library, we can verify that the implementation works correctly even in complicated cases, since the patterns of the face are different when they appear with glasses. Despite this and how we can observe in the following Figure 50, the faces that appear incomplete or cut out are more difficult to recognize because they do not retain all the characteristics. I do not consider this important, since in most cases where more than one face appears, the face of the user to recognize that it is the one that interests us, it is very unlikely that it is the one that appears incomplete. Once the operation has been verified, we will proceed to apply the program on a set of images, that is, we will recognize all the faces that appear in all the photos (downloaded with the scraper) of the user to be analyzed. To do this, we create a folder called "faces" within the folder where the user's photos are, where all the clippings of the faces that appear in the photos will be stored. First, all scrambles will be stored (those of the user mixed with the faces of other people that appear in the photos), later we will see how we select only the user's faces. The main program is made up of a loop, which goes through all the images performing the steps mentioned above one by one (converting to grayscale and launching the detect Multiscale method). For each image we obtain a list of coordinates. To store only the cut of the face, just select the area formed by the coordinate rectangle on the general image and save it using miswrite in a file. We show a clipping of how a part of the resulting faces would be stored after running the application, in Figure 51. Now, we are only interested in the faces where the user appears so that we can train the network only with his face and recognize him later. For this reason, we need to discard the other detected faces, but how do we know which face is the user's face among all of them? We will explain how to make this selection below. Once we have obtained all the faces that appear in the photos, we want to keep only the user's faces, which we will use to train the convolutional facial recognition network and thus obtain greater efficiency. The difficulty of this section

consists of how to assign a face to the user we are dealing with, that is, which of all the cropped faces belong to the person in question. In our case, the percentage of cropped faces belonging to the user oscillates around 60%, This is why we will obviously assume that the majority class will belong to the user. The problem comes when an Instagram user does not contain photos in which her face appears, so the majority class prediction will fail, misassigning the user's face. In spite of everything, as we believe that this failure (in very few accounts it is the case that the user's face appears the least) is very unusual, we will stick to our assumption of assigning the majority face to the user in question. 4.2.1 Image processing Before applying any algorithm on the images, they need to be previously processed to be manipulable. In other words, we need to process these images through a convolutional network in order to extract all the relevant characteristics (later, in the Image Captioning section, we will use a different model to carry out this transformation). CNN and embeddings generation Based on everything that we have discussed in the theoretical sections on neural networks and convolutional networks, below, we will detail how to use this concept in our project. The main objective of this phase within the image processing will be to process all the input data (in this case the faces of each user resulting from applying face detection) through a convolutional network (CNN). This network will be responsible for, after an input image, extract all the characteristics of the face, returning them in the form of a vector, or also called embbeding. In this way, the images, which we currently store in the database, following its structure, in the form of a physical file (at the programming level in two-dimensional matrices, since they are in grayscale), will be processed through the model obtaining a set of embeddings as output. These vectors, on the one hand, at the physical level will be stored in a file within the path of each user. In this way we save ourselves from having to process all the faces every time we run the program, making use of a function that massively loads the content of the files (more efficient). And, on the other hand, at the computational level, it will store them in a dictionary (the user names the keys and the embeddings the values). The model: FaceNet in order to process the data and extract the characteristics, we first need to define a model and carry out its corresponding learning. For facial recognition to work properly, as we have seen in theory, the definition of the model plays a very important role and that it generalizes the data well. To achieve an optimal parameter fit, a sufficiently large and complete training data set is necessary, and besides, it is also necessary to have enough resources and time to perform this training of the model. Given the lack of the latter, we are going to make use of a very popular model, which is used in most of the AI facial recognition problems today., which is

previously trained and its integration is very simple. It is a convolutional network model (Facenet). Through this, the input images (faces) are broken down into the main characteristics of the face (eyes, nose, mouth ...). To use the model in the project, simply add the Facenet file in the path of our project (file containing the weights of the pretrained model) and load it with the load_model function provided by keras. Once this is done, we can use it through predict and the input image.

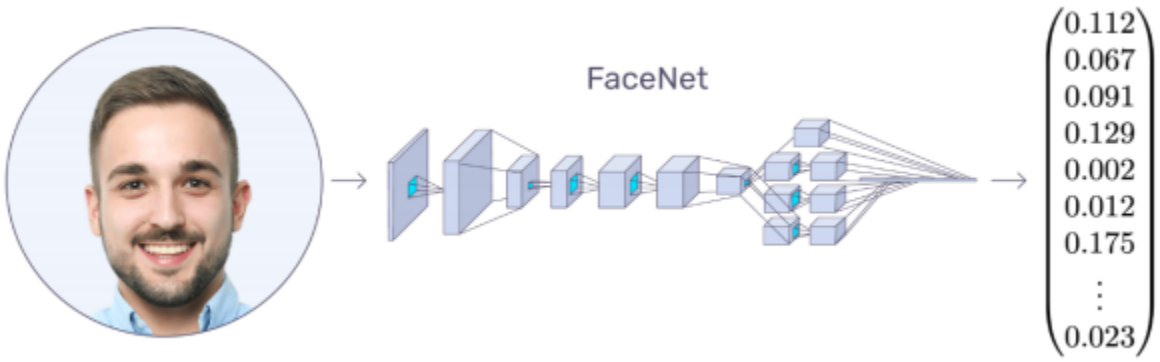

Figure 3.17: FaceNet layers

The first step that our program carries out will be to load the model mentioned as we have explained and go through the folder where the faces are stored, passing them one by one through it. From there we will obtain all the encodings of the images in vector form (embedding), which we will store in a dictionary together with the name or id of each image. In Figure 53 we show another visual example of this procedure. The embedding's will be vectors of dimension 4096, where each position of the vector will refer to a specific feature of the processed image, weighting that feature as defined by the model. In our FaceNet model, these weights will be focused on highlighting reliefs and details which make one face distinguishable from another. Where I'm going is that, obviously, if we had made a network focused on the detection of cars in images, the weighting of the weights and the extraction of characteristics would be completely different.

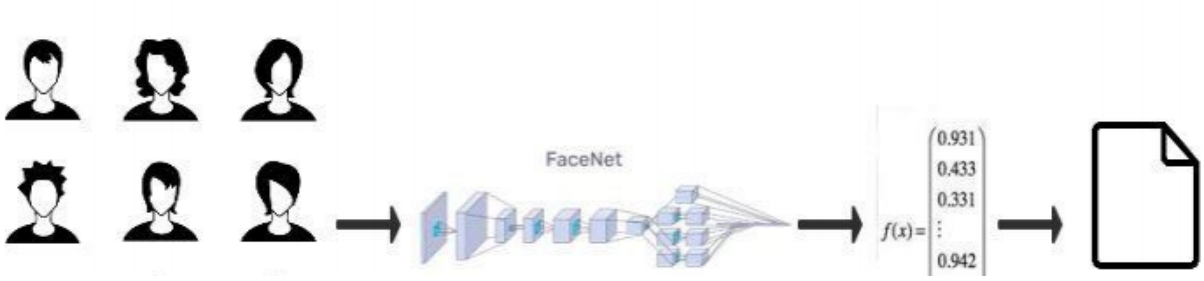

Figure 3.18: faces to layer data

As we well know, there is an algorithm called PCA, which allows data to be reduced in size, making it more distinguishable from each other. This is mathematically based on the calculation of the correlation matrix, in such a way that a linear transformation is applied to the original set of input data. The Sklearn library provides us with a functionality called PCA, which what it does is automatically perform a feature reduction by implementing the algorithm, thus being able to reduce the size of input features from 4096 (in our case) to the number of components we want. We have wanted to apply this technique to visualize the data in three-dimensional graphs (that is, we reduce the data set to three components), to observe in space the different groups of similar images that are generated by each user. We only do this process to visualize and better understand the data, since, to classify it, we will use all the characteristics of each embedding. We have applied this process to different users of the database to see the graphic representation of the faces of each one, generating, as we have said before, three main components for each image. In this way and as we can see in Figure 54, each graph refers to a user in the database, and the points are the spatial arrangement of the faces cut out of each one of them. We can see more or less clearly that in some, quite distinguishable data groups are generated from each other. Now, the next step is to know which of all the faces cut out by each Instagram profile is that of the user who owns the account. Starting from the basis, as we have said before, that we assume that the majority face in the data model will be that of the user, we will perform the K-means algorithm to classify the data that we have processed so far, and discard the unnecessary data (faces badly detected and faces that are not those of the user we are analyzing). We will try to briefly summarize the operation of the algorithm to get into context. Once we have identified the user's face, and we have discarded all the others, we obtain a reduced set of files that we will use to train our facial recognition system in the near future. In many users, and especially in accounts where the number of photos in which the user's face to be identified appears is low, it is necessary to increase the size of the training set to increase the

effectiveness of the model. But, how do we increase data only from what we already have? Indeed, we will not need to invent anything new, we will create new data, logically coherent, following different techniques that we have mentioned previously. In the field of artificial vision, where a large amount of training data is necessary, many times, as in our case, it is impossible to collect new information apart from what we already have, which would be optimal. Failing that, it resorts to creating information, altering what we already have in different ways. More specifically if we focus on the field of facial recognition, in addition to those we have already mentioned before, there are many other more specific techniques to increase our face training set. Three types are distinguished: - Generic transformations: These are differentiated into geometric (spatial transformations of the faces, such as rotations, turns ...) and photometric (changes in image color, contrast).

4. PROPOSED METHOD

In today's world, human-computer interfaces, such as those previously presented, are becoming more common and necessary, and imagining a system that a few years ago was only seen in a Hollywood movie is no longer so difficult. Due to various factors, the use of automatic interfaces, in which the person has to interact quickly and unobstructed, are becoming more and more common. The main motivation of this memory work is the study of facial characteristics, with the purpose of using them for example in human-computer interfaces. Of course, there are many other possible uses for the study of facial characteristics, such as security, automatic indexing or also as an aid to the always difficult goal of facial recognition. The characteristics that we seek to detect and classify in this memory work correspond to the most common characteristics that we can find in a face, and many times, the characteristics that as humans, we tend to memorize when we see a face for the first time. These characteristics can help to discern between different characteristics of the person, which can then be used for different purposes. There are, in the aforementioned classifiers and detectors, multiple difficulties when it comes to discerning, the most complex being the high degree of variety they present. For example, lenses, there are infinite types and styles of lenses, which suggests that it will be difficult to make an automatic classification of these. Why else mention beards, which today have gone from forming religious or political characteristics to a kind of accessory on men's faces. Although the Royal Spanish Academy defines them as "hair located under the area of the mouth", there are different types and positions of these

Face detection is a computational task that attempts to quickly and robustly detect faces in images or videos. As such, it is the first and one of the most important steps in the Face Analysis task, such as facial recognition, facial expression recognition, gender recognition, and others. With the rapid evolution of automated systems worldwide, face detection and analysis is becoming a necessary tool. This is why, despite starting in theory in the 70s, it has had a huge boom in the last decade in conjunction with the exponential increase in memory and processing capacity in computers. Faces, not being an easily definable object, and also because they are so variable, then face a double difficulty when trying to detect them. This is why the face detector must be robust enough to detect faces of different sizes, ages, genders, colors, with different hairs, lenses, beards or mustaches. Without neglecting the fact that these may have different backgrounds and lighting

The different types of detectors, particularly those of faces, are separated into two large levels of architecture: those based on the characteristics of the object to be detected, in this case predefined characteristics on the faces, and those based on images of examples, in this case, expensive and not expensive. As can be seen in the diagram in Figure 2, the former can also be separated into: Low Level Analysis, Components and Active Geometric Models. The Low Level Analysis is the one that occupies the information of the pixels, for example intensity and color. The second looks for geometric components that can be compared with the object to be detected. And the third, seeks to adjust geometric models that are associated with what you want to detect in the image, for example an ellipse to the face, circles to the eyes and a line to the mouth. Some of the detectors combine these methods with good results. With the disadvantage that occurs at the time of having variable funds, where some structures could seem a lot to one side. Like the work presented in [1] where first, using a skin detector, the skin is separated from the images and then using an ellipse-shaped blob detector, the faces are separated, to then detect eyes and mouth:

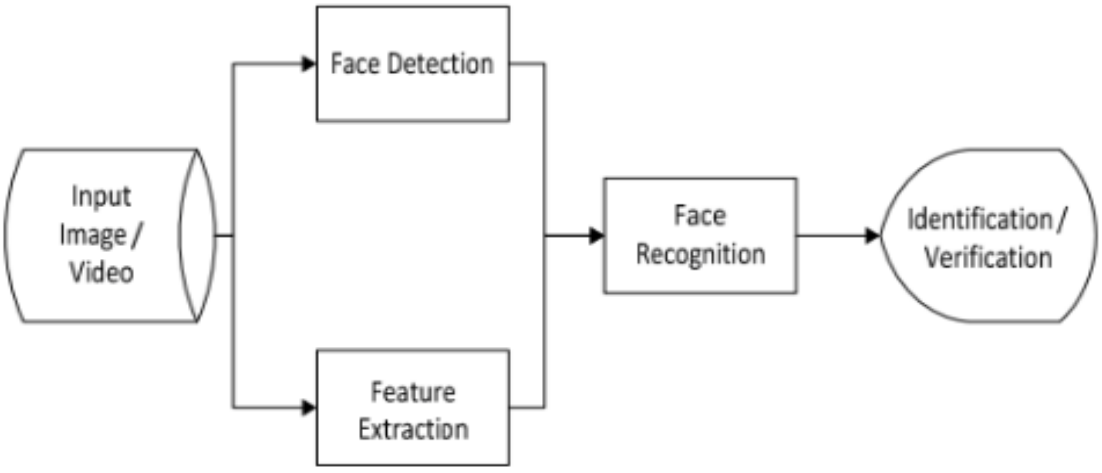

Figure 4.1: Input image to proposed network

4.1 FACIAL FEATURES CLASSIFIERS USING ADABOOST

It may sound strange, but this method serves both for the detection of faces and for the subsequent detection and classification of facial characteristics. However, as long as it is a two-class problem, AdaBoost meets all the requirements. At first, FACES and NON-FACES are used to detect faces in the images, and then, depending on what is required to be detected or characteristic that is required to be classified, images of the POSITIVE class and NEGATIVE class are used. The biggest difference in how to proceed with the detection problem and the facial characteristics classification problem is that when detecting with an AdaBoost classifier, the same method is used as when detecting faces. But this time on a face already detected as can be seen in Figure 5. That is, all possible windows are obtained within the Image (which this time could be only one face, or a part of it). Then each one of these images is delivered to a classifier based on the images, if the result is positive there is a success in detection. Otherwise, in that window extracted from the image, you cannot find what you want to detect. the detection of facial features using AdaBoost classifiers does not need, a priori, patterns to determine the detection. However, extensive training based on statistical algorithms is needed to arrive at the best possible classifier. Of course, it should be noted that the number of examples available is proportional to the result obtained, so the use of large databases is required in these cases. It should also be noted that, as mentioned before, in this situation an AdaBoost cascade is not used, but a single strong AdaBoost classifier. This is because by having the face already detected, the search area to detect a certain pattern is smaller and therefore much faster. In the case of classification, an important reason for occupying a single strong classifier is that obtaining objects of the negative or positive class is not so automatic. This is why it is much more difficult to conduct a training as long as the one that involves occupying more waterfalls. However, the fact of using a single cascade does not reduce the operation of the detectors or classifiers, since even so, the results are very encouraging.

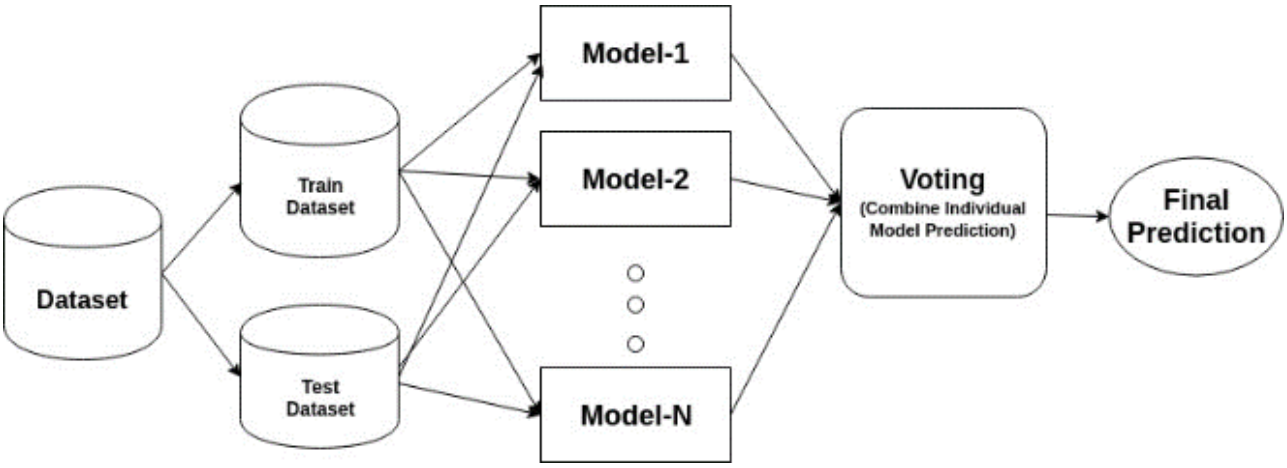

Figure 4.2: Adaboost classifier.

4.2 Proposed method

Before starting with the explanation of our Adaboost based model, certain criteria will be defined that will help to better understand the detectors and determine their performance. The first thing that is deemed convenient to introduce is the so-called Detection Rate. In the case of the face detector, the detection rate is defined as the percentage of faces that have been detected in an image, and as such it is the percentage to be maximized. On the other hand, the False Rates must also be defined, which are two. The first is the False Negative rate, which is the opposite of the Detection Rate, that is, the faces that have been forgotten by the detector. And the second is the rate of False Positives, which would be the non-expensive ones that have been erroneously detected by the detector as faces. If you think carefully it is very easy to obtain a detector with a high Detection Rate. You only need to detect faces everywhere and you will surely get the true faces between them. However, the true goal of a classifier is to maximize the Detection Rate by minimizing the False Positive Rate. In real life, it is extremely complex or impossible to obtain detectors that perfectly meet these two conditions, so it is sought to do the best possible. It is understood that there is some type of Trade-Off between these two conditions. Another important criterion for the analysis of classifiers based on the performance rates (Detection Rate and False positives) are the ROC curves. ROC (19 Receiver Operating Characteristic) curves are commonly used by doctors in diagnosing diseases. These result from graphing the detection rate vs false positives, moving the detection threshold at each point. Point from where the results greater than this are positive decisions and minor negative decisions. In this way, two classifiers of the same characteristic can be openly compared. The main objective is that there is a threshold

point from which the detection rate is maximum and the false positive rate is minimum. In Figure 7 a graph of two ROC curves is presented where it is observed that classifier 2 achieves better results than classifier 1 when the Detection Rate is greater than 73%.

4.2.1 Modified Local Binary Pattern M-LBP

The LBP (Local Binary Pattern, LBP for its acronym in English), was introduced and for the first time discussed in [9]. It is a texture analysis operator that is defined as a gray scale invariant measure, derived from the definition of texture in a local neighborhood. This has been widely used in countless image analysis work. There are different versions of LBP [10] although basically the idea is the same: a binary code that describes the local texture of a pixel and is built by doing a binary operation between this pixel and its neighbors. The first LBP 27 that was introduced in [9]. This operates with the 8 neighbors of a pixel and used the value of the intensity of the pixel in the center (in grayscale) as a reference to make a binary comparison with its neighbors, in this work, the mLBP, (modified LBP), introduced in [8] was used. The difference with the LBP already explained is that the mLBP uses the average intensity of all the pixels located in the neighborhood as a reference and makes the binary comparison with this average intensity, as shown in Figure 10. One of the advantages of using the average intensity as a reference is that it makes it invariant to lighting changes that affect all pixels equally. By using the average intensity of the neighborhood of pixels, any change in illumination becomes invariant to it.

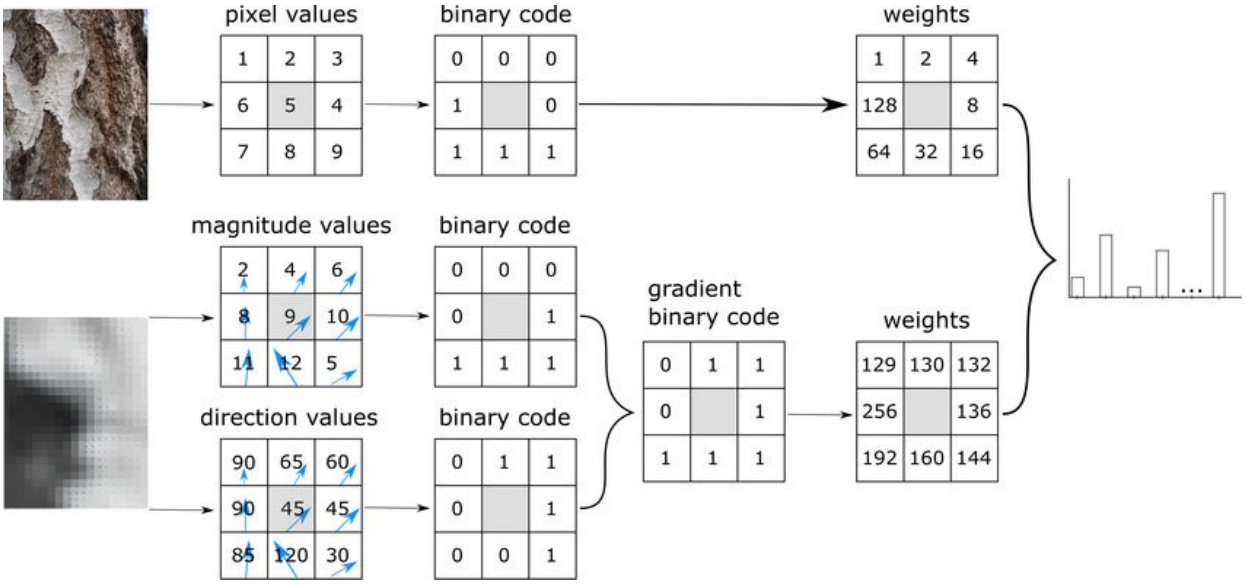

Figure 4.3: Pixels to features

Each of the classifiers separately met the proposed classification objective. Very good results were obtained for the detection of mouths, which were tested in 2 databases, 1 with controlled environments (BioID) and one with uncontrolled environments (UChileDB). Although better results were reported in the database with controlled environments, as expected, the results for the detection of mouths in the database with uncontrolled environments are very good. These results give great assurance that the detector can be used in other applications. Good results are obtained from the lens classification, which was tested on the FERET and UCHEYEGLASSES data bases. The first to contain a high range of examples of the class lenses and images of people with and without glasses. And the second for having examples with uncontrolled environments. In both databases the results were good. However, better results were presented in [25] on the FERET database, using the same method. It should also be taken into account that the result of the lens classifier is accompanied by the correct detection of the eyes. It is common that, because the example images had lenses, the detection of the eyes did not have the necessary precision to make an effective classification. Even though the results were removed the eye detector performed very poorly.

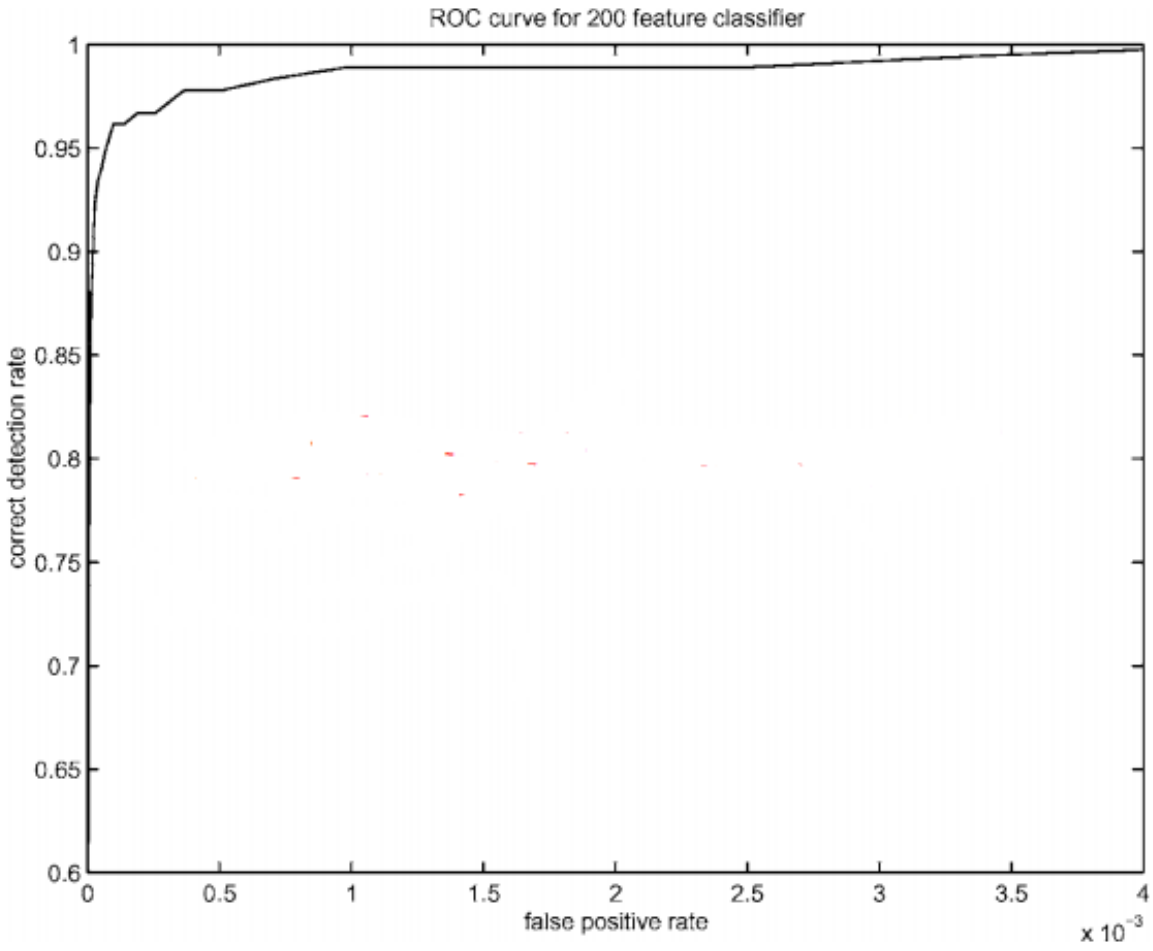

Figure 4.4: ROC curve of the classifier

Thus, it is proposed in the future to do a classifier test with the information from the marked eyes and not with the detected eyes. Although the results may improve as in [4], in the case of the gender classifier, occupying the classifier in this way does not make sense if you want to implement it in an automated user interface system. Both the Beard and Whisker classifier results were very good, considering how diverse and different the classification groups are. However, the study of other methods is pending in the future. It is perhaps thought that a simple study of the percentage of skin detected in the same 92 areas that is being classified could obtain the same or better results.

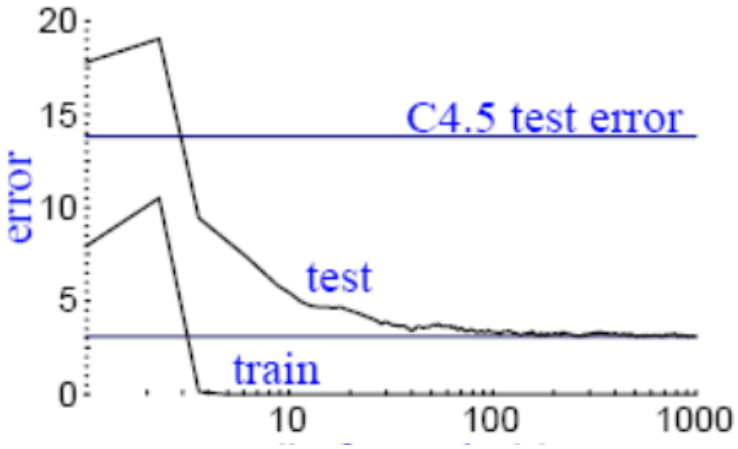

Figure 4.5: Generalized capability of Adaboost

Furthermore, the study of these same classifiers on databases with more examples would provide a better perception of the operation of both classifiers. It is also clear that the objective of the mouth detection, which was to improve the results of the beard and mustache classifier, was fully achieved. Both classifiers vastly improved the results. Since no subsequent work was found on the classification of these characteristics, the correct validation of the results is impossible. It is also important to conclude that both the Rectangular and LBPm features work better depending on the problem addressed, therefore it is always very important to study both features when developing an application. Along with that it is not always better to train classifiers or detectors with high resolution images. It is exemplified in the mouth detector and the whisker classifier where lower resolution images reported better and more robust results. For the age classification, and despite the fact that two studies were found with more than encouraging results, it is considered that the solution proposed in this study defines a more reliable result. Also robust since it does not occupy information on the geometry of the face. However, from what is seen in both works, it is clear that the separation of some specific classes is effective enough to be applied initially in the classification block. Finally, it is considered that the implementation of classifiers and detectors of facial characteristics using statistical learning was successful. It is observed how the detection of facial characteristics obtains amazing results and with much better processing speed than other methods also based on images. Also, in the field of the classification of facial characteristics, very positive results are seen. However, it is believed possible to be able to find even better results, for example, by expanding the variety in the training databases.

Despite this, the present work gives certainty to maintain that the Adaboost classifiers deliver very good results for the proposed objectives.

5. CONCLUSION

the problem of the detection and classification of facial characteristics using statistical algorithms was studied. Each of these was treated independently, so that they could be implemented as separate blocks in different systems. The choice of Adaboost as the statistical learning algorithm was made based on the good results presented in different works. It is clear that Adaboost works with very good results both for the detection of facial characteristics, and for their classification. In images with uncontrolled environments and different characteristics. For this reason, the study of classification with other types of algorithms was not included. When using image-based algorithms like Adaboost, databases are extremely important and must be sufficiently representative of the classes to be evaluated. For the cases of classification of facial characteristics, the bases occupied in some of the classifiers were not sufficiently numerous or varied. That is why in the future building larger and better databases would improve the results. The use of the semi-automatic creation of databases proposed in, used and modified in this work, provides an excellent source for the creation of databases. As long as said bases had the correct information (marked with the central point of the eyes and the mouth). Otherwise, manual markup of images was used to get the databases. The trainings carried out for each of these classifiers are also a very important step and as already mentioned, it goes hand in hand with obtaining the area of interest in the formation of the databases. These should be carried out multiple times trying to maximize your results. It is clear that the Adaboost classifier training system carried out in and used in the present work is highly effective and could be used for different applications.

REFERENCES

[1] Fg-net aging database. http://www.fgnet.rsunit.com.

[2] Ms-celeb-1m challenge 3. http://trillionpairs.deepglint.com.

[3] A. F. Abate, M. Nappi, D. Riccio, and G. Sabatino. 2d and 3d face recognition: A survey. Pattern recognition letters, 28(14):1885–1906, 2007.

[4] W. Abdalmageed, Y. Wu, S. Rawls, S. Harel, T. Hassner, I. Masi, J. Choi, J. Lekust, J. Kim, and P. Natarajan. Face recognition using deep multi-pose representations. In WACV, pages 1–9, 2016.

[5] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. IEEE Trans. Pattern Anal. Machine Intell., 28(12):2037–2041, 2006.

[6] G. Antipov, M. Baccouche, and J.-L. Dugelay. Boosting cross-age face verification via generative age normalization. In IJCB, 2017.

[7] G. Antipov, M. Baccouche, and J.-L. Dugelay. Face aging with conditional generative adversarial networks. arXiv preprint arXiv:1702.01983, 2017.

[8] Y. Atoum, Y. Liu, A. Jourabloo, and X. Liu. Face anti-spoofing using patch and depth-based cnns. In IJCB, pages 319–328. IEEE, 2017.

[9] A. Bansal, C. Castillo, R. Ranjan, and R. Chellappa. The dos and donts for cnn-based face verification. arXiv preprint arXiv:1705.07426, 5, 2017.

[10] A. Bansal, A. Nanduri, C. Castillo, R. Ranjan, and R. Chellappa. Umdfaces: An annotated face dataset for training deep networks. arXiv preprint arXiv:1611.01484, 2016. 19

[11] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua. Cvae-gan: finegrained image generation through asymmetric training. arXiv preprint arXiv:1703.10155, 2017.

[12] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua. Towards open-set identity preserving face synthesis. In CVPR, pages 6713–6722, 2018.

[13] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. IEEE Trans. Pattern Anal. Mach. Intell., 19(7):711–720, 1997.

[14] J. R. Beveridge, P. J. Phillips, D. S. Bolme, B. A. Draper, G. H. Givens, Y. M. Lui, M. N. Teli, H. Zhang, W. T. Scruggs, K. W. Bowyer, et al. The challenge of face recognition from digital point-and-shoot cameras. In BTAS, pages 1–8. IEEE, 2013.

[15] S. Bianco. Large age-gap face verification by feature injection in deep networks. Pattern Recognition Letters, 90:36–42, 2017.

[16] V. Blanz and T. Vetter. Face recognition based on fitting a 3d morphable model. IEEE Transactions on pattern analysis and machine intelligence, 25(9):1063–1074, 2003.

[17] N. Bodla, J. Zheng, H. Xu, J.-C. Chen, C. Castillo, and R. Chellappa. Deep heterogeneous feature fusion for template-based face recognition. In WACV, pages 586–595. IEEE, 2017.

[18] K. W. Bowyer, K. Chang, and P. Flynn. A survey of approaches and challenges in 3d and multi-modal 3d+ 2d face recognition. Computer vision and image understanding, 101(1):1–15, 2006.

[19] K. Cao, Y. Rong, C. Li, X. Tang, and C. C. Loy. Pose-robust face recognition via deep residual equivariant mapping. arXiv preprint arXiv:1803.00839, 2018.

[20] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age. arXiv preprint arXiv:1710.08092, 2017.

[21] Z. Cao, Q. Yin, X. Tang, and J. Sun. Face recognition with learningbased descriptor. In CVPR, pages 2707–2714. IEEE, 2010.

[22] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma. Pcanet: A simple deep learning baseline for image classification? IEEE Transactions on Image Processing, 24(12):5017–5032, 2015.

[23] B. Chen, W. Deng, and J. Du. Noisy softmax: improving the generalization ability of dcnn via postponing the early softmax saturation. arXiv preprint arXiv:1708.03769, 2017.

[24] B.-C. Chen, C.-S. Chen, and W. H. Hsu. Cross-age reference coding for age-invariant face recognition and retrieval. In ECCV, pages 768–783. Springer, 2014.

[25] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun. Bayesian face revisited: A joint formulation. In ECCV, pages 566–579. Springer, 2012.

[26] D. Chen, X. Cao, F. Wen, and J. Sun. Blessing of dimensionality: Highdimensional feature and its efficient compression for face verification. In CVPR, pages 3025–3032, 2013.

[27] G. Chen, Y. Shao, C. Tang, Z. Jin, and J. Zhang. Deep transformation learning for face recognition in the unconstrained scene. Machine Vision and Applications, pages 1–11, 2018.

[28] J.-C. Chen, V. M. Patel, and R. Chellappa. Unconstrained face verification using deep cnn features. In WACV, pages 1–9. IEEE, 2016.

[29] J.-C. Chen, R. Ranjan, A. Kumar, C.-H. Chen, V. M. Patel, and R. Chellappa. An end-to-end system for unconstrained face verification with deep convolutional neural networks. In ICCV Workshops, pages 118–126, 2015.

[30] Y. Cheng, J. Zhao, Z. Wang, Y. Xu, K. Jayashree, S. Shen, and J. Feng. Know you at one glance: A compact vector representation for low-shot learning. In CVPR, pages 1924–1932, 2017.

[31] I. Chingovska, A. Anjos, and S. Marcel. On the effectiveness of local binary patterns in face anti-spoofing. 2012.

[32] J. Choe, S. Park, K. Kim, J. H. Park, D. Kim, and H. Shim. Face generation for low-shot learning using generative adversarial networks. In ICCV Workshops, pages 1940–1948. IEEE, 2017.

[33] F. Chollet. Xception: Deep learning with depthwise separable convolutions. arXiv preprint, 2016.

[34] A. R. Chowdhury, T.-Y. Lin, S. Maji, and E. Learned-Miller. One-tomany face recognition with bilinear cnns. In WACV, pages 1–9. IEEE, 2016.

[35] F. Cole, D. Belanger, D. Krishnan, A. Sarna, I. Mosseri, and W. T. Freeman. Synthesizing normalized faces from facial identity features. In CVPR, pages 3386–3395, 2017.

[36] N. Crosswhite, J. Byrne, C. Stauffer, O. Parkhi, Q. Cao, and A. Zisserman. Template adaptation for face verification and identification. In FG 2017, pages 1–8, 2017.

[37] J. Deng, S. Cheng, N. Xue, Y. Zhou, and S. Zafeiriou. Uv-gan: Adversarial facial uv map completion for pose-invariant face recognition. arXiv preprint arXiv:1712.04695, 2017.

[38] J. Deng, J. Guo, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. arXiv preprint arXiv:1801.07698, 2018.

[39] J. Deng, Y. Zhou, and S. Zafeiriou. Marginal loss for deep face recognition. In CVPR Workshops, volume 4, 2017.

[40] W. Deng, J. Hu, and J. Guo. Extended src: Undersampled face recognition via intraclass variant dictionary. IEEE Trans. Pattern Anal. Machine Intell., 34(9):1864–1870, 2012.

[41] W. Deng, J. Hu, and J. Guo. Compressive binary patterns: Designing a robust binary face descriptor with random-field eigenfilters. IEEE Trans. Pattern Anal. Mach. Intell., PP (99):1–1, 2018.

[42] W. Deng, J. Hu, and J. Guo. Face recognition via collaborative representation: Its discriminant nature and superposed representation. IEEE Trans. Pattern Anal. Mach. Intell., PP (99):1–1, 2018.

[43] W. Deng, J. Hu, J. Guo, H. Zhang, and C. Zhang. Comments on “globally maximizing, locally minimizing: Unsupervised discriminant projection with applications to face and palm biometrics”. IEEE Trans. Pattern Anal. Mach. Intell., 30(8):1503–1504, 2008.

[44] W. Deng, J. Hu, J. Lu, and J. Guo. Transform-invariant pca: A unified approach to fully automatic facealignment, representation, and recognition. IEEE Trans. Pattern Anal. Mach. Intell., 36(6):1275–1284, June 2014.

[45] W. Deng, J. Hu, N. Zhang, B. Chen, and J. Guo. Fine-grained face verification: Fglfw database, baselines, and human-dcmn partnership. Pattern Recognition, 66:63–73, 2017.

[46] C. Ding and D. Tao. Robust face recognition via multimodal deep face representation. IEEE Transactions on Multimedia, 17(11):2049–2058, 2015.

[47] C. Ding and D. Tao. Trunk-branch ensemble convolutional neural networks for video-based face recognition. IEEE transactions on pattern analysis and machine intelligence, 2017.

[48] P. Dou, S. K. Shah, and I. A. Kakadiaris. End-to-end 3d face reconstruction with deep neural networks. In CVPR, volume 5, 2017.

[49] C. N. Duong, K. G. Quach, K. Luu, M. Savvides, et al. Temporal nonvolume preserving approach to facial age-progression and age-invariant face recognition. arXiv preprint arXiv:1703.08617, 2017.

[50] H. El Khiyari and H. Wechsler. Age invariant face recognition using convolutional neural networks and set distances. Journal of Information Security, 8(03):174, 2017.

[51] C. Galea and R. A. Farrugia. Forensic face photo-sketch recognition using a deep learning-based architecture. IEEE Signal Processing Letters, 24(11):1586–1590, 2017.

[52] M. M. Ghazi and H. K. Ekenel. A comprehensive analysis of deep learning based representation for face recognition. In CVPR Workshops, volume 26, pages 34–41, 2016.

[53] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In NIPS, pages 2672–2680, 2014.

[54] P. J. Grother and L. N. Mei. Face recognition vendor test (frvt) performance of face identification algorithms nist ir 8009. NIST Interagency/Internal Report (NISTIR) - 8009, 2014.

[55] G. Guo, L. Wen, and S. Yan. Face authentication with makeup changes. IEEE Transactions on Circuits and Systems for Video Technology, 24(5):814–825, 2014.

[56] S. Guo, S. Chen, and Y. Li. Face recognition based on convolutional neural network and support vector machine. In IEEE International Conference on Information and Automation, pages 1787–1792, 2017.

[57] Y. Guo, J. Zhang, J. Cai, B. Jiang, and J. Zheng. 3dfacenet: Real-time dense face reconstruction via synthesizing photo-realistic face images. 2017.

[58] Y. Guo and L. Zhang. One-shot face recognition by promoting underrepresented classes. arXiv preprint arXiv:1707.05574, 2017.

[59] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In ECCV, pages 87– 102. Springer, 2016.

[60] A. Hasnat, J. Bohne, J. Milgram, S. Gentric, and L. Chen. Deepvisage: ´ Making face recognition simple yet with powerful generalization skills. arXiv preprint arXiv:1703.08388, 2017.

[61] M. Hasnat, J. Bohne, J. Milgram, S. Gentric, L. Chen, et al. von ´ mises-fisher mixture model-based deep learning: Application to face verification. arXiv preprint arXiv:1706.04264, 2017.

[62] M. Hayat, M. Bennamoun, and S. An. Learning non-linear reconstruction models for image set classification. In CVPR, pages 1907–1914, 2014.

[63] M. Hayat, S. H. Khan, N. Werghi, and R. Goecke. Joint registration and representation learning for unconstrained face identification. In CVPR, pages 2767–2776, 2017.

[64] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, pages 770–778, 2016.

[65] R. He, X. Wu, Z. Sun, and T. Tan. Learning invariant deep representation for nir-vis face recognition. In AAAI, volume 4, page 7, 2017.

[66] R. He, X. Wu, Z. Sun, and T. Tan. Wasserstein cnn: Learning invariant features for nir-vis face recognition. arXiv preprint arXiv:1708.02412, 2017.

[67] X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang. Face recogni- 20 tion using laplacianfaces. IEEE Trans. Pattern Anal. Mach. Intell., 27(3):328–340, 2005.

[68] S. Hong, W. Im, J. Ryu, and H. S. Yang. Sspp-dan: Deep domain adaptation network for face recognition with single sample per person. arXiv preprint arXiv:1702.04069, 2017.

[69] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.

[70] G. Hu, Y. Yang, D. Yi, J. Kittler, W. Christmas, S. Z. Li, and T. Hospedales. When face recognition meets with deep learning: an evaluation of convolutional neural networks for face recognition. In ICCV workshops, pages 142–150, 2015.

APPENDIX OF ALL THE CODES USED

```
      %
      % Supervised descent method for facial landmark detection and
face tracking
      %
      % Example code for facial landmark detection
      %
      close all;
      clear;
      clc;

      addpath('./src/');

      DEBUG = 0; % set this value to 1 to show intermediate results

      %% load training data and test data of the COFW dataset,
inlcuding face images, landmarks and face bounding boxes
      load('./data/COFW_train_color.mat');
      load('./data/COFW_test_color.mat');

      % convert the data format of COFW to ours
      for i = 1:length(IsTr)
          if size(IsTr{i},3) == 3
              train_img(i).data = rgb2gray(IsTr{i});
          else
              train_img(i).data = IsTr{i};
          end
      end
      train_gt_shape = phisTr(:,1:58)';
      train_bbox = bboxesTr';

      for i = 1:length(IsT)
          if size(IsT{i},3) == 3
              test_img(i).data = rgb2gray(IsT{i});
          else
              test_img(i).data = IsT{i};
          end
      end
      test_gt_shape = phisT(:,1:58)';
      test_bbox = bboxesT';

      clear bboxesT bboxesTr IsT IsTr phisT phisTr;

      %% resize all the training faces
      mean_face_size = sum(train_bbox(3,:) .*
train_bbox(4,:))/length(train_bbox);
      for i = 1:length(train_img)
          face_size = train_bbox(3,i) * train_bbox(4,i);
          scale = mean_face_size / face_size;
          train_img(i).data = imresize(train_img(i).data, scale);
          train_gt_shape(:,i) = train_gt_shape(:,i) * scale;
```

```
        train_bbox(:,i) = train_bbox(:,i) * scale;
    end

    %% generate initial shapes for each training image
    % !!!!! It should be noted that you can generate multiple
initial shapes for each face image !!!!!
    % !!!!! This will greatly improve the performance of your
trained model  !!!!!

    mean_shape = mean(train_gt_shape,2);
    for i = 1:length(train_gt_shape)
        for j = 1
            train_init_shape(i).data(:,j) = project_s2b(mean_shape,
train_bbox(:,i));
        end
    end

    %% debug
    if DEBUG
        for i = 1:10
            imshow(train_img(i).data);
            hold on;
            plot(train_init_shape(i).data(1:end/2,1),
train_init_shape(i).data(end/2+1:end,1), 'y*');
            plot(train_gt_shape(1:end/2,i),
train_gt_shape(end/2+1:end,i), 'ro');
            hold off;
            pause(0.5);
        end
    end

    %% train cascaded linear regressors in SDM
    % intialise model parameters, you can tune these parameters
using cross validation
    cr_model.deepth = 5; %number of weak regressors in cascade
    cr_model.lambda = 1000; %weight of the regularisation term
    cr_model.patch_radius = 16; %radius of the local patch around a
landmark, used for extracting local features
    cr_model.mean_shape = mean_shape; %mean shape
    cr_model.mean_face_size = mean_face_size; %mean face size

    % train a new model
    % !!! or load a pre-trained model from our data by uncommting
the next line
    % load('./model/cr_model_cofw.mat');
    cr_model.model = train_sdm(train_img, train_init_shape,
train_gt_shape, cr_model);
    save('./model/cr_model_cofw.mat', 'cr_model');

    %% apply the trained model to test images
    for i = 1:length(test_img)
        face_size = test_bbox(3,i) * test_bbox(4,i);
        scale = cr_model.mean_face_size / face_size;
```

```
          tmp_img = imresize(test_img(i).data, scale);
          tmp_bbox = test_bbox(:,i) * scale;
          init_shape = project_s2b(cr_model.mean_shape, tmp_bbox);
          predict_shape(:,i) = fit_sdm(tmp_img, init_shape, cr_model)
/ scale;

          if DEBUG
              imshow(test_img(i).data);
              hold on;
              plot(predict_shape(1:end/2,i),
predict_shape(end/2+1:end,i), 'y*');
              plot(test_gt_shape(1:end/2,i),
test_gt_shape(end/2+1:end,i), 'ro');
              hold off;
              pause(0.5);
          end
      end

      %% plot results
      % caculating the normalised fitting error for each test image
      for i = 1:length(test_img)
          error(i) = 0;
          for j = 1:size(predict_shape,1)/2
              error(i) = error(i) + sqrt((predict_shape(j,i) -
test_gt_shape(j,i))^2 + (predict_shape(j+end/2,i) -
test_gt_shape(j+end/2,i))^2);
          end
          eye_distance_vector = [test_gt_shape(17,i) -
test_gt_shape(18,i), test_gt_shape(17+end/2,i) -
test_gt_shape(18+end/2,i)];
          error(i) = error(i) /
norm(eye_distance_vector)/(size(predict_shape,1)/2);
      end

      disp(['Average error normalised by inter-ocular distance: '
num2str(sum(error)/length(error))]);

      % plot the cumulative error distribution curve
      error = sort(error);
      plot(error, [1:length(error)]/length(error), 'm-', 'linewidth',
2);
      xlim([0 0.2]);
      ylim([0 1]);
      grid on;
      legend('SDM');
      xlabel('Normalised error by inter-ocular distance');
      ylabel('Propotion of test images');
      title('cumulative error distribution curve')



      %
      % Supervised descent method for facial landmark detection and
```

```
face tracking
      %
      % Subfunction used for fitting SDM to a new image
      %
      % Copyright @ Zhenhua Feng, fengzhenhua2010@gmail.com

      function predict_shape = fit_sdm(img, init_shape, cr_model)

      predict_shape = init_shape;

      for i = 1 : cr_model.deepth
          feature = obtain_features(img, predict_shape, cr_model);
          predict_shape = predict_shape + cr_model.model(i).A(1:end-
1,:)' * feature + cr_model.model(i).A(end,:)';
      end

      predict_shape = predict_shape(:);
      end



      %
      % Supervised descent method for facial landmark detection and
face tracking
      %

      function features = obtain_features(img, shape, cr_model)
      features = [];
      for i=1:size(shape,1)/2
          x = floor(shape(i));
          y = floor(shape(i+end/2));
          IX_X = x - cr_model.patch_radius + 1 : x +
cr_model.patch_radius;
          IX_Y = y - cr_model.patch_radius + 1 : y +
cr_model.patch_radius;
          IX_Y(IX_Y<1) = 1;
          IX_X(IX_X<1) = 1;
          IX_Y(IX_Y>size(img,1)) = size(img,1);
          IX_X(IX_X>size(img,2)) = size(img,2);
          tmp = img(round(IX_Y), round(IX_X),:);
          hogf = extractHOGFeatures(tmp);
          features = [features; hogf(:)];
      end
      end
```