‘Learning convolutional Neural Network for Face Anti-Spoofing”

Beatriz Gómez Ayllón

June 20, 2017

Abstract

This thesis proposes an algorithm based on Deep Learning whose main objective consists in being able to detect face spoofing attacks.

CASIA, FRAV and MFSD-MSU databases are used in the entire document for each experiment. Each database is constituted by face images of genuine users and different spoofing attacks on people. Spoofing images nature could be printed photos, masks and photos or videos displayed on a smartphone or tablet device of genuine users.

Once having established that, a Convolutional Neural Network (CNN) have to be designed and built for this specific problem and the best suitable classifier have to be discussed and selected.

The architecture of the Convolutional Neural Network is developed from LeNet-5 architecture. In order to get the final architecture of the Convolutional Neural Network, all of the steps and decisions made, as well as the the personalization of each classifier are all explained.

Images of databases are used to feed the input of the CNN. The features obtained at the output of the CNN are used as the input for training and testing classifiers. The selected classifiers which are used are K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Tree and Logistic regression. Additionally, reduction algorithm methods such as LDA and PCA are utilized with each classifier.

Classifiers and dimensionality reduction algorithms are personalized for each database and each experiment.

Python has been used to succeed with the objective of the thesis as the programming language. Conversely, Theano has been preferred as the main library to develop the architecture of the Convolutional Neural Network and training and testing processes.

From the results that were obtained, we can conclude with the fact that the most precise classified database is RGB+NIR (added in feature level) FRAV database due to

the perfectly classification result that is obtained from it.

Acknowledgements

First and foremost, I would like to share my sincere gratitude to my supervisor, Dr. Aristeidis Tsitiridis, for his continuous support, his extensive patience, and the advice that have guide me during the development of the thesis and the research work. He has encourage me to improve my work and my professional skills.

I am grateful for the help of Dra. Cristina Conde, co-director of this thesis. Furthermore, I would like to thank the FRAV research group due to the given facilities. Particularly, I would like to thank to David Ortega because of his help. Thanks to Enrique Cabello for his apportions to this thesis.

I offer my gratitude to the Artificial Vision Master professors who have shared their knowledge and have helped to complete the Master.

I would like to share my gratitude to Sara Tascon for her support, effort and English knowledge.

Last but not the least, I would like to thank my family and friends for the patience, cheering me up and for supporting me spiritually throughout the master, the thesis research, writing this document and my life in general.

Contents

Chapter 1

Introduction

In this chapter, the reasons that have led to developing this work are exposed. Furthermore, the objectives of this thesis and the structure of the document are also described.

1.1 Motivation

Artificial intelligence (AI) is a remarkably researched field which in recent years is gaining importance and is rapidly expanding. Due to my interest in AI, I decided to study an Artificial Vision Master having as a main motivation learning about Machine Learning, more specifically, Deep Learning.

How bio-inspired algorithms (as Artificial Neural Networks) learn from the input data and classify with a magnificent accuracy the testing samples or how Neural Networks are able to create music and pictures impressed me and it was a field of learning that I wished to discover in more depth.

The Human body as means of identification, hence as a key, is a fast and private identification and recognition system. Biometrics is a technology that is currently being implemented in our society and its presence is increasing. At the same time that biometric systems increase its potential, attacks are more and more powerful. From the acquired knowledge throughout the Master, I would like to expand them to be able to contribute preventing attacks or improving the security of biometrics systems. Applied biometrics has inspired my interest.

Connecting these two technologies, biometrics and Deep learning, is an opportunity to deepen both interesting fields. The Anti-spoofing with CNN researched works began a few years ago. Developing this thesis could contribute to this investigation.

FRAV group is currently working on the *ABC4EU* European Project, where face

biometric is utilized. Learning from a project that is currently being developed with technology that interested me, was a favorable and beneficent opportunity. Whatsmore, developing a thesis with them and with biometrics and Deep Learning has been an excellent opportunity that has greatly encouraged me.

1.2 Objectives and contributions

Objectives

The main objective of this thesis is being able to detect face spoofing attacks made on images with Deep Learning. These images are taken from genuine users (real users) and from people trying to impersonate (attacks) with printed images, masks or images shown on devices such as smartphones or tablets.

the second objective is obtaining basic theoretical Deep Learning fundamentals knowledge the same way as learning Theano, the main Deep Learning Python library which is going to be used.

The following objective is to build a Convolutional Neural Network architecture which is tailored to this thesis and is, therefore, used to extract the features of the images from the databases.

The next objective is working with different classifiers and dimensionality reduction algorithms and selecting the best one from the results obtained.

Afterwards, the objective would be selecting the most precise database out of the compared and analyzed results.

Finally, the last objective is finding a pattern among the incorrectly classified samples in order to know if physical attributes contribute to the wrong classification.

Contributions

This thesis contributes with a Convolutional Neural Network architecture and a classifier which optimizes the classification.

Also, a study among the different classifiers used and dimensionality reduction algorithms are presented for each database.

The next contribution is a discussion among the results obtained in each database in order to know which database is best, hence, whose samples are the most correctly classified.

1.3 Thesis structure

This thesis is constituted by a brief introduction, presented in chapter 1, where the motivation to carry out this project has been exposed and the main objectives of this thesis are described.

With the purpose of knowing the advances in the area of this thesis, a literature review is detailed in chapter 2.

Prior to the development of the main part, in chapter 3, a short introduction to neural network and anti-spoofing definitions are presented.

In chapter 4, the methodology is explained and the programming language, the databases, classifiers and metrics used in the document are detailed.

Experiments that have been carried out are specified in chapter 5, as well as the results obtained.

In chapter 6, the results obtained along the experiments are discussed.

In the final chapter 7, the conclusions obtained from the thesis are explained and the future work is detailed.

Chapter 2

Literature Review

In this chapter, the researched work in the same field as this thesis is summarized.

2.1 Related Works

Anti-spoofing is an ample investigation field, and concretely, face anti-spoofing is one of the most investigated next to fingerprint biometrics.

The investigation carried out thus far about anti-spoofing and Deep Learning is analyzed next with the purpose of continuing the investigation in this matter while taking into account previous works.

In [1], a Convolutional Neural Network architecture is built for face anti-spoofing purposes. The databases that authors use are CASIA and REPLAY-ATTACK face anti-spoofing datasets. Input data is pre-processed, faces are located and cropped with different spatial sizes with the aim of investigating if the background affects the classification. The convolutional neural network is trained with a sequence of frames and with single images in order to know if temporal features obtained from frames are helpful for face anti-spoofing. In order to classify CNN output features, SVM is utilized. Authors conclude that a background in images is necessary and by using a sequence of frames, the performance obtained is better than the results obtained with single images. Authors assert that in spite of not carefully selecting the parameters of the CNN, the results obtained are successful.

In [2] a Long Short Term Memory combined with a Convolutional Neural Network (LSTM-CNN) and a single Convolutional Neural Network are used for face anti-spoofing. Authors use LSTM with the purpose of providing memory to the system to learn temporal features. Experiments are made on the CASIA database, hence, authors use video samples to train and test the network. Input samples are pre-processed, faces are located

and different scales are used to study the background consequences. Results show that the LSTM has a better performance than single CNN architecture. Moreover, authors conclude that the background information is essential to detect face anti-spoofing.

The Convolutional Neural Network architectures used in [1, 2] are based on *Imagenet* [3], this is an extensively used architecture whose layers are defined.

A siamese architecture is developed in [4], the siamese architecture is composed by two identical Convolutional Neural network architectures. The purpose of this research is the spoofing identification in verification tasks. AT&T is face database used in this research work, it is formed by pairs of singles images (genuine and impostor) which are used to feed the network. Authors realize an exhaustive search of the best CNN parameters. Results obtained are competitive, yet the performance could be improved.

Deep Learning spoofing researched is not as broad as the textured-based researched work: LBP, HOG or DoG algorithms are feature extractors. In [5] authors study Local Binary Patterns (LBP) and variants of this method for face anti-spoofing.

Motion-based is also widely investigated for face anti-spoofing: the lip movement, head rotation or blinking eyes is detected and studied for spoofing detection. In [6] authors use the blink eyes movement.

Methods based on image quality analysis are also researched, such as the study of image distortion for face anti-spoofing in [7].

Chapter 3

Background Theory

In this chapter, the theoretical basis of anti-spoofing and convolutional neural networks are exposed.

3.1 Anti-spoofing and biometrics systems

At the same time that technology has evolved, capture systems and processing algorithms have proceeded too. This opens up a wide variety of options, one of them being the capability of developing biometrics systems, giving way to identification and recognition systems.

A biometric system is a software and hardware system which is based on human physical characteristics or psychological behavior in order to identify a person.

3.1.1 Historical introduction

It was in the 1870s when the necessity of identifying people by getting physical characteristics appeared. Alphonse Bertillon had the desire to identify jail prisoners, and in order to satisfy this need, the skull diameter, arm and foot length were used in the USA until the 1920s [8].

It was back in the 1880s when fingerprint and facial identification were proposed. With the appearance of digital signals processing systems in 1960, voice and fingerprint biometric systems began to be investigated and researchers started to have the idea of using this system to identify people in access control security [8].

Ten years later, the geometry of the hand commenced to be an area of interest for automated identification technologies. The retina and signature verification appeared in

the 80s and after a short period of time, the face systems appeared too [8].

The last biometrics systems appeared in the 1990s with the iris recognition [8].

3.1.2 Biometrics

Biometrics refers to characteristics that humans own. These characteristics should be inherent and are exclusive to each human. Because of this, they are used for identification purposes.

Biometrics could be distinguished as physical or behavioral biometrics. Physical biometrics are characteristics which every human is born with and trey are purelt genetic, fingerprint, face, DNA, ear and iris are physical biometrics. Behavioral biometrics are characteristics that a person has developed, they are psychological characteristics, signature and gait are behavioral biometrics [9].

Nowadays, the quantity of biometrics is considerable. There is not a biometric which is only used or could be selected as optimal, although fingerprints are the most popular biometrics [10]. The characteristic of each biometrics makes it appropriate for each particular application [8].

3.1.3 Biometric system

A biometric system could be defined as a structure which collects specific biometric data from a user. The input data is processed in order to obtain features which are compared with template samples. A biometric system could be labeled as a recognition system or verification system depending on the task carried out [11]:

Verification system: user claims an identity and the system validates the identity comparing the acquired data with the data stored in the template database. It is a one-to-one comparison.

Recognition or Identification system: user data is compared with the all the template database to find the user's identity. The identity is not claimed by the user and is given by the system according to the features. It is a one-to-all comparison.

Figure 3.1 (figure obtained from [11]) briefly represents the process of the acquired data until it is compared with the system database to see if it is a verification or recognition system. The modules shown in figure 3.1 are described [11]:

Sensor: the first module is the sensor, which obtains the user's biometric. Depending on the biometrics, the sensor can vary greatly.

Feature extractor: input data is processed to obtain the features. The features that are extracted in this module should be the same as the one stored in the database.

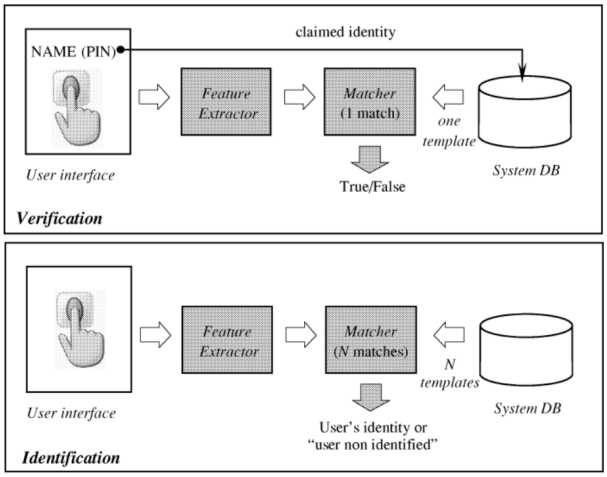


Figure 3.1: Verification and Identification system diagram. Image obtained from [11].

Matcher: features obtained from the previous module are compared or classified with the templates database so that the identity of the user could be verified or assigned.

Database: the database is made by templates which are used as true samples to compare the ones obtained with the sensors. A user should be registered to the system contributing with its biometric template.

Some biometrics could be used at the same time, in the same biometrics system, by adding the biometrics in the feature or score level. It is labeled as multimodal and it is used to complement vulnerabilities of biometrics [12].

3.1.4 Spoofing

At the same time that biometrics authentication systems appeared, the attacks to trick systems also emerged and they were called *spoofing attacks*. For instance,making plaster molds for geometric hand biometrics or fingerprints would be an attack just like the one presented in figure 3.2 (image obtained from [13]).

Spoofing is referenced to impersonate a biometric system trying to be verified or to get an identity when the user is not a genuine. The attack would depend on the biometry and the capturing system [12]. Attacks are usually made with artificial articles.

In order to identify the tricks and putting an end to spoofing attacks, *anti-spoofing* tries to minimize the attacks or prevent them.

Anti-spoofing methods could be distinguished depending on the biometric system module which is combined [12]:

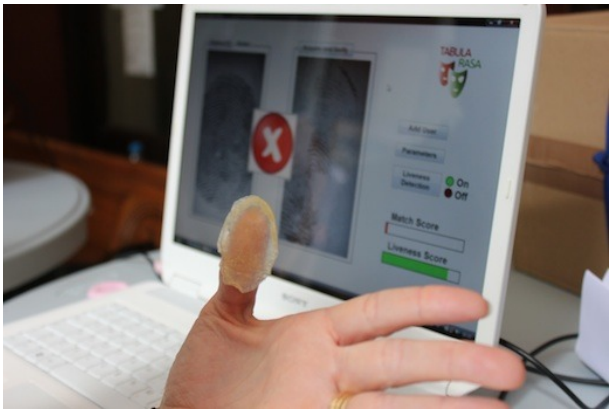


Figure 3.2: Spoofing fingerprint. Image obtained from [13].

Sensor level: extra equipment is added to the sensor, the new device is responsible for getting a living attribute as sweat or blood pressure.

Feature level: anti-spoofing detection is made after the sample has been acquired. The sample is processed and the software decides if it is a genuine or fake user.

Score level: after the feature extraction, when the scores are obtained, the biometric system decides if the user is genuine or not fusing methods. This method is the newest.

Anti-spoofing systems and scenarios could be evaluated. Depending on how they are evaluated, two methodologies are distinguished [12]:

Algorithm-based or technology evaluation: Evaluates algorithms that prevent anti-spoofing.

System-based or scenario evaluation: Evaluates the entire acquisition systems including the sensor.

In this project, an algorithm is going to be developed so as to detect anti-spoofing attacks when the face is used as biometrics. Given an image, the system has to decide if it is an attack or a genuine user. It would be a verification task.

3.1.5 Face anti-spoofing

Face biometrics is a frequently used biometric system because of its attributes since it is not intrusive, it is easy capturing images and it is considerably accepted by people as biometrics.

The principal or main used acquire systems are visible light cameras, apart from, other cameras as infra-red or depth camera that can also be used too. Moreover, using different types of cameras could help to detect anti-spoofing attacks.

The disadvantages of cameras as sensors are that the illumination needs to be controlled and images from different angles may not always be be suitable. In addition, peoples expressions or face occlusions are complicated to work with [10, 14]. On the contrary, the main advantage of using a visible light camera as sensor is the cost, which is affordable.

The location of verification or identification systems could be in a static and specific place as the office entrance or a wide place as a subway or an airport [14].

The spoofing attacks could be 3D attacks, if 3D masks are used; printed photos; 2D mask and displaying an image or video on a smartphone or tablet can also be considered 2D attacks [10]. The cost of 2D attacks is not high [15] and less expensive than 3D spoofing attacks.

Figure 3.3: 3D face masks. Image obtained from [16].

In figure 3.3 (image obtained from [16]) 3D resin face masks are represented. These masks are used for a 3D face mask database, but are an example of face spoofing attacks. A 3D mask could be made of silicone too and is more economical than a resin mask.

A 2D mask, printed photos or images and videos displayed on a smart device are attacks that easier to obtain thanks to the accessibility to personal information in social networks or the easy purchase of visible light cameras.

To discuss about spoofing attacks, the feature level is the most studied method. Literature separates it into two different groups: the dynamic and the static approach [12]:

Dynamic: it is based on motion to detect anti-spoofing. blinking eyes or optical flow assessment for example. Therefore, a temporal sequence of images are needed. It

could be efficient when image attacks are used, but not too accurate when videos are used.

Static: a unique image is analyzed.

Different techniques to decide if a user is being impersonated or not are being researched and could be used with image and video sequences [12].

Texture-based methods are one of the most investigated. Local Binary Patterns (LBP), Histogram of Oriented Gradients (HOG) or Difference of Gaussian (DoG) are algorithms used to extract images features [12, 15].

Motion or liveness detection methods search for movement in videos, asking the user to make a movement or analyzing the video frames sequence. The liveness could be detected in the blinking of the eyes, lip movement or head rotation [12, 15].

Others methods, as multispectral-based anti-spoofing, use images obtained from an alternative to visible light cameras such as infra-red cameras [15].

Generally, the explained methods work with 2D face anti-spoofing. However, 3D anti-spoofing is present too used as sensor depth cameras [10]. This thesis is focused on 2D face anti-spoofing.

3.2 Neural Network background theory

In this section, the basis and the theoretical background of the convolutional neural networks theory are presented.

3.2.1 Historical introduction

Neural Networks are introduced for the first time in the 40's by Warren McCulloch and Walter Pitts, more specifically, it was simple models of neural networks (switches based on neurons) that were able to calculate almost every arithmetic problem and logic operations. A few years later, the recognition field of spatial patterns was defined as an interesting scope for neural networks [17].

In 1949 the 'Hebbian rule' was postulated by Donald O. Hebb. The rule describes the general learning basis of neural networks. This rule explains that two connected neurons have a bigger strength if they are active at the same time, and the strength changes proportionally to the product of the two activities [17].

From 1951, the golden age of neural networks initialized. The first neurocomputer, which was capable of adjusting the weights by itself, was developed in 1951 and was called *Snark*. The neurocomputer which was able to recognize simple numbers was called *'Mark I perceptron'* and was developed between 1957 and 1958 [17].

It was in 1959 when Franck Rosenblatt defined the *perceptron*; the *perceptron convergence theorem* was verified. One year later, the *ADALINE (ADAptive LInear NEuron)* was developed, being the first commercially used neural network, a fast and precise system. One important characteristic was the *delta rule* rule used for the training procedure. Before the golden age finished, researchers figured out that the XOR function was not able to be solved with just one perceptron [17].

After 1960, the neural networks golden age finished and the importance of this researched field decreased until computational resources were enough. However, some important advances were developed such as the *linear associator* model (an associative memory model). The backpropagation of error is a very used learning procedure that was defined in 1974 by Paul Werbos. *The self-organizing feature maps* were described in 1982 alternatively known as Kohonen maps. In 1983, a neural model able to recognize handwritten characters was developed as an extension of the Cognitron, the new model was called Neocognitron [17].

In recent years, neural networks are having a significant importance and are researched widely and Computational resources are more capacious and consequently, the exploration of this area is faster, more accurate and more extensive.

3.2.2 Introduction to ANN

Humans, along the history of time, have tried to reproduce nature. Evolution has turned out to be a big coordination in nature. Object recognition, associating concepts, memorizing or extracting the semantic of images are competences that humans are capable of. These processing information tasks are being investigated and it is now when technology is being more accurate, not as precise as humans skills.

Neural Networks are an important Artificial Intelligence (AI) subject and a sophisticated information processor which is inspired in the information processing in the brain [18].

3.2.3 Biological Neural Networks

Biological neural networks are part of the nervous system and are formed by neurons units or nervous cells. Each neuron is capable of processing information in different ways

by itself [18].

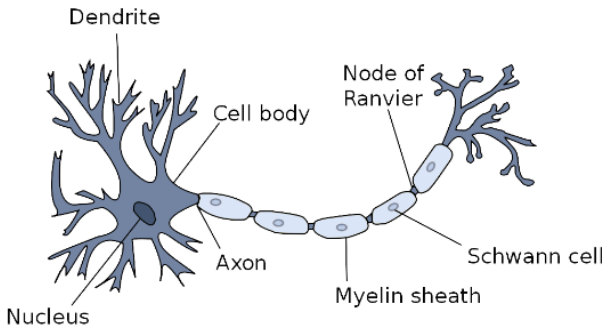


Figure 3.4: Biological Neural Network. Image obtained from [17].

The principal components of a biological neuron are the nucleus, dendrite, cell body, axon, Schwann cell, the node of Ranvier, etc. In figure 3.4, a biological neuron is illustrated and its components are signalized [17].

The soma is the spherical central part of the nerve cell in which there is salt and potassium concentration which is covered by the neuronal membrane. Inside the soma we can find the neuronal nucleus and from the soma, branches extend from it(dendrites). Nerve cells are connected among each other by dendrites. Neurons are continuously transmitting and receiving nervous signals, communicating among them. The information transference is produced, more specifically, in the synaptic clef (space between the connection of two neurons). This exchange of information is denominated synapses and it is made through electrochemical activities. The axon is a soma extension whose responsibility is transmitting the information electrochemical out of the neuron by the nervous system thanks to its terminal branches [17, 19].

3.2.4 Artificial Neural Networks (ANN)

Just as the nervous systems is formed by neurons, artificial neural networks are composed of artificial neurons. Each artificial neuron is a processing unit whose input is processed and shared to another neuron or to the output. Neurons are connected among them [17]

To sum up, there are three groups of artificial neurons:

Input neurons, if they belong to the input layer. These neurons receive the input data of the network.

Hidden Units, if they belong to the processing layers. There are many types of layers: convolutional, pooling, dropout, etc. There could be as many hidden layers and

hidden units as the user desires. The connectivity and the topology of the layers define the topology of the network.

Output neurons, if they belong to the output layer. These neurons give the user the processed information.

In figure 3.5 the schematic of a general neural network is possible to visualize the schematic of a general neural network with an undefined number of neurons in each layer and an undefined number of hidden layers.

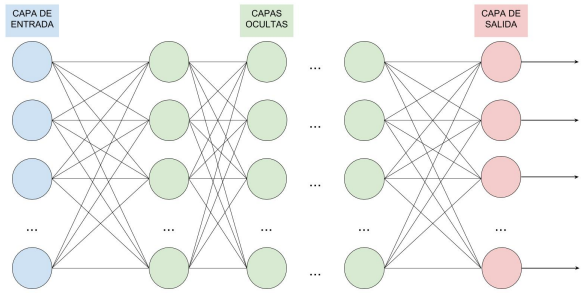


Figure 3.5: Schematic of a general neural network.

A single neuron is formed by an input, a weight W and an associated bias b (independent term). The value of the bias is always 1, but it has an associated weight that makes the value of the bias change. Weight and bias values could be modified. The output of the neuron is associated to an activation function.

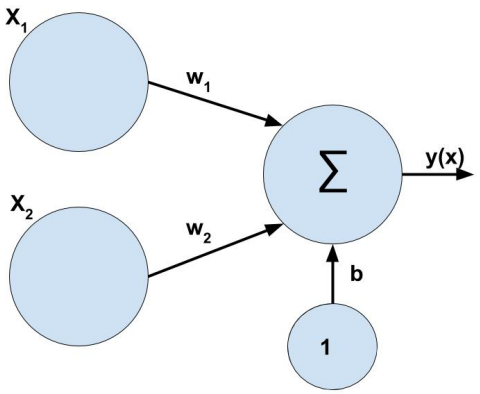


Figure 3.6: Simplest neural network architecture.

The simplest example of a neural network is formed by two input neurons and an out-

put neuron as demonstrated in figure 3.6in which $W = [w_1, w_2]$ are referred to the weights assigned to each neuron. The b value is referred to the bias term. $X = [x_1, x_2]$ are the input of the network architecture and *y(x)* the output. The output is predetermined by the equation 3.1. So, the output would depend on the input, weights and bias send the activation function (or transfer function) F [20].

$$y(x) = F(\sum_{i=1}^{2}(w_i * x_i + b)) \tag{3.1}$$

Due to the activation function, the output of the neuron would change if the input is bigger than a defined threshold. This threshold is defined by the own activation function. There are various functions that are used: sigmoid function, Heaviside function, Fermi function or hyperbolic tangent among others. In figure 3.7 (which has been obtained from [17]) the quoted functions are portrayed. The output would be 0 or 1 values if the function is sigmoid or Fermi and the output would be -1 or 1 if the function is a hyperbolic tangent.

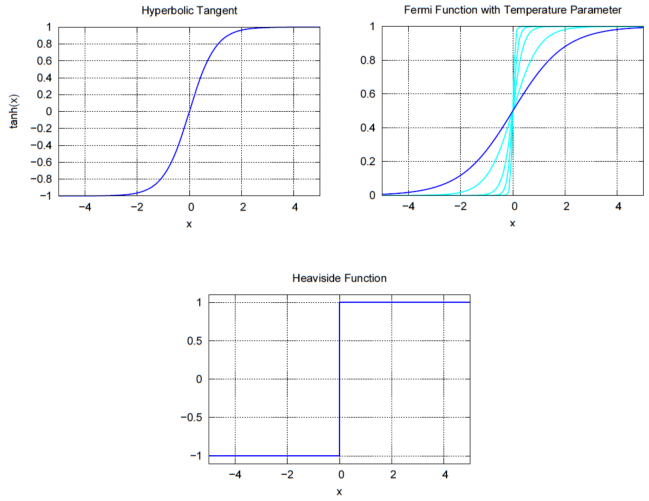


Figure 3.7: Different activation functions. Image obtained from [17].

Analogies between ANN and Biological Neural Networks

Due to the fact that Artificial Neural Networks are based on Biological Neural Networks, analogies remain.

The most visible discernible analogy is that which correspond yo the biological neuron and the artificial neuron. The cell body correspond with the transference function. The output to other neurons would correlate with the axon and synapses with weights and bias. The soma would correspond with the transfer function. Those analogies are summarized in table 3.1 [21]. Artificial neural networks are inspired in Biological neural networks, and thus both work in a different (although similar) way.

Artificial Neural Component	**Analogy**
Neuron Artificial Neuron	Biological Neuron
Transfer function	Soma
Connexion between Artificial neurons	Axon
Weight	Synapses

Table 3.1: Analogies between artificial neural networks and biological neural networks [21].

Learning

Neuronal networks are able to learn features from the input to classify or with the purpose of extracting features from the input data. In order to know how to get the desired output, a learning process is needed. During the learning process weights are modified, as well as bias values, for each layer with the purpose of getting a better output [22].

There are three main different learning procedures [22, 23]:

Supervised Learning: the input training data is given with its target (the correct result of its corresponding sample), that means that the training samples has associated with the desired output. The learning consists in adjusting the weights and bias until the output of the network is the same or the closest value to the target.

Reinforcement Learning: in the training, the correct target is not provided, but a grade is given to the network. Weights and bias are updated regarding the grade.

Unsupervised Learning: input data is constituted by samples, targets are not provided. Network clusters the data with its own criteria modifying weights and bias values.

What Neural Network learning procedures have in common is the modification of weights and bias to learn and obtain the desired output, hence, the error at the output is minimized. The most useful learning procedure is supervised learning.

The gradient descent procedure is an algorithm whose goal is to minimize the error of a function *J*. *J(a)* would be the minimum if a is the solution. To get the solution the gradient of J ($\nabla J(a)$) is calculated for each value of a: $\nabla J(a(1))$ is obtained and *a(2)* is

obtained moving some distance from *a(2)*, in steps called *learning rate* η, and in direction of the negative gradient. The *a(k+1)* value would be: [23].

$$a(k+1) = a(k) - \eta(k)\nabla J(a(k)) \tag{3.2}$$

One of the most used supervised techniques of neural network learning is the back-propagation, this learning rule is based on the gradient descent. Most generally with this method, when given a random initialization of weights, these begin to update or change in the gradient descent direction:

$$\delta w = -\eta \frac{\partial J}{\partial w} \tag{3.3}$$

Being $\frac{\partial J}{\partial w}$ the gradient of J in function of weights w.

The error is calculated as defined in equation 3.2.4

$$E = \sum_p E^p = \frac{1}{2}\sum_p (\delta^p - y^p)^2 \tag{3.4}$$

The error is backpropagated from the last layers until the first layer, modifying the weights to balance the error proportionally to the gradient of the error function. This is called the chain rule or the delta rule [17,20,23]. The backpropagation equation is defined in equation 3.5:

$$\Delta_p W_{jk} = eta\delta_k^p y_j^p \tag{3.5}$$

Being k the unit which receives the input and j the output.

There are three useful training protocols according to the use of the training subset:

Stochastic training: samples are selected randomly and for each sample, weights are updated.

Batch training: all training samples are used to the learning process.

On-line training: there is no memory, at the same time that samples are received, they are used once to train.

Type of Layers

The layers that make up the networks could be different depending on the mathematical operation. The most common used layers are described below:

Convolutional layer: the convolution operator is processed at the input. Weights are the applying filters and the output of the network is a feature vector. In one convolutional layer, the number of filters is defined by the user.

Pooling layer: in this layer, the dimensionality is reduced. The most important information is preserved. It is usually used at the output of the convolutional layer [24]. The most used pooling layer is the max-pooling layer, the maximum values are saved.

Normalization layer: this layer normalizes the activities of the neurons, and thus, the processing time could be reduced during training or testing. There are two types of normalization layers: The Local Response Normalization (LRN) and the Batch Normalization.

Dropout layer: the output of a network could rely on the output of an specific neuron being this an overfitting cause. To prevent it, during training, some neurons are 'turned off setting its value to 0. Thus, neurons are more adaptable. Alternatively, instead of *eliminate* neurons, weights could be set to 0, in this case, the layer would be **"DropConnect"** [24].

Fully-connected layer: the fully connected layer is a vector of neurons where the input is connected to each neuron that conforms this layer. This layer is the high-level reasoning layer and therefore, it is used prior to the classification.

One neural network is defined by its type of layers and the number of neurons. Furthermore, there are parameters that affect the network behavior (learning rate, etc.) [25].

3.2.5 Convolutional Neural Network (CNN)

Convolutional Neural Networks are a determined typology of neural network, CNN are inspired in how the visual cortex of a cat works [24]. This type of neural networks are used with images at the input of the network.

This specific neural networks is able to learn features from training images, therefore CNN have been successfully used in recognition and classification tasks such as document and object recognition, face detection or robotics navigation among others [25, 26].

Chapter 4

Methodology

In this chapter, is explained the programming language used, the databases that are going to be trained and tested, the classifiers with which test samples are going to be tested and metrics used to compare results.

4.1 Programming language and frameworks

The programming language which has been used to develop the thesis is *Python*. *Python* is an object-oriented language and very used nowadays. The version used is the 2.7.

Python libraries have been used to help to implement the code. The main framework used is *Theano*. *Theano* has been used to build the convolutional neural network and its training procedure because of the possibility that offers of working with symbolic variables, mathematical expressions and multidimensional arrays. *Theano* documentation could be found in http://deeplearning.net/software/theano/.

Scikit-learn is another *Python* library which has been used as the basis for building the classifiers and some of the metrics. Its documentation it is available in the following url https://docs.scipy.org/doc/numpy/.

Others libraries such as *NumPy* or *Matplotlib* have been needed. *NumPy* is a library that allows users to works with multidimensional arrays or random simulations among others qualities. The documentation of this framework is available in https://docs.scipy.org/doc/numpy/. *Matplotlib* has been utilized as graphic tool to plot figures.

4.2 Databases

In this section the databases that are used along the thesis are described in this section.

One database has utilized to learn *Theano* and three face databases have been used in order to detect anti-spoofing. The face anti-spoofing databases have been generated with face anti-spoofing finality and has been used previously for the same purpose.

All the databases are formed by three subsets whose samples are not repeated among subsets:

Training subset: is used to train the network during epochs.

Validation subset: is used to check the performance of the network while is training. The validation subset is usually used when hyper-parameters are calculated.

Test subset: is used just at the end of the training process. The best model is chosen with regard to the best validation error. This subset should not be used until the network architecture and classifiers parameters are selected.

4.2.1 MNIST digit database

MNIST digit database is an image database of human written digits. This database is frequently used to learn machine learning techniques. Because of that, the database has been used, in this thesis, for learning Theano and convolutional neural networks. In addition, this database has been used in a implemented convolutional neural network (LeNet).

Some examples of the digit image MNIST database could be seen in 4.1, image obtained

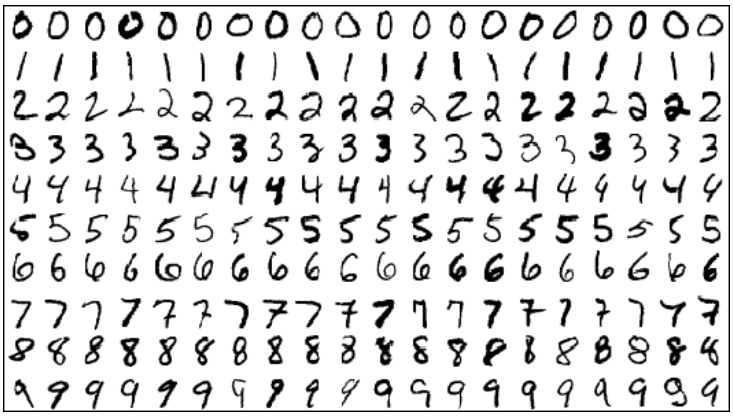


Figure 4.1: MNIST digit images database. Image obtained from [27].

from [27] and the characteristics of this database are the following ones:

- There are 70.000 number of unique samples.
- Despite the original size of the database is 32x32 pixels, the samples of this downloaded database are 28x28 pixels in gray scale, that is 784 features per image.
- 10 classes could be differentiated, one per digit.

– The samples are directly separated into train, test and validate subset.

4.2.2 FRAV dataset

FRAV database is an anti-spoofing face database built in the *FRAV* research group of the *URJC University* and which is part of the Automated Border Control Gates for Europe project [28].

FRAV database is formed by genuine users and spoofing attacks. Moreover, this database is formed by RGB images and NIR images, almost each RGB sample has been captured with infrared camera too.

Both databases are composed by genuine users samples and four distinctive attacks: printed photo, 2D mask, 2D mask with eyes cropped and real user image displayed on a tablet device.

– Original images of people represented in figure 4.2(d) and figure 4.3(e) for RBG images and figure 4.3(j) for NIR image.
– Images of people printed (attack) represented in figure 4.2(a) and figure 4.3(a) for RBG images and figure 4.3(f) for NIR image.
– Images of people with a mask (attack)represented in figure 4.2(b)and figure 4.3(b)for RBG images and figure 4.3(g) for NIR image.
– Images of people with a mask with the eyes cropped (attack)represented in figure 4.2(c) and figure 4.3(c) for RBG images and figure 4.3(h) for NIR image.
– Images of people in a tablet (attack)represented in figure 4.2(d) and figure 4.3(d) for RBG images and figure 4.3(i) for NIR image.

The characteristics of this database are described:

– There are 939 people in each RGB class and 195 in each NIR class.
– There are 2 or 5 classes: genuine class and four attacks that could be put in the same class or not.
– Each user has a genuine image and four attacks.
– There is one image per person.
– Each image has its own shape.
– The faces are centered in the image.
– RGB images have been obtained with a professional visible light camera and NIR images with a Near infrared camera.
– RGB images are in RGB space and NIR images are in gray-scale space.

FRAV RGB database

A representation of RGB database is shown in figure 4.2, where genuine user (e) and attacks (a-d) from the same user are exposed.

(a) printed image attack (b) mask attack (c) eye mask attack (d) tablet attack (e) real user

Figure 4.2: Four attacks and real user from RGB FRAV database.

FRAV RGB+NIR database

This database is smaller that RGB database because not all users have its corresponding NIR image. A representation of RGB database is presented in figure 4.3, where each RGB image has its corresponding NIR image.

When RGB and NIR (figure 4.3)images are used at the same time, two different methods, with the finality of using both types of images together, are determined:

- Adding images in characteristic level: adding the NIR image as another layer to RGB image, so the resultant image have hightxweightx4 dimensions (NIR images has one layer because it is a gray scale image and RGB images have three layers, one per each primary color). The network is feed with the resultant images like other times.
- Adding images in classification level: after the network training and before feeding the classifier, RGB and NIR databases would be trained separately and its features would be appended as the input of the classifier.

To conclude, this database is going to be used in the three different ways: just RGB images, RGB and NIR images added in characteristic level or classification level.

When the classes are built, two ways are possible to be done: the first one where real people are one class (positive) and the different attacks are another class (negatives), so

(a) printed RGB image attack (b) mask attack RGB (c) eye mask attack RGB (d) tablet attack RGB (e) real user RGB

(f) printed NIR image attack (g) mask attack NIR (h) eye mask attack NIR (i) tablet attack NIR (j) real user NIR

Figure 4.3: Four attacks and real user of RGB and NIR FRAV database.

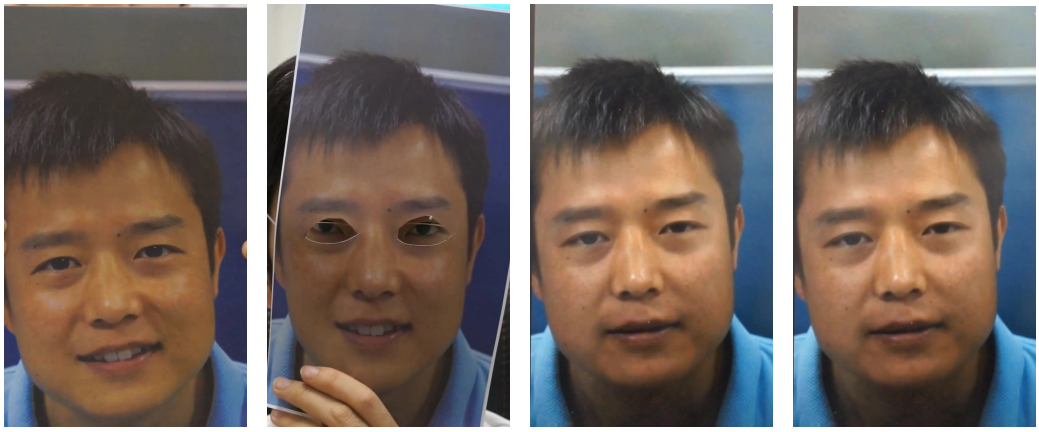
two classes have been used; and the second way where each attack correspond with a class, so five classes (4 attacks and 1 real) have been utilized.

4.2.3 CASIA dataset

The CASIA Face Anti-Spoofing database is a database from the Chinese Academy of Sciences Centre for Biometrics and Security Research (CASIA-CBSR) [29].

CASIA database is formed by real or genuine images of people and three different attacks of the same people:

- Images of people printed (attack) represented in figure 4.4(a).
- Images of people with a mask with the eyes cropped (attack) represented in figure 4.4(b).
- Tablet attack represented in figure 4.4(c)
- Images of real users represented in figure 4.4(d).



(a) printed image attack (b) eye printed image attack (c) tablet attack (d) real user

Figure 4.4: Three attacks and real user from casia database.

Originally, this database is a video database, in which each sample is a different video, but for the experiments developed no videos (entirely or fed directly to the network) have been used. The CASIA database has been used in two different ways:

- Using a single image per person and class. When this database is used, it is going to be referred as CASIA image database.
- Reading three frames per video and saving each frame as an independent image sample. This database is going to be referred as CASIA video database.

The characteristics of the CASIA image database are the following ones:

- There are 49 images per user, so there are 196 unique samples.
- Samples do not have the same size.
- Samples are in RGB space.
- The face of the image is centered.

The characteristics of the CASIA video database are the following ones:

- There are 8 videos per person (two videos for real user, and two per each attack).
- There are two videos because one is filmed horizontally and the other vertically, one filmed with a smartphone and the other with the frontal camera of a laptop.
- There are 50 different users, so there are 400 different videos.
- For each video 3 frames are read, so there are 1200 unique samples.
- Samples are in RGB space.
- Faces are centered in the image and blink expression and movement of people are produced.

At the time of assigning a class, it could be done using two classes, the positive class to the real users and the negative class to the attacks; If each attack is assigned to an independent class, it would be four different classes, the real user class and three attack classes.

This database has been used in different articles [1, 2, 7, 12], it is a very utilized databased for face biometrics.

4.2.4 MSU-MFSD database

The MSU Mobile Face Spoofing Database (MSU-MFSD) is a video face anti-spoofing database [7].

In figure 4.5 are represented the three attacks and a real user which forms this database:

- Printed photo attack represented in figure 4.5(a).
- Tablet (iPad Air) attack where a video is Replayed represented in figure 4.5(a).
- Smartphone (iPhone 5s) attack represented in figure 4.5(a).
- Real user represented in figure 4.5(a).

(a) printed image attack (b) tablet attack (c) smartphone attack (d) real user

Figure 4.5: Three attacks and real user from a person of MFSD database.

Originally, the database is a video database, but only one frame per user and class is used. The characteristics of the database are the following ones:

- There are 35 images per attack or genuine user. There are 140 unique samples.
- Images are in RGB space.
- Faces are centered in images.
- The size of each image is not equal. Approximately images are 300 pixel height and 335 pixel width.

– Images have been acquired with a Macbook Air laptop and a Google Nexus4 smartphone.

4.3 Classifiers, Reduction of the dimensionality algorithms and Cross Validation

Classifying is called to the task of assign a category to an object. The classification task is based in the obtained features of the object and the characteristics of the feature extractor [23]. The particular case of this thesis, features obtained at the output of a Convolutional Neural Network are utilized for classification.

The output of the convolutional neural network could be bigger enough and some features could not be relevant for the classification. To solve the speed and robustness issues that could appear because of the quantity of features [30], techniques to reduce the dimensionality are used.

Classifiers must be customized to each problem, to find the optimal parameters for each occasion, cross validation technique has been used.

4.3.1 Classifiers

In this section, the classifiers used along the thesis are described.

Logistic Regression

Logistic regression is a probabilistic and a linear classifier. It is customized by a weight matrix W and a bias vector b.

The logistic regression weights and bias define a linear hyperplane which is the decision boundary of the classes. In order to find the parameters, the Maximum likelihood estimation is used during training [31]:

$$\prod_{i=1}^{n} P(y_i|X_i, W, b) \tag{4.1}$$

Given an input vector x, which belongs to the i class (a value of a stochastic variable Y), its probability could be described as follows:

$$P(Y = i|x, W, b) = \frac{e^{W_i x + b_i}}{\sum_j e^{W_i x + b_i}} \tag{4.2}$$

The class of a new sample (y_pred) would be classified as:

$$y_{pred} = argmax_i P(y_i|X_i, W, b) \tag{4.3}$$

A sample would belong to a class depending on position in the space with respect to the hyperplane that separates the classes.

Support Vector Machine

Support Vector Machine (SVM) is a two-class (bi-class) classifier. The smallest generalization error is linked to the *margin* concept. Margin is the perpendicular distance between the closest sample of the database and the calculate hyperplane [32]. An hyperplane is optimal if the margin is the maximum and this margin is calculated (as the same way as logistic regression):

$$\arg\max_{wb} \left\{ \frac{1}{||W||} \min_n [t_n(W^T \phi(X_n) + b)] \right\} \tag{4.4}$$

Where w, b are the parameters that should be optimized in order to maximize the distance. t_n are the training samples. ϕ is a fixed feature-space transformation, b is the bias parameter.

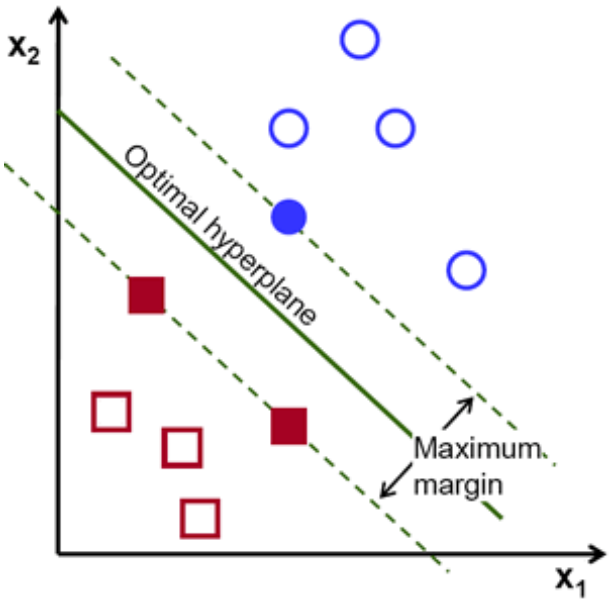


Figure 4.6: Optimal hyperplane and the decision boundary. Image obtained from [33].

In figure 4.6 the optimal hyperplane between two classes are represented with its corresponding margin. In the given example, the two classes are well differentiated. Figure 4.6 has been obtained from [33].

Based on estimate the hyperplane that maximize the distance between classes, the closest vectors of each class are selected [32,34]. In practice, the margin is determined by C, a parameter that should be chosen by user to get the optimal margin.

The SVM performance is join to a kernel function which allows variability in nonlinearity and flexibility in the model [31,35]. There are various kernels (polynomial, sigmoid, etc.), two of them are utilized:

1. Linear kernel: $K(x_i, x_j) = x_i^T x_j$.
2. Radial basis function (RBF) kernel: $K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2), \gamma > 0$

K Nearest Neighbours

K- Nearest Neighbour (KNN) is a generative and non parametric classifier. For classifying, the density estimation procedure is utilized. The difference between this classifier and others is that data is used in this algorithm directly for classification, without building a model first [31]. The density function is determined by the form [32]:

$$p(x) = \frac{K}{NV} \tag{4.5}$$

Where K is the number of points inside the region R whose volume is V and N is the number of total samples or observations.

This classifier uses the observation directly to classify and needs all the samples to predict a new one. The probability of a sample x belonging to a class C_k is defined by [32]:

$$p(x|C_k) = \frac{K_k}{N_k V} \tag{4.6}$$

Where N_k are the observations of a class C_k and K_k of it class points are contained in the volume V.

The K value is fixed and constant, it should be calculated and optimized by user of each application.

Decision Tree

Decision Tree classifier is based in a natural classification based in a sequence of true/false or yes/no questions [23]. It could be used as a binary classifier or a k classes classifier.

The input data is split to maximize its separation, resulting a tree structure [31] as is described in figure 4.7 (image obtained from [36]). Where depending on the features, a sample changes from a principal branch to a branch of this until a class is signed. The last branches correspond to the classes and the same class could be in different final branches.

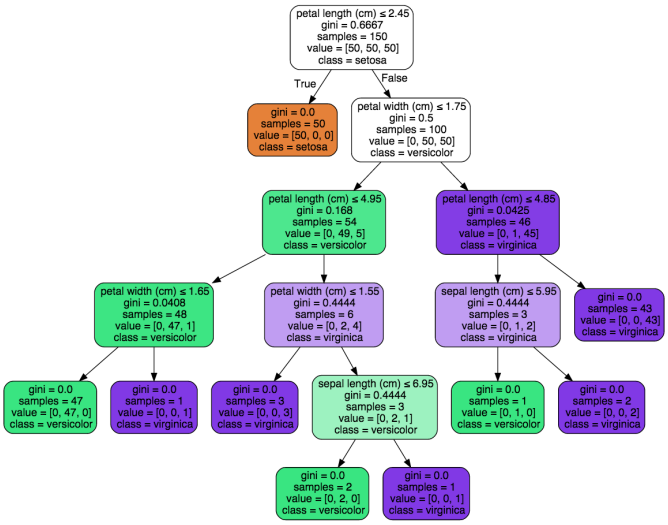


Figure 4.7: Decision Tree Classifier. Image obtained from [36].

4.3.2 Dimensionality reduction algorithms

The objective of those algorithms is transformed by the characteristic vector into another characteristic vector but with a lower dimensionality. Linear methods, that projects the dimensional data onto another space whose dimensionality is lower [23], have been used. The two techniques, the most common used, are described and used along the thesis.

Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) looks for the vectors in the space that best discriminate among classes; LDA pretends to maximize the between-class measure (equation 4.3.2) at the same time as the within-class measure (equation 4.3.2) is minimized [30].

$$S_w = \sum_{j=1}^{c} (\mu_j - \mu)(\mu_j - \mu)^T \tag{4.7}$$

$$S_w = \sum_{j=1}^{c} \sum_{i=1}^{N_j} (X_i^j - \mu_j)(X_i^j - \mu_j)^T \tag{4.8}$$

The data of d dimensions would be projected onto a s dimensions and being $s < d$. The minimum value that s could take would depend on the number of classes n: $s \geq n-1$.

Principal Component Analysis

Principal Component Analysis (PCA) uses a subspace t in which the the variance direction among basis vectors is maximum in the original space f [30]. PCA faces the problem of reducing the n dimensional samples vector to a single vector X_0. The X_0 vector would be the smallest result of the sum of the squared distances between X_0 and various features X_k [23].

The new subspace is usually smaller than the original space [30].

The linear transformation from one space into another would be denoted as W, and its columns are eigenvalues which has eigenvectors associated.

The feature vectors (y) from the f space would depend on W:

$$y_i = W^T X_i, i = 1, ..., N \tag{4.9}$$

Being N the total number of feature vectors.

4.3.3 Cross Validation

Classifiers are defined by certain parameters. For example,the number of neighbours (k) in KNN classifier is a value that user have to determine. In this case is useful use Cross Validation to determine the value of k.

This method uses the training samples. They are split in groups (*kfolds*) and used to train and test the classifier with different values; in the KNN case, the value k would be change. A metric (score) is calculated and it is possible to determine the value of the classifier in which the metric is the optimum.

This technique has been used to calculate the value which a classifier is defined. For SVM classifier the value C, for KNN classifier the value k, for Decision Tree classifier the depth of the tree, Softmax classifier to determine the learning rate when it has not been trained at the same time of the network and for PCA and LDA the number of components.

4.4 Metrics

To characterize a system, it is necessary metrics that evaluate it. In this section the used metrics along the thesis are exposed.

Before describing the parameters it is necessary define that the posed problem is bi-class, that means that only two classes would be used:

Positive class: are the samples of the real users, the genuine or *bona fide*.

Negative class: are the different attacks samples which that pretend to be real users but not.

4.4.1 Cost and Error rate

The first parameter that is used is the cost. The cost is used while the neural network is training, in fact, is the value that must be minimized during the training. The lower value, The better performance of the network.

The cost calculated with the Minibatch Stochastic Gradient Descent (MSGD) is the Negative Log-Likelihood Loss. The MSGD is a variant of the Stochastic Gradient Descent in which the cost is calculated with a mini batch of data, not each sample independently and the Loss is the accumulation [37].

The loss is calculated in the following way:

$$Loss(\theta, D) = -\sum_{i=0}^{|D|} logP(Y = y^{(i)}|x^{(i)}, \theta) \tag{4.10}$$

The error in the validation process is calculated after the logistic regression classification, because is the classifier used during the training process. The error during the testing procedure depends on the used classifier. In both cases, the error represents the number of misclassified samples over the total samples used.

4.4.2 True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN)

If each predicted class is compared with it real target, True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN) values could be calculated. These metrics are usually used for bi-classes problems.

Those metrics, are gotten when positive or negative samples are correctly or incorrectly classified [38]. The classified or predicted sample is compared with its real target.

If a positive sample is classified as positive is a true positive (TP), but if it has been classified as negative is a false negative (FN).

If a negative sample is classified as negative is a true negative (TN), but if it has been classified as positive, is a false positive (FP).

From those four metrics, it could be extracted the confusion matrix for binary classification which is defined in table 4.1 [38, 39]:

Real / Classified	**Positive**	**Negative**
Positve	TP	FN
Negative	FP	TN

Table 4.1: Confusion Matrix.

The confusion Matrix resume these four metrics in a table. Both, the confusion matrix or the parameters individually, are widely utilized.

4.4.3 ROC curve and Equal Error Rate (EER)

From the confusion matrix, it is possible calculate others parameters [38]: precision, recall, specificity, accuracy, etc. because its values depend on TP, TN, FP, FN:

- The False Positive Rate (FPR) is defined as the proportion of all the negative samples (N) that are classified as positive incorrectly [39]:

$$FPR = \frac{FP}{N} \tag{4.11}$$

  Where:

$$N = FP + TN \tag{4.12}$$

- The True Positive Rate (TPR) is defined as the proportion of all the positive samples (P) that are classified correctly [39]. This parameter could be known as Recall too:

$$TPR = Recall = \frac{TP}{P} \tag{4.13}$$

  Where:

$$P = TP + FN \tag{4.14}$$

- False Acceptance Rate (FAR): is defined as the incorrectly accepted users with respect all the samples.

$$FAR = \frac{FP}{P + N} \tag{4.15}$$

- False Rejection Rate (FRR): is defined as the incorrectly rejected users with respect all the samples.

$$FRR = \frac{FN}{P + N} \tag{4.16}$$

– Accuracy is defined as the proportion of the correctly classified samples of all the samples [38]. The classifiers implemented in scikit-learn library return this value as metric.:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4.17}$$

The Receiver Operator Characteristic (ROC) curve is the representation how the number of positives samples, which has been classified correctly, changes with the number of negative samples incorrectly classified. The ROC curve is defined by the parameters False Positive Rate (FPR) and True Positive Rate (TPR) [39].

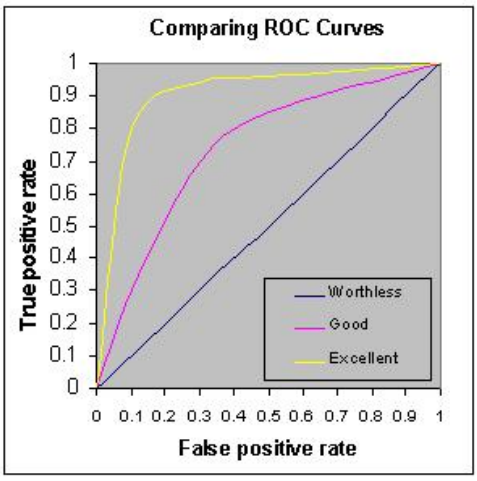


Figure 4.8: ROC curves. Image obtained from [40].

The figure 4.8, obtained from [40], demonstrates a ROC graph in which three curves are portrayed. The yellow one represents a good classification, it is the desired ROC curve, but the blue curve represents a bad classification and it is not the desired result.

From the ROC curve, the Area Under the Curve (AUC) could be obtained, this value is the integral of the ROC curve, and its maximum value is 1 that means a perfect performance of the classifier. If the value of this parameter is lower than 0.7 the classifier performance required to be improved significantly.

From FAR and FRR rates it is possible to calculate the Equal Error Rate (ERR) and it is obtained at the point where FAR and FRR acquire the same value. The closest is its value to 0, the better is the performance of the classifier.

4.4.4 APCER and BPCER

The ISO/IEC 30107-3 [41] is the collaboration result of the International Organization for Standardization (ISO) with the International Electrotechnical Commission (IEC).

The ISO defines the terms related to the tests, the reports and the biometric presentation of biometrics systems. In addition, the performance methods, specify principles as well as metrics are defined. From this document, the APCER, BPCER and APCER-BPCER curve metrics have been obtained:

Attack Presentation Classification Error Rate (APCER) is defined as the proportion of presentation attacks that has been classified incorrectly (as *bona fide* presentation.)

$$APCER_{PAIS} = \frac{1}{N_{PAIS}} \sum_{i=1}^{N_{PAIS}} (1 - Res_i) \tag{4.18}$$

Bona fide Presentation Classification Error Rate (BPCER) is defined as the proportion of *bona fide* presentations incorrectly classified as presentation attacks.

$$BPCER = \frac{\sum_{i=1}^{N_{BF}} Res_i}{N_{BF}} \tag{4.19}$$

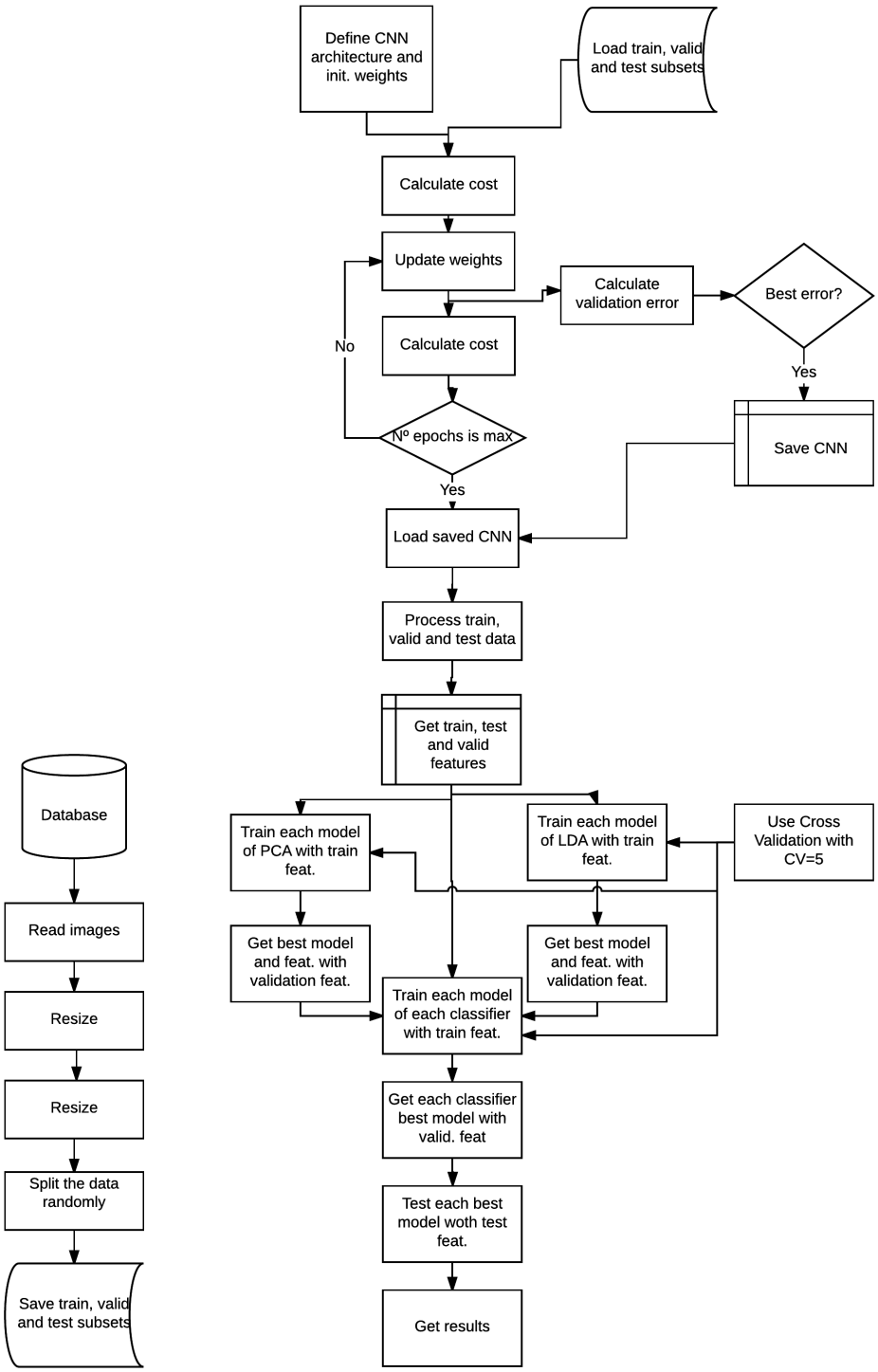
where:

- N_{BF} is the number of *bona fide* presentations
- Res_i is 1 if i^{th} presentation is classified as an attack and 0 if is classified as a *bona fie* presentation.
- N_{PAIS} is the number of attack presentations

The APCER-BPCER curve illustrates in the same graph both parameters. The ideal system would have a low APCER (0) and a low BPCER (0) because it means that samples are not incorrectly classified.

4.5 Algorithm diagram chart

The flowchart of the code used for this thesis development in section 5.2 is shown in 4.9. In the figure, the subsets generation flowchart from the databases is presented in 4.9(a); it has been used for each database (CASIA images, CASIA videos, RGB FRAV, RGB+NIR feature level FRAV, RGB+NIR classification level FRAV and MFSD-MSU). The programmed code is summarized in figure 4.9(b) which has been run, as described in the figure, as many times as databases are available.



(a) Subsets generator diagram

(b) General algorithm diagram

Figure 4.9: Code diagram chart.

Chapter 5

Experiments

In this chapter the evolution of the built Convolutional Neural Network architecture, the experiments carried out and its corresponding results, are described.

5.1 LeNet-5

LeNet-5 [42] is the name of a certain architecture of a convolutional network designed for document recognition (handwritten, machine printed characters) developed by Yan Lecun *et al.*

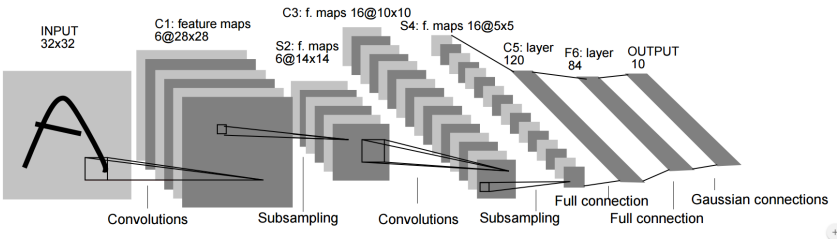


Figure 5.1: LeNet-5 Arquitecture.

The basic architecture of LeNet-5 is two convolutional layers, each one of them followed by a max pooling layer and then a fully-connected layer. This architecture could be visualized in figure 5.1 where it is possible to visualize the input image dimensions across the layers and its final shape.

LeNet is a useful convolutional neural network that is usually used by beginner users to learn deep learning matters due to its short architecture and because it is implemented in many deep learning frameworks using it to explain the framework. Because of this, LeNet-5 has been used as the basis of the project and to learn the Theano implementation

and the Convolutional Neural Networks theory.

The code of LeNet in Python using the Theano library, and its explanation, is openly available in www.deeplearning.net.

5.1.1 LeNet-5 specifications

The architecture of the LeNet-5 downloaded code, used to start working in this project, is formed by two convolutional layers (size 5x5) with 20 kernels in the first convolutional layer and 50 in the second one. Each one of these followed by a max pooling-layer of size 2x2. Those four layers are followed by a fully-connected layer with 500 neurons at the output.

The classifier, which has been used is the logistic regression, has indeed been at the same time as the Convolutional Neural Network. The activation function of the convolutional layers and the logistic regression is the hyperbolic tangent (tanh). The learning rate used is 0.1 and the network runs by 200 epoch.

The cost function, or loss that must be minimized during the training, is the negative log-likelihood. LeNet-5 uses the stochastic gradient method with mini-batches (MSGD).

The data used is MNIST digit database, whose characteristics are described in section 4.2.1. The data consists of three subsets: training, testing and validating. Each subset is used for training, testing or validating, respectively.

The data is not fed to the network all at once, each subset is grouped in small subsets called batches and whose size is chosen by the user.In the code available of LeNet-5 the batch size is 500 samples. The network is fed by batches, so the size of the net depends on the batch size not the (train, test or validate) subset. In this example, the batch size, is the same for the three subsets. When the subset is divided into different batches, if there are samples that are not enough for a batch, those samples are not used.

The network train for a specified number of epoch. Each epoch has as many iterations as necessary to go through all batches of the train subset. The reason for using batches is to define the size of the network and because, usually, the quantity of samples used in deep learning is large (thousand, millions..) and too much memory would be needed to build a network of its size and the computational resources available may not be enough.

As the MNIST digit database available with the code has 50000 samples for training, 10000 for testing and 10000 for validating, the number of batches for each subset (with 500 samples for each batch) is 100 for training, 20 for testing and 20 for validating

The training procedure is being carried out for too many epochs as user has selected and returned the cost of the procedure. While the training is running, the validation is calculated for each epoch and the validation returns the error of the procedure. The training cost and the validation error are used to know the behaviour or the learning process of the network and how it generalizes with the purpose of choosing the best model.

The test is realized while the training is being executed, more specifically, when the validation has been done and the best results have been obtained during the whole process until that iteration. The result that is used to compare with others classifiers or with others articles.

In addition to the number of epochs, an alternative approach would be to stop the training procedure (*early-stopping*). This method is used to avoid over fitting tracking the validation process [43]. decision to stop the training depends on *the patience*. User chooses *patience* value.

5.1.2 LeNet-5 Results

While the training is being calculated, the weights are being updated in each iteration. When a model is selected or saved, weights are indeed what is being chosen or saved. An example of weights is represented in figure 5.2 where twenty first weights at epoch 10 of the first convolutional layer are demonstrated. Therefore when it is being trained, what the network is doing is adapting the weights to the input to get a good performance.

The training procedure returns the cost function in each iteration to evaluate the training behavior. The cost function obtained at training by executing LeNet-5 is represented in figure 5.3, and its value decreases as the iterations rise converging in almost 0; this curve is the desired one for each training practice, as it is does not oscillate abruptly and converges in a very low value logarithmically.

The error obtained at validation is represented in figure 5.4, where it could be seen that the value decreases logarithmically to converge, approximately, in 1 (a low value) and this behavior of the curve is the desired one in a validation process. The convergence value of the validation is not usually as low as the training convergence value, and the point where it starts to converge is at a subsequence later than in the training.

The test result has been calculated with the model acquired in iteration 18300 (epoch 37^{th}), with a validation error of 0.91%. The error rate obtained is 0.92% which means that 920 samples out of 100000 of the testing subset are being misclassified.

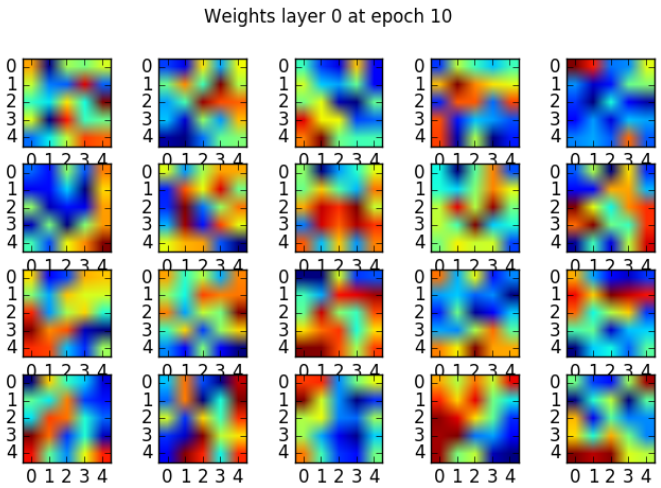


Figure 5.2: Weights at epoch 10 of the first convolutional layer.

5.1.3 Modifying LeNet

Modifications have been made to LeNet-5 architecture. Firstly, the batch size has been changed. Then the tanh activation function has been replaced, a normalization layer has been added and the weight initialization has been changed too. The database used for these experiments is the MNIST digit database.

Changing the batch size

Two experiments have been developed in order to compare the results when the batch size is changed and how the network behaviour changes too with regard to LeNet-5 with the original batch size (500).

The first experiment contained 20 samples per batch. For the second experiment, the batch size used is 100. In both cases, the training process was stopped by the early-stopping; at epoch 31^{th} it stopped the first experiment, because from the epoch 16^{th} epoch the error at validating was not improving and for the second experiment, at 33^{th} epoch, the early-stopping finished the training.

The validation error for both experiments and the original LeNet are represented in figure 5.5. From the images it could be seen that the validation error in first epochs is lower (9 % in the first experiment) than the validation error when the batch size is bigger. When the batch size is equal to 20, the optimal test error rate is obtained in the first 15^{th}

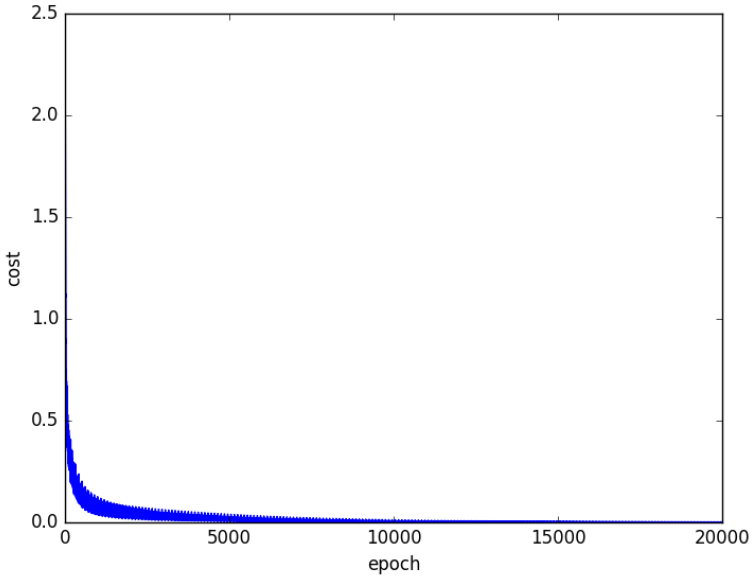


Figure 5.3: Cost function at training running LeNet-5 with MNIST digit database.

epochs, the same error rate than in the original case (0.92%). However for 100 samples per batch, it has not been possible to get to that error rate, the best test error rate has been 1.04% at iteration 8500.

To conclude, more epochs are necessary when the batch size is bigger, because there are not enough updates in each epoch [43]. Usually, the value of the batch size used is 32 [43] and generally, the choice is computational.

In figure 5.5 the error in each epoch is represented with a batch size of 500, the original size, 20 and 100. In the original case, the error starts with a value of 9% approx. When the batch size equals 20, the error in the first iteration is about 2.4%. Finally, with a batch of 100 images, the validation error is 3.5%.

With the original size and size equal to 20, it is possible to get to the same minimum error, the difference between those examples is that each one gets to that conclusion into different epochs. With a batch size equal to 100, the code stopped in epoch 33 because of the early-stop with a patience of 10000 obtaining a 1.04% test error at iteration 8500 and a 1.01% validation error. With a batch size equal to 20, the code also stopped earlier because of the same reason. Nonetheless, in this case it was possible to get to the same minimum as with the original size: the epoch in which it stopped was 31, it ran without getting a better validation score for 15 epochs.

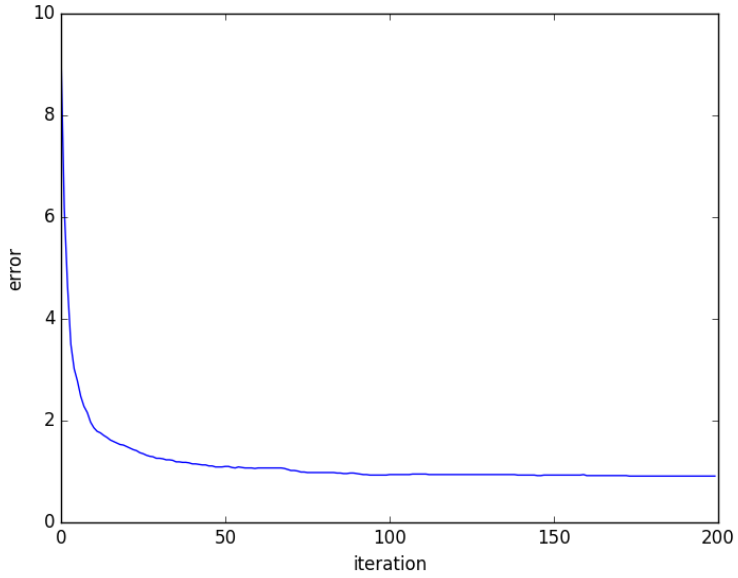


Figure 5.4: Validation error obtained with LeNet-5 with MNIST digit database.

Changing the activation function, normalization and weights initialization

Just as the batch was changed and its results were compared in the previous subsection, the activation function has also been changed, a normalization layer has been added and weights initialization has been changed.

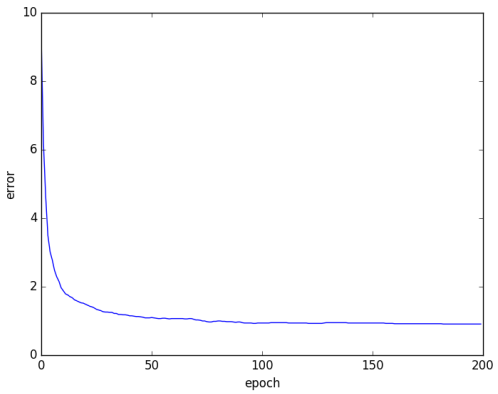
LeNet-5 does not use any normalization layer, but in this experiment from the different normalization availables (batch normalization, local normalization, etc.) Local Response normalization is added after the max-pooling layers.

The activation function used in LeNet-5 is tanh (hyperbolic tangent), it has been substituted by the rectified linear unit (ReLu) activation function.

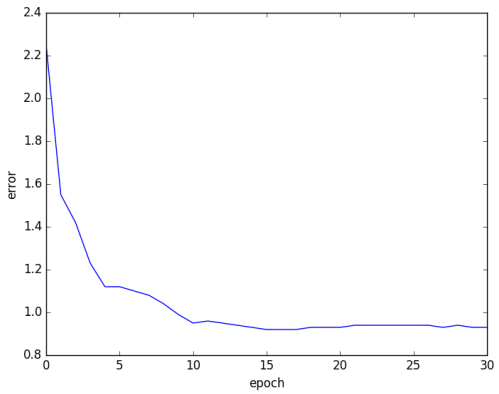
Regarding the initialization of weights, in LeNet, for the convolutional layers and the fully connected layer, a normalized initialization [44] is used:

$$W \sim U[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}] \tag{5.1}$$

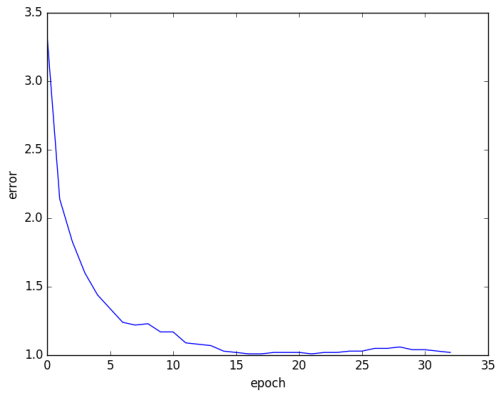
Where n_j is the number of neuron of the current layer and n_{j+1} is number of neurons of the following layers. In this experiment, this initialization has been changed to a weight initialization with a Gaussian distribution.



(a) batch size = 500 (original size)



(b) batch size = 20



(c) batch size = 100

Figure 5.5: Validation error in each epoch for different sizes of batches.

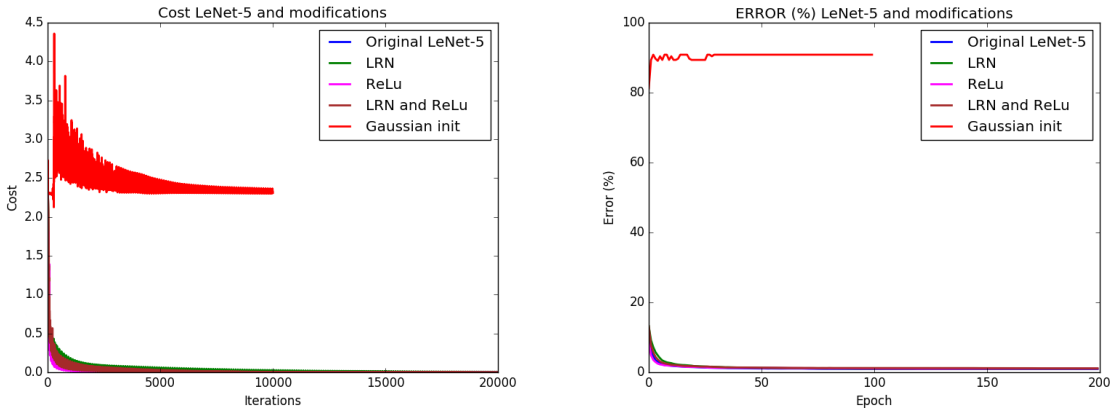
Here below the details of each experiment are described:

– Experiment 1: using Local Response Normalization (LRN): in which a normalization has been carried out in the convolutional-max pooling layers.

– Experiment 2: using ReLu as activation function: the activation function tanh has been substituted by ReLu activation function in convolutional and fully connected layers.

– Experiment 3: using ReLu and LRN: the activation function used is ReLu and LRN has been used as normalization layer.

– Experiment 4: changing weights initialization: weights initialization has been changed by Gaussian. In which the mean value that has been used is 0 and std is 0.01.



(a) Cost at training

(b) Error at validation

Figure 5.6: Cost at training and error at validatioin of LeNet-5 and the four experiments.

CNN model	Description	Best validation Error	Test Error
Lenet-5	Original Lenet-5 without modifications	0.91% (17400 iter.)	0.92%
Experiment 1	Using Local Response Normalization	0.99% (13400 iter.)	1.6%
Experiment 2	Using ReLu as activation function	1.04% (11900 iter.)	2.4%
Experiment 3	Using ReLu and LRN	1.18% (19500 iter.)	1.08%
Experiment 4	Gaussian weight initialization	81.22% (100 iter.)	80.90%

Table 5.1: Lenet-5 experiments results.

Weights initialization has been changed in convolutional and fully connected layers. Also, bias initialization has been changed by ones.

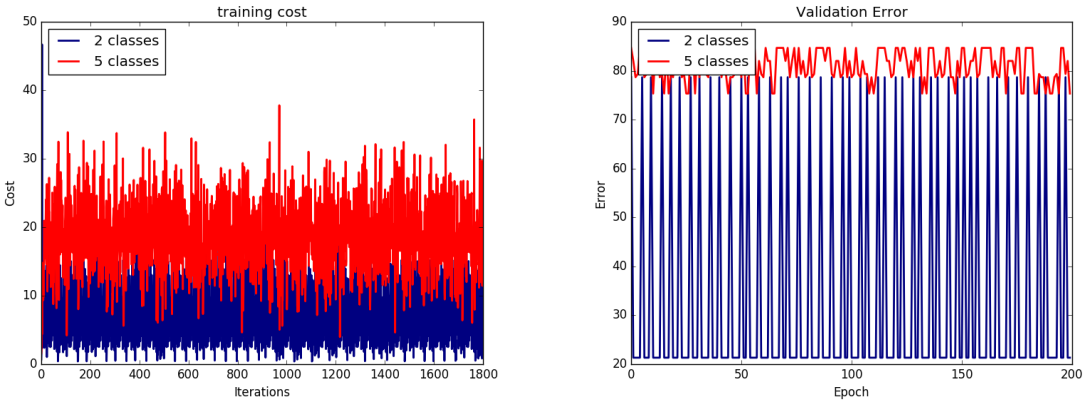
In figure 5.6 what is represented is the cost at training (figure 5.6(a)) and the error at validation (figure 5.6(b)) of the four training process experiments with the original LeNet-5 results. Each network has run 200 epoch, except when early-stopping was needed.

From figure 5.6, the first observation is that the Gaussian initialization has worsened the original result significantly, the cost converges in 2.3 while in other experiments it converges in a value close to 0; therefore, the training has converged in a local minimum. Moreover, the early-stopping led the training process for the fourth experiment Gaussian initialization experiment) to stop in 100^{th} epoch.

From figure 5.6, it can also be deduced that the training and validation processes in other experiments' behaviour is similar to the original LeNet-5 performance.

The results obtained at validation and testing for each experiment are exposed in table 5.1.

From table 5.1, it could be concluded that the best configuration for the network is



(a) Cost (b) Error

Figure 5.7: Cost at training (a) and Error at validation (b) when LeNet has been used with RGB FRAV database.

the original one. With Gaussian initialization, the network does not find a local minimum in such time. Using LRN and ReLu, the test result is closer to the one obtained with the original LeNet-5, but not as good as the last one. Changing the activation function has not been a good change. Not taking into account the original LeNet-5, the best test performance has been obtained with 1,08% using ReLu and LRN, but the best validation error is 0,99% obtained using just LRN. The modifications results are similar, except in experiment 4.

5.1.4 LeNet-5 and RGB FRAV faces database

The objective of this thesis is to train a convolutional neural network for face anti-spoofing, for which the following step is to feed LeNet-5 with a face database, RGB FRAV database.

In order to work with images, because of the difference among the shape of the images, they have been resized into 252x180. This new shape is proportional to $0.7 * height = weight$ because all studied images save that proportion approximately.

The network has been tested with this database in the two ways to classify images: with two classes (genuine and attacks) and five classes (genuine and four classes, one per type of attack).

The architecture used is the same as LeNet-5 except for the batch size that has been reduced to 50 because there are not as many samples as in the MNIST database.

Figure 5.7 represents the training cost 5.7(a) and the validation error 5.7(b) of the

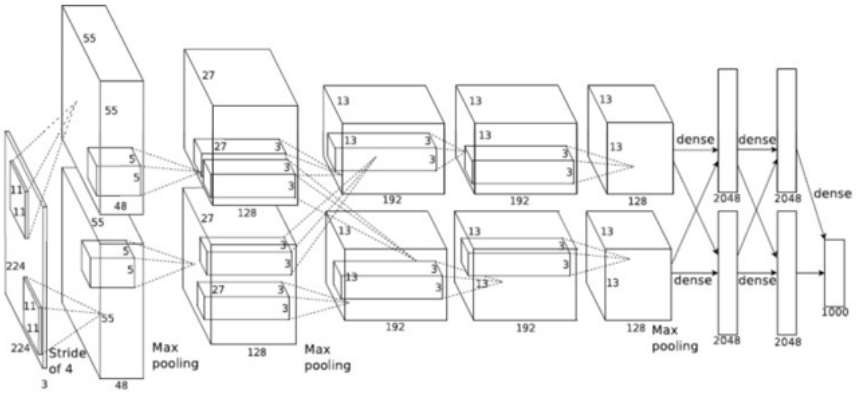


Figure 5.8: Imagenet architecture.

training process. In the figure, it is represented in blue when 2 classes are used to classify and red when 5 classes are used, for both cases (cost and error). From the figure it is possible to visualize that the whole training process is not desirable for 2 classes and 5 classes, because it seems that the network does not learn, the architecture requires to be modified.

5.2 Adapting LeNet-5 architecture

From LeNet-5 architecture, changes have been made in order to build a similar convolutional architecture as described in [1]. LeNet-5 architecture is simple to use for this projects purpose.

In [1] authors develop a convolutional neural network with similar objectives as the one described in this thesis. The CNN is based on *Imagenet* architecture [3], this is a specific architecture used to classify objects which are well known and regularly used in the deep learning community. Its architecture is described in figure 5.8.

Imagenet is formed by convolutional layers, max-pooling layers, normalization layers, dropout layers and a fully-connected layer.

Authors from [1] slightly describe the architecture, the parameters and the changes made from Imagenet. Therefore, it is not possible to develop an architecture based on the one used in the article [1], hence, it is going to be based on Imagenet.

5.2.1 New Architecture

The new architecture comprises the following layers:

- Convolutional layer with 96 kernels of size 11x11 followed by a max-pool layer (2x2 size) and a local response normalization.
- Convolutional layer with 256 kernels whose size is 4x4.
- Convolutional layer with 386 kernels whose size is 3x3.
- Convolutional layer with 384 kernels whose size is 3x3.
- Convolutional layer with 256 kernels whose size is 3x3 followed by a max-pool layer of 2x2 size.
- Dropout layer with 4096 neurons.
- Dropout layer with 4096 neurons.
- Fully Connected layer with 2000 neurons at the output.

ReLu has been used as an activation function. Logistic regression has been used in the training process. Weights have been initialized with Gaussian initialization and bias to 1. The learning rate value is 0.01.

5.2.2 Databases

The databases used in the experiments are CASIA images, CASIA videos, RGB FRAV, (RGB+NIR) FRAV feature level database and MFSD-MSU database. Databases' description is available in section 4.2. Just two classes are going to be used in this experiment: class 0 (real users class) and class 1 (attack class).

The distribution of samples is described in table 5.2 where the number of samples per class and per subset (train, test and validation) are summarized for each database. As it can be seen the number of positive (class 0) samples is lower for all the databases, but more specifically, the amount of positive samples in MFSD-MSU database is very small.

	RGB FRAV	RGB+NIR FRAV	Images CASIA	Videos CASIA	MFSD-MSU
Train samples class 0	157	133	26	255	30
Train samples class 1	459	417	111	81	68
Valid samples class 0	10	8	7	105	2
Valid samples class 1	83	70	13	39	12
Test samples class 0	19	16	16	540	3
Test samples class 1	167	70	23	180	25

Table 5.2: Samples distribution for each database.

Experiment	Color legend
Training cost	
General Experiment	Blue
Experiment 2 and 3	Orange
Experiment 4	Brown
Validation Error	
General Experiment	Blue
Experiment 2	Green
Experiment 3	Magenta
Experiment 4	Brown

Table 5.3: Experiments colour legend.

Databases are built independently from CNN process. Images are read, shuffled and split into train, test and validation subsets in order to be saved in a pickle file and used in the same way as many times as necessary without reading the raw data.

5.2.3 Description of the experiments

Databases described in the previous section 5.2.2 are used with the purpose of carrying out some experiments which are going to be described. In order to test, the used classifier is SVM with RBF, the parameter C has been searched for each experiment:

- General experiment: the architecture described in 5.2.1 is trained without changes.
- Experiment 2: with the purpose of trying to know if the configuration of the network is correct, only it is going to be trained with 20 samples for each subset, except for the MFSD-MSU database in which 14 samples are used.
- Experiment 3: with the aim to of provoke overfitting, 20 samples are used for each subset, however validation samples are the same as training samples.
- Experiment 4: the finality is the same as Experiment 3, knowing the behaviour of the network and provoke overfitting, but differs in the learning rate that has been changed to 0.001 value.

Because the training process in experiment 2 and experiment 3 should be the same (the difference among these experiments is the validation process, and thus, the test process), only one cost will be represented.

In figures 5.9, 5.10, 5.11, 5.12, 5.13 and 5.14 what can be seen is the cost at training and the error at validation for each database, when the four described experiments have been developed. All figures have the same color legend which is summarized in table 5.3.

	SVM Optima C	TP	TN	FP	FN
RGB FRAV	0.05	136	24	11	9
RGB+NIR FRAV (feat level)	0.1	113	24	1	2
CASIA images	5	9	2	0	9
CASIA videos	0.1	478	75	105	62
MFSD-MSU	10	19	1	8	0

Table 5.4: TP, FP, TN and FN obtained with each database.

From the general experiment, it can be seen that the cost converges in low values, the validation error converges (from a 10% error to very low values near to 0% error) for all databases except for MFSD-MSU database as could be seen in figure 5.14 which means that the generalization is not correct.

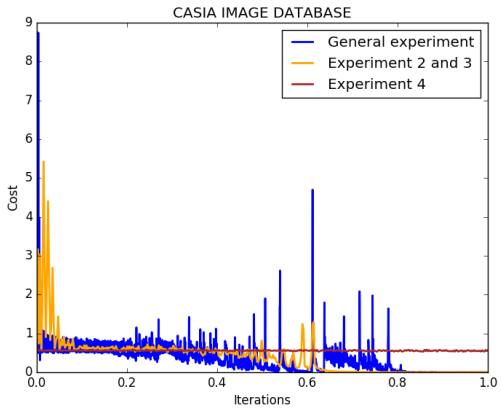
From Experiment 2, one would expect that the training and validation processes converge without oscillations and in less epochs, but because of the unbalanced samples, the validation results are worsen.

From Experiment 3 and experiment 2, overfitting would be expected. Just in Experiment 3 and in databases, except in the FRAV, it is obtained.

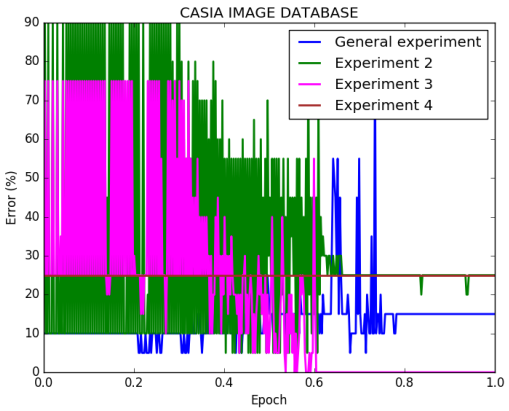
Decreasing the learning rate from 0.01 to 0.001 does not improve the performance as shown for all databases in figures 5.9, 5.10, 5.11, 5.12, 5.13 and 5.14. The training curve should converge with more epochs, but the training and validation error is constant.

Table 5.4 shows the positive and negatives rates obtained for each database when the general experiment (the first one described) has been tested. The results obtained with the RGB+NIR FRAV database are the best since just 3 samples have been misclassified from 140 images. In general, the classification is not bad. The database whose samples are misclassified in a big amount is CASIA videos, but it should also be pointed out that the total number of samples is much bigger too.

What is concluded from this is the need of a balanced database, at least to do this experiment. In which the number of class 0 samples is the same as the number of class 1 samples. This is due to the fact that the network would not learn in the same way if, in some cases, the number of samples of the attack class is four times the number of samples of class 1. Just predicting 0 would have a 25% accuracy, and having less than 5 samples in validation or testing is not a good generalizer (2 samples in class 0 MFSD database).

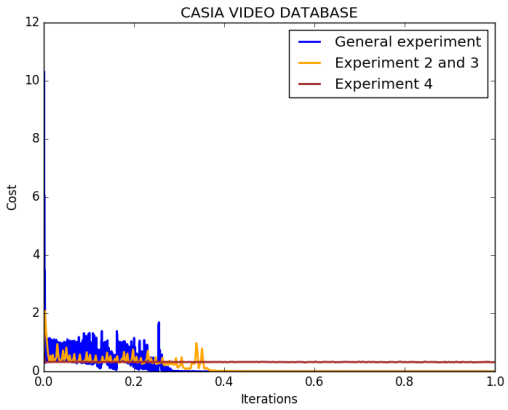


(a) Cost

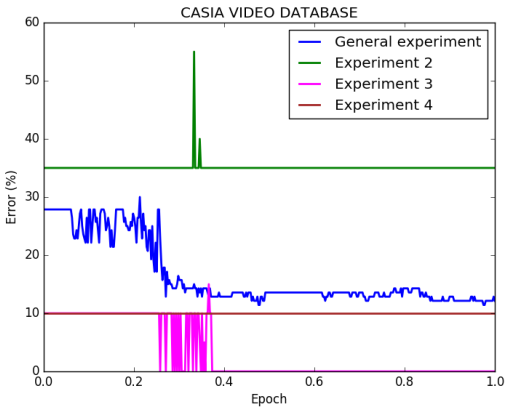


(b) Error

Figure 5.9: Cost at training (a) and Error at validation (b) for general experiment when CASIA image database has been used.

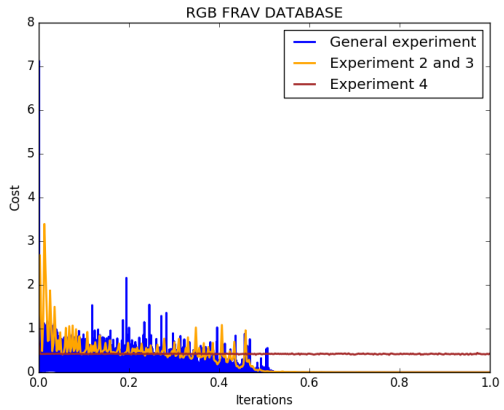


(a) Cost

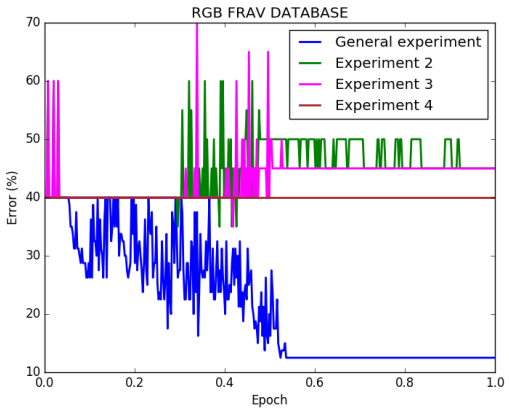


(b) Error

Figure 5.10: Cost at training (a) and Error at validation (b) for general experiment when CASIA video database has been used.

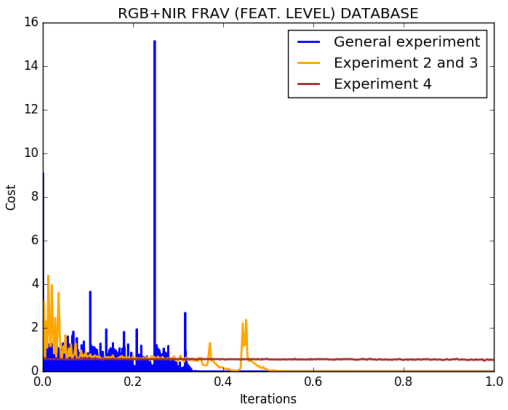


(a) Cost

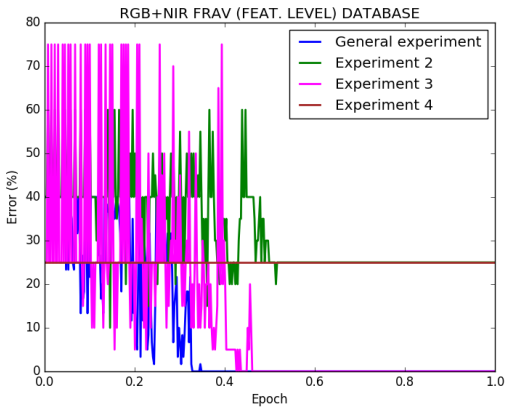


(b) Error

Figure 5.11: Cost at training (a) and Error at validation (b) for general experiment when RGB FRAV database has been used.

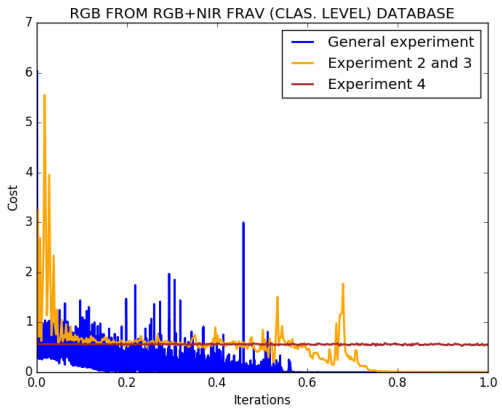


(a) Cost

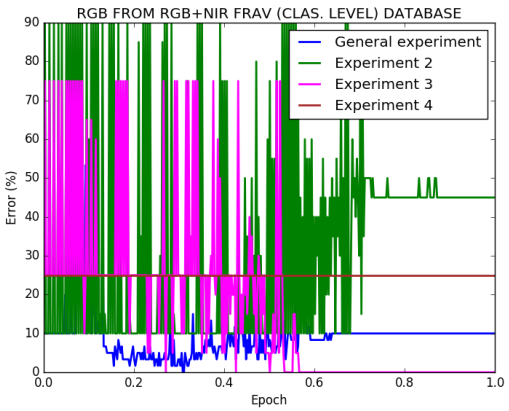


(b) Error

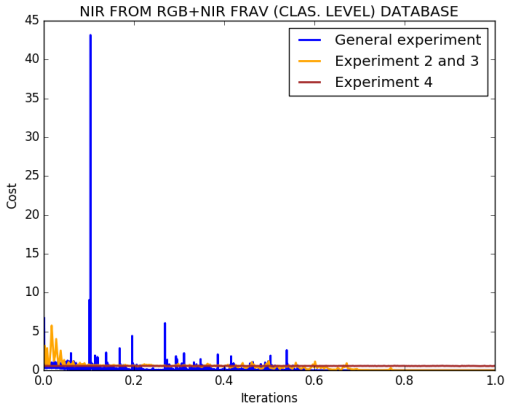
Figure 5.12: Cost at training (a) and Error at validation (b) for general experiment when RGB+NIR (feature level) FRAV database has been used.



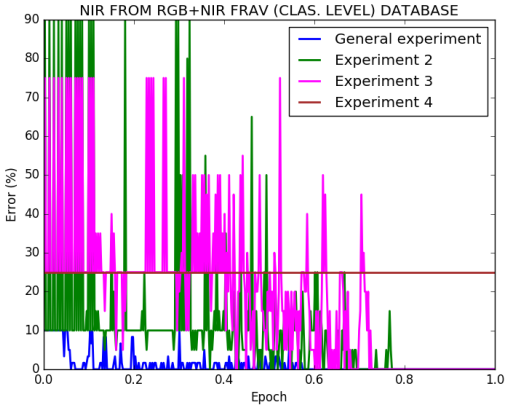
(a) RGB Cost



(b) RGB Error

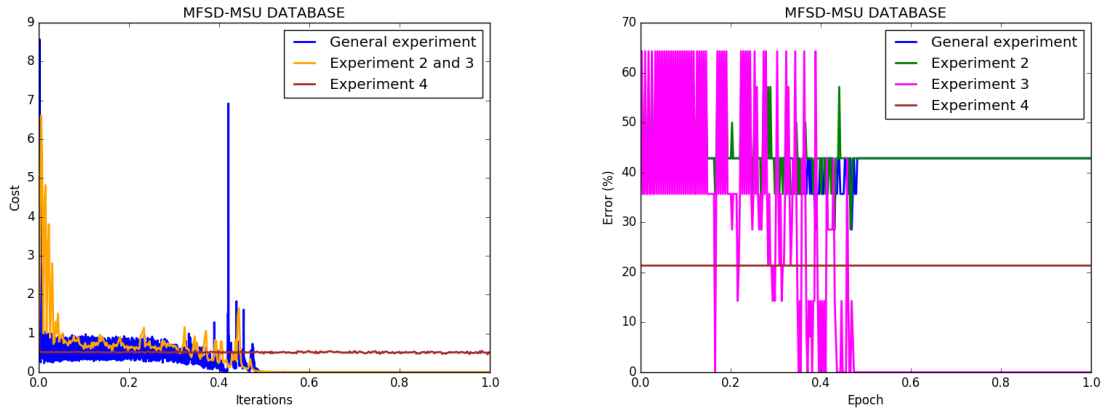


(c) NIR Cost



(d) NIR Error

Figure 5.13: RGB Cost at training (a) and Error at validation (b), NIR Cost at training (c) and Error at validation (d) for general experiment when RGB+NIR (classification level) FRAV database has been used.



(a) Cost (b) Error

Figure 5.14: Cost at training (a) and Error at validation (b) for general experiment when MFSD-MSU database has been used.

5.3 Description of the final experiment

In this section, the last, and the most relevant experiment is described.

The architecture used for this experiment is the same as the one used in 5.2.1. The learning rate (0.01), the classifier used at training (logistic regression) and the batch size (20 and 14) are the same too. The number of epochs has been increased to 600.

All the databases are utilized. The distribution of samples is described in table 5.2, the same one used in 5.2.2.

The difference between this experiment and the carried out in 5.2.3 is the classification task.

Each database has been trained and validated individually. The best model obtained at validation is saved for testing, for each database. Features obtained at the output of the best Neural Network model are used to feed each classifier for training (with training samples), validating (with validation samples) and testing (with testing samples).

Classifiers are trained with the features of the training samples. The best model of the classifier is selected depending on the features obtained of validation samples and the test is carried to with the features of the test samples.

The classifiers used to classify the features of the output of the CNN and to get the

results are the SVM (with RBF and linear kernel), KNN, Decision Tree and logistic regression. Also, PCA and LDA techniques have been used with each classifier separately to reduce the dimensionality of the features.

The classifiers, before using them, have been personalized for each and every particular time (for each database and if LDA or PCA is used). To do so, a cross validation has been used, more specifically, the *cros_val_Score()* function from *Sklearn* has been used with 10 folders:

- For the SMV classifier, the C parameter has been searched among the following values: 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2, 3, 5 and 10.
- For the KNN classifier, the number of neighbours (K) has been searched among the following values: 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28 and 30.
- For the Deep Trees classifier, the depth of the tree has been searches among the following values: 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28 and 30.
- For the PCA, the number of components has been found in a range of 3 to 5000, in 3 to 3 steps.
- For the LDA, the number of components has been found in a range of 1 to the length of the characteristic vector in 10 to 10 steps.

5.4 Final experiment results

In this section, the test results obtained from section 5.3 are presented as well as the training process.

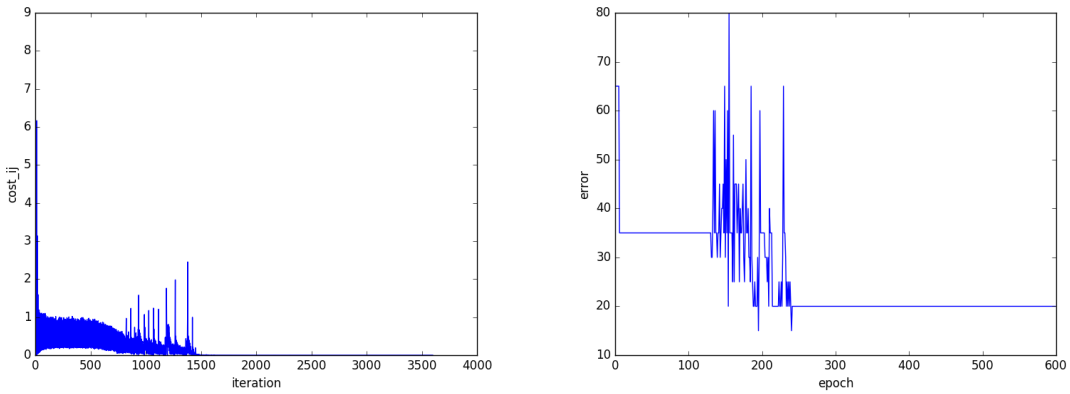
5.4.1 Training and validation process

The convolutional neural networks has been trained independently for each database as explained in 5.3. In this section, the training and validation process are detailed for each database.

CASIA Image database

In figure 5.15 the cost and training are represented (5.15(a) and 5.15(a) respectively). The training process converges in low values, smaller than 0.0001; while the validation process converges in a 20% error from the 250^{th} epoch. The iterations, where the training process oscillates sharply, agree with the epochs where the cost oscillates sharply too. After those oscillations, the validation becomes constant and the training error variates in its lowest

values.



(a) Cost at training

(b) Error at validation

Figure 5.15: Cost (a) and error (b) at training CASIA image database.

The saved model to carry out the test performance is the one obtained at iteration 1176 with a 15% validation error.

CASIA Video database

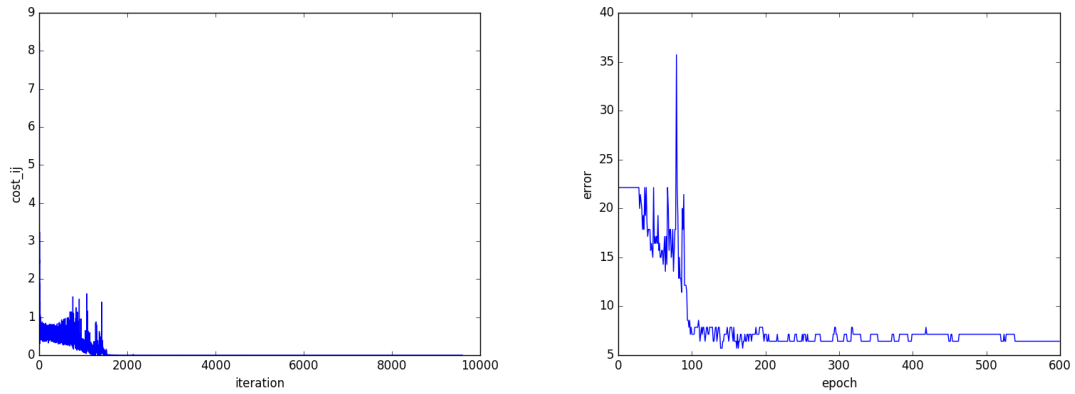
The cost and error obtained at training and validation processes when using the CASIA video database, is represented in figure 5.16. The minimum cost is almost 0 and it is obtained before the 2000^{th} iteration, as represented in figure 5.16(a). The validation converges at the same time as the training, where it oscillates between 7% and 6%.

The best validation performance is obtained at iteration 2240 with a 5.71% validation error.

RGB FRAV database

The cost and error obtained at training and validation processes respectively are represented in figure 5.17. The cost, that can be seen in figure 5.17(a) has a low value when it converges before the 5000th iteration; the cost obtained is 10^{-6}. The validation error, represented in figure 5.17(a), fluctuates considerably. The curve should decrease, yet it does not, and in the 200th epoch, the error is constant in 11.25%. The generalization at validation is not very good and before the 100th epoch the network seems to overfit.

The best validation score of 1.25% has been obtained at iteration 2016 which is saved as a model for testing.

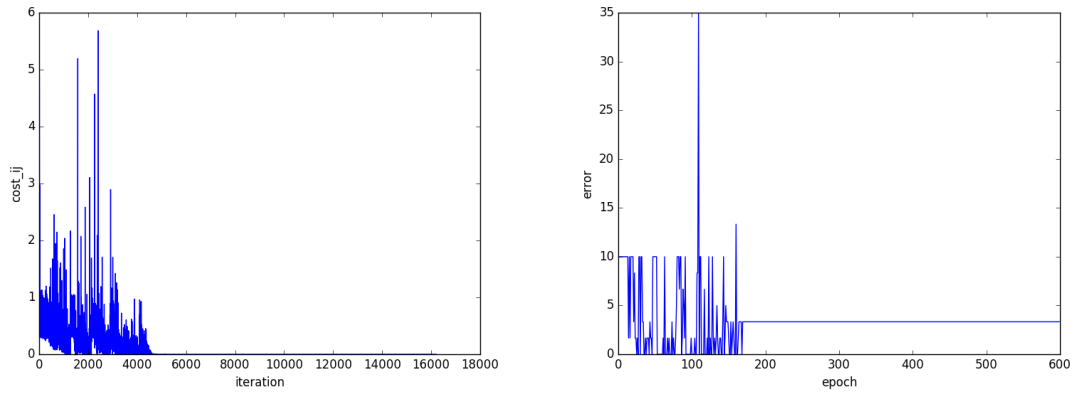


(a) Cost at training (b) Error at validation

Figure 5.16: Cost (a) and error (b) at training CASIA video database.

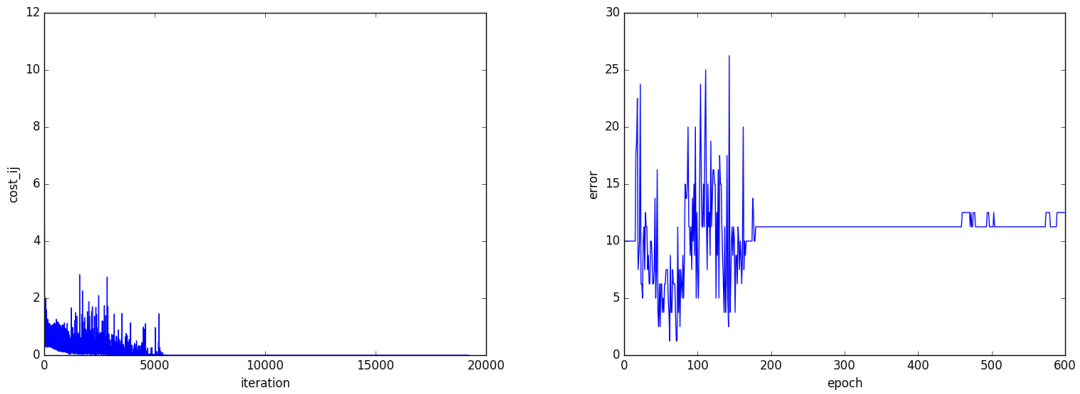
RGB+NIR (feature level) FRAV database

The RGB+NIR FRAV database, whose images have been added before the training process, has a cost and error which is represented in figure 5.18. The cost at training, that can be seen in 5.18(a) oscillates in the 3500 first iterations until it decreases its value to 0 in the last 100 iterations. The cost at validation oscillates in the same oscillation cost period, where the value gets to 0% at different times. Subsequently, the value gets to 3,3%. Although the validation performance does not decrease in a desirable way, it gets a 0% error which means that the generalization is very good.



(a) Cost at training (b) Error at validation

Figure 5.18: Cost (a) and error (b) at training RGB and NIR FRAV (feature level) image database.



(a) Cost at training

(b) Error at validation

Figure 5.17: Cost (a) and error (b) at training RGB FRAV image database.

The best model is obtained at 702 iteration, when the validation gets 0% for the first time.

RGB+NIR (classification level) FRAV database

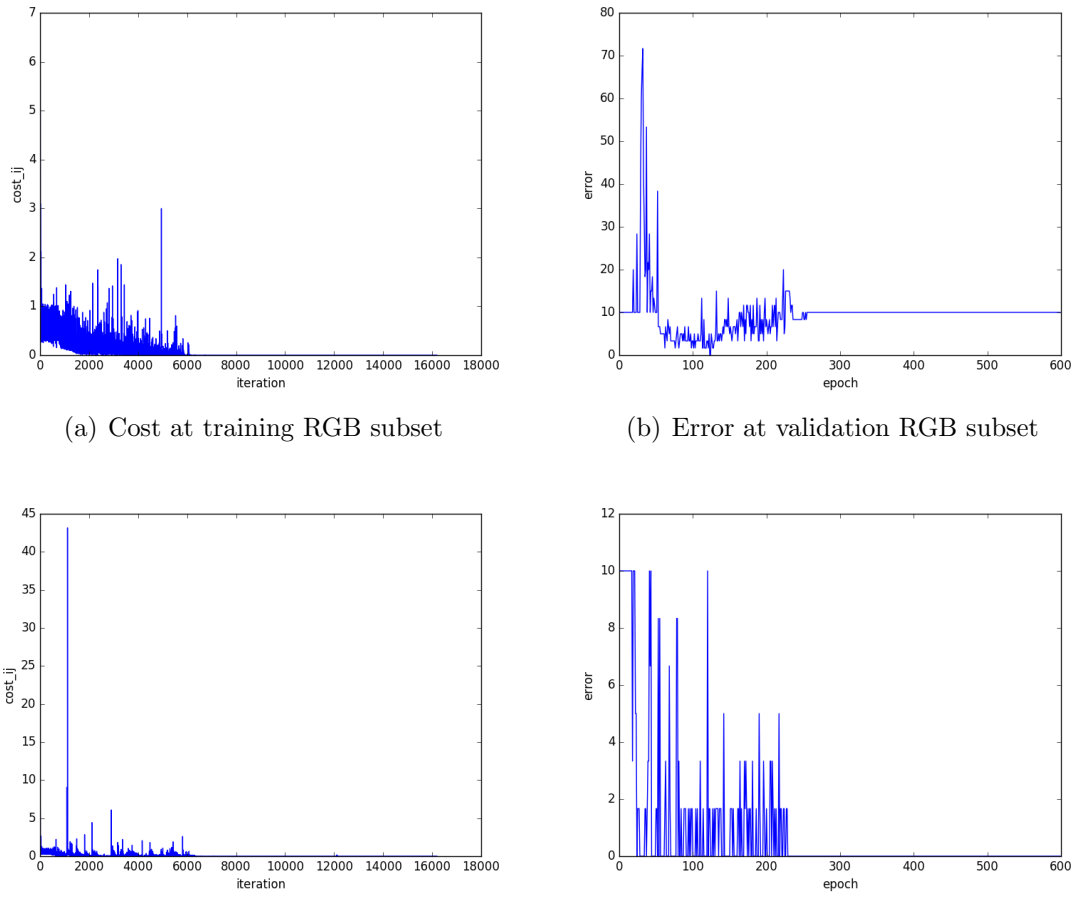
In figure 5.19 the training cost and the validation error is represented when the database has been trained, first the RGB subset and then NIR subset. These two subsets are trained independently and features of each best model are concatenated to test the performance.

In RGB and NIR cost training, it got a very low value as in others databases after decreasing its value. Regarding the cost in validation, it gets 0% in both cases. The difference is that in RGB, this lowest value is gotten twice at epoch 124 and 125 and then the value is increased until it becomes constant at 10%, what seems to be an overfit. Nevertheless, in the NIR validation error, it oscillates until it gets 0% value and is constant until the end.

The best RGB model is the one obtained at iteration 3347, when 0.0% is gotten at validation error. The best NIR model is the obtained at iteration 674, the first time that the validation error gets 0%.

MFSD database

The training results obtained with theMFSD database are represented in figure 5.20. The cost slowly decreases until it get a desirable cost value, near to 0%. The validation error curve graph (figure 5.20(b)) is not as good as the cost curve. The lower error value is obtained in the first 100 iterations, which is constant and then it increased until 35%.



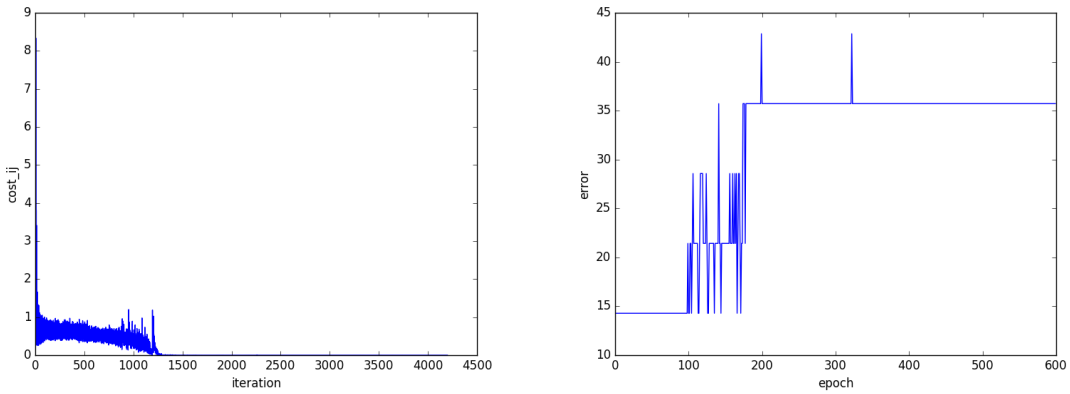
(a) Cost at training RGB subset

(b) Error at validation RGB subset

(c) Cost at training NIR subset

(d) Error at validation NIR subset

Figure 5.19: Cost (a) and error (b) at training RGB subset, cost (c) and error (d) at training NIR subset for FRAV (RGB+NIR at classification level) image database.



(a) Cost at training

(b) Error at validation

Figure 5.20: Cost (a) and error (b) at training MFSD image database.

The best model is the one obtained at iteration 7 in the first epoch, where the validation error is 14,28%.

5.4.2 Testing process

From the best CNN model, for each database, A SVM (with linear and RBF kernel), a KNN, a decision tree and a logistic regression have been trained. Furthermore,the testing has been carried out after training a LDA and PCA (which have been trained too for each database) with each classifier.

The test performance for each classifier has been executed after chosing the best classifier model. The results are going to be exposed independently of each database and then compared among all databases.

Best obtained results are highlighted in bold in each table.

CASIA Image database

The results obtained testing the CNN model for CASIA image database are shown.

In table 5.5 what is exposed are the APCR and BPCR parameters for each classifier. All the attacks samples are classified correctly because the APCR value is 0. The number of genuine samples misclassified is what changes: it takes value 1 when using SVM with kernel lineal, hence all the positives samples have been misclassified. And it takes value 0.5 when using the decision tree, whose number of misclassified samples is 5, (half of the

Classifier	APCR	BPCR	Classifier + PCA n components = 103	APCR	BPCR	Classifier + LDA n components = 1	APCR	BPCR
SVM - RBF C = 0.5	0	0.125	SVM - RBF C = 1	0	0.125	SVM - RBF C = 0.5	0	0.125
SVM - lineal C = 0.001	0	1	SVM - lineal C = 0.001	0	1	SVM - lineal C = 0.005	0	1
KNN k = 2	0	0.125	KNN k = 2	0	0.125	KNN k = 4	0	0.125
Decision Tree Depth = 12	0	0.5	Decision Tree Depth = 4	0	0.125	Decision Tree Depth = 2	0	0.125
Logistic Regression l. rate = 0.01	0	0.125	Logistic Regression l. rate = 0.0001	0	0.125	Logistic Regression l. rate = 0.0001	0	0.125

Table 5.5: APCR and BPCR classifying results using CASIA image database.

total real samples of the test subset). The rest of the classifiers take a 0.125 APCR value, just one sample is misclassified, the same sample which could be seen in figure 5.21.

Regarding the APCR and the BPCR, it is not possible to get the best combination with which the results are better because most classifiers work correctly.

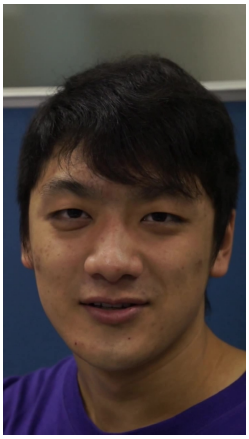


Figure 5.21: Casia image misclassified sample.

Figure 5.22 represents FAR and FRR curves, from which is obtained a 0,025 EER at 0,015 threshold. The EER obtained is a desirable result, on the contrary, the threshold is low, it is not usually used a 0,015 threshold.

CASIA Video database

Next analysed results are from the CASIA video database.

In table 5.6 APCR and BPCR results are summarized. The best result has been obtained with SVM classifier and kernel RBF (C= 0.05) after applying LDA with 1 component. The worst values obtained are the SVM with lineal kernel, even if PCA or LDA

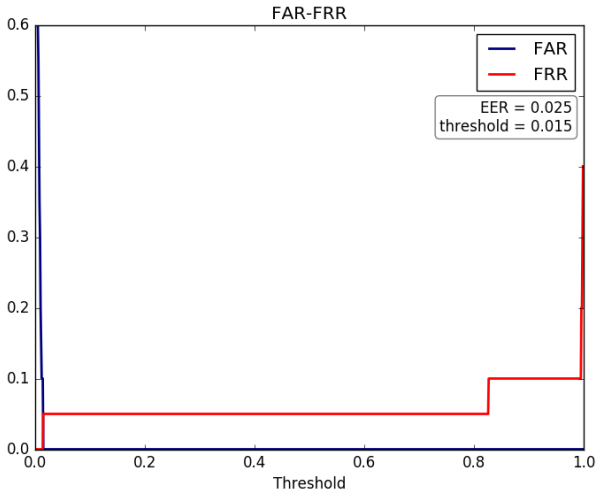


Figure 5.22: FAR-FRR curve of CASIA image database using SVM (RBF kernel).

Classifier	APCR	BPCR	Classifier + PCA n components = 283	APCR	BPCR	Classifier + LDA n components = 1	APCR	BPCR
SVM - RBF C = 0.1	0.14	0.53	SVM - RBF C = 0.5	0.17	0.46	SVM - RBF C = 0.05	**0.16**	**0.44**
SVM - lineal C = 0.001	0	1	SVM - lineal C = 0.001	0	1	SVM - lineal C = 0.001	0	1
KNN k = 2	0.19	0.42	KNN k = 2	0.2	0.42	KNN k = 2	0.16	0.47
Decision Tree Depth = 2	0.19	0.49	Decision Tree Depth = 2	0.15	0.53	Decision Tree Depth = 2	0.15	0.46
Logistic Regression l. rate = 0.01	0.15	0.49	Logistic Regression l. rate = 0.005	0.23	0.34	Logistic Regression l. rate = 0.005	0.23	0.37

Table 5.6: APCR and BPCR classifying results using CASIA video database.

have been used, because all the positives samples have been classified as negative.

From the table it is not possible to know if LDA and PCA are significant because with some classifiers APCR and BPCR results have been improved or worsened.

Also The last conclusion that could be obtained from the table is that the attack samples are better classified than the genuine samples.This is due to the fact that it has been trained with more negative samples.

Figure 5.23 represents the FAR-FRR curve of the best obtained model: CNN architecture and using LDA with SVM RBF kernel. From the figure, the Equal Error Rate obtained value is 0.115, it is a improvable value; EER is obtained at 0,64 threshold.

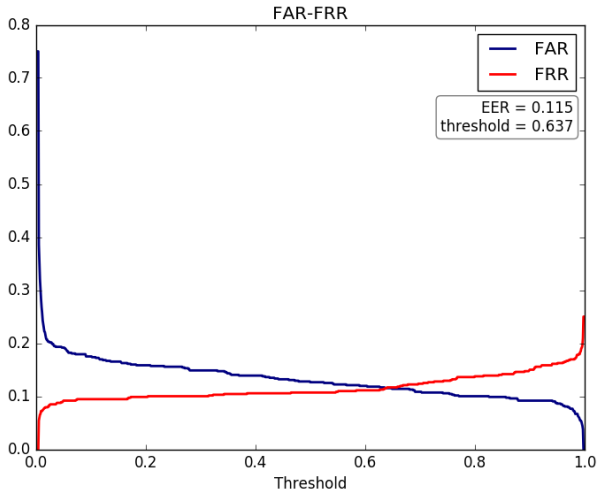


Figure 5.23: FAR-FRR curve of CASIA videos database using LDA and SVM (RBF kernel).

RGB FRAV database

The RGB FRAV database results are presented and analyzed.

Classifier	APCR	BPCR	Classifier + PCA n components = 463	APCR	BPCR	Classifier + LDA n components = 1	APCR	BPCR
SVM - RBF C = 0.5	0.06	0.06	SVM - RBF C = 1	**0.04**	**0.06**	SVM - RBF C = 0.1	0.05	0.06
SVM - lineal C = 0.005	0	1	SVM - lineal C = 0.005	0	1	SVM - lineal C = 0.5	0.05	0.06
KNN k = 16	0.07	0.06	KNN k = 26	0.06	0.06	KNN k = 20	0.05	0.06
Decision Tree Depth = 2	0.04	0.11	Decision Tree Depth = 2	0.06	0.11	Decision Tree Depth = 2	0.06	0.06
Logistic Regression l. rate = 0.01	0.01	0.17	Logistic Regression l. rate = 0.05	0.06	0.06	Logistic Regression l. rate = 0.005	0.06	0.06

Table 5.7: APCR and BPCR classifying results using FRAV RGB database.

APCR and BPCR results are in table 5.7. In general, results are very good because values are lower than 0.5. With SVM lineal kernel, all the samples are classified as negatives except when it has been used when LDA, those results are as good as the ones obtained with RBF kernel.

The best result is obtained when SVM with RBF kernel has been used with C = 1 and using LDA with 1 component. LDA improves the results obtained when neither PCA nor just the classifier have been used, although the improvement is not much.

The misclassified samples are repeated among the classifiers. The two samples shown

in figure 5.24 represent the two most misclassified images, almost every classifier has classified it incorrectly.

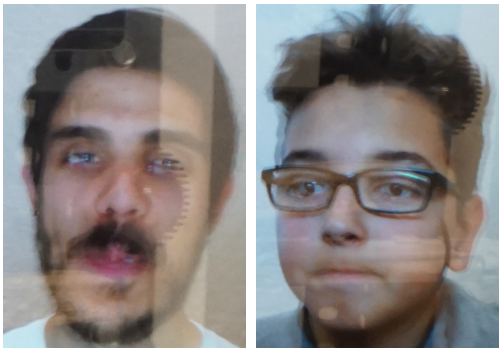


Figure 5.24: RGB FRAV misclassified samples.

The FAR-FRR graph is represented in figure 5.25 from which the EER value acquired is 0.022, a satisfactory value, at 0.954 threshold. The performance of the FAR and FRR curves is acceptable because the values obtained are low, and that means that the false positives and false negatives are a small quantity.

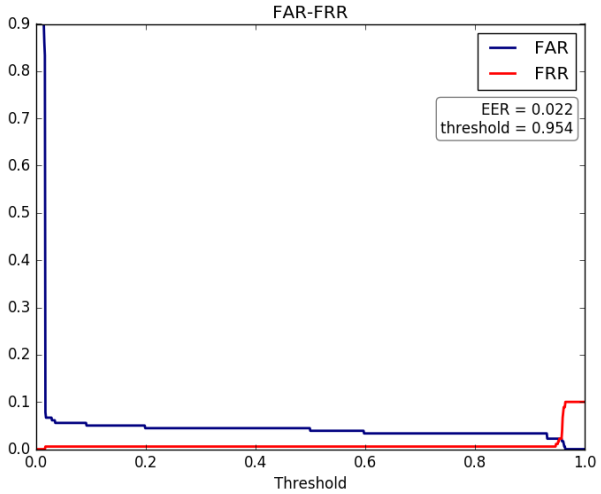


Figure 5.25: FAR-FRR curve of RGB FRAV database using LDA and SVM (RBF kernel).

FRAV RGB+NIR (feature level) database

Results obtained at classifying RGB+NIR (feature level) database are shown next.

Table 5.8 shows the APCR and BPCR values that have been obtained for each classifier. Except for SVM with lineal kernel, which has classified all the samples as negatives

Classifier	APCR	BPCR	Classifier + PCA n components = 463	APCR	BPCR	Classifier + LDA n components = 1	APCR	BPCR
SVM - RBF C = 2	0.02	0	SVM - RBF C = 5	0.02	0	SVM - RBF C = 0.1	0.04	0
SVM - lineal C = 0.005	0	1	SVM - lineal C = 0.001	0	1	SVM - lineal C = 10	0.05	0
KNN k = 6	0.09	0	KNN k = 12	0.12	0	KNN k = 10	0.04	0
Decision Tree Depth = 2	**0**	**0**	Decision Tree Depth = 2	0.01	0	Decision Tree Depth = 2	0.05	0
Logistic Regression l. rate = 0.01	**0**	**0**	Logistic Regression l. rate = 0.05	0.06	0	Logistic Regression l. rate = 0.05	0.07	0

Table 5.8: APCR and BPCR classifying results using FRAV RGB+NIR (feature level) database.

when just the classifier has been used or if PCA has been applied, classifications have obtained good results. The APCR and BPCR values are very low, close to 0. In fact, the Decision Tree and logistic regression classify correctly all the samples because APCR and BPCR are equal to 0.

PCA and LDA have barely improved results. SVM lineal with LDA does not classify all the samples as negatives. When PCA and LDA is applied with logistic regression and Decision tree, the classification is not perfect as it is when none is applied.

In general, the same samples are misclassified in the classification tasks. The two most misclassified samples are represented in figure 5.26.

The perfect performance could be also demonstrated in figure 5.27, where FAR and FRR curves are plot when logistic regression and Decision Tree classifiers are used. In both cases, because of the perfect classification, the EER value obtained is 0.

FRAV RGB+NIR (classification level) database

Results obtained when the RGB+NIR FRAV database (classification level) is used are described below.

APCER and BPCER results are summarized in table 5.9. Results are closer to 0, but no classifier classifies perfectly. When using LDA, one can see that APCR results improve and the BPCR results are perfect.

SVM lineal classifier classifies incorrectly positives samples although PCA is used too. With LDA, that does not happen.

In most cases, classifiers just mis-classify one sample, the same sample represented in figure 5.28. Classifiers that mis-classify more than one sample, the sample 5.28 is one of

Figure 5.26: RGB+NIR FRAV (feature level) misclassified samples.

the misclassified.

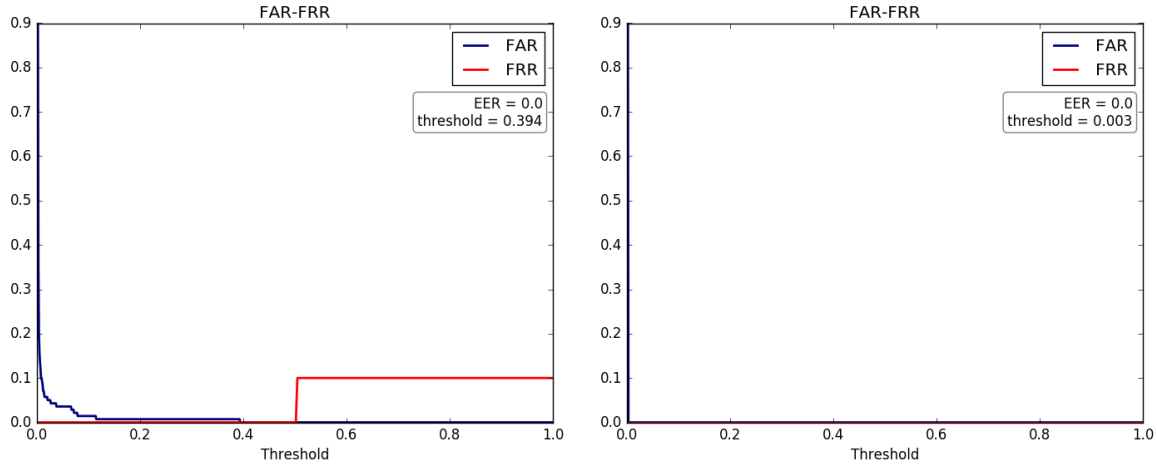
There are various classifiers which realize the best performance, FAR-FRR curve it is illustrated for SVM (RBF kernel) with LDA method in figure 5.29. The result obtained is acceptable, a 0,004 ERR. The number of incorrectly classified samples is low.

MFSD-MSU database

MFSD-MSU database results are described following.

Classifier	APCR	BPCR	Classifier + PCA n components = 463	APCR	BPCR	Classifier + LDA n components = 1	APCR	BPCR
SVM - RBF C = 10	0.01	0	SVM - RBF C = 1	0.02	0	SVM - RBF C = 2	0.01	0
SVM - lineal C = 0.001	0	1	SVM - lineal C = 0.001	0	1	SVM - lineal C = 1	0.01	0
KNN k = 6	0.32	0	KNN k = 14	0.04	0	KNN k = 6	0.01	0
Decision Tree Depth = 2	0.04	0.07	Decision Tree Depth = 2	0.05	0.07	Decision Tree Depth = 2	0.01	0
Logistic Regression l. rate = 0.01	0	0.29	Logistic Regression l. rate = 0.05	0.07	0	Logistic Regression l. rate = 0.005	0.02	0

Table 5.9: APCR and BPCR classifying results using FRAV RGB+NIR (classification level) database.



(a) logistic regression classifier (b) Decision Tree classifier

Figure 5.27: FAR-FRR curve for RGB+NIR FRAV database (feature level).

Figure 5.28: RGB+NIR FRAV (classification level) misclassified samples.

The classification with this database is not accurate because all the samples are classified as negatives, as could be seen in table 5.10 where , independently of the classifier or if LDA and PCA are used, the BPCR value is always 1 and the APCR value is 0.

Because of the similarity in results, only FAR-FRR curve of SVM with RBF is demonstrated. Figure 5.30 showa FAR-FRR curve. The result obtained is an 0,089 EER, it is not an unacceptable value, but it is due to the fact that all samples are classified as negative samples.

5.4.3 Comparative among databases

In this section it is going to discuss databases classification in common.

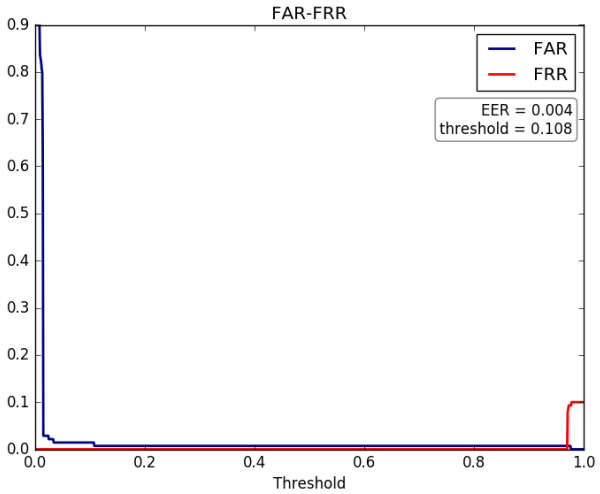


Figure 5.29: FAR-FRR curve of RGB+NIR FRAV (classification level) database using LDA and SVM (RBF kernel).

Classifier	APCR	BPCR	Classifier + PCA n components = 463	APCR	BPCR	Classifier + LDA n components = 1	APCR	BPCR
SVM - RBF C = 0.01	0	1	SVM - RBF C = 0.001	0	1	SVM - RBF C = 1	0	1
SVM - lineal C = 0.001	0	1	SVM - lineal C = 0.001	0	1	SVM - lineal C = 10	0	1
KNN k = 24	0	1	KNN k = 30	0	1	KNN k = 30	0	1
Decision Tree Depth = 12	0	1	Decision Tree Depth = 2	0	1	Decision Tree Depth = 2	0	1
Logistic Regression l. rate = 0.01	0	1	Logistic Regression l. rate = 0.0001	0	1	Logistic Regression l. rate = 0.0001	0	1

Table 5.10: APCR and BPCR classifying results using MFSD-MSU database.

CASIAs databases

A comparative between the obtained results of the two CASIAs databases: Image and Videos. In order to do so, KNN classifier is going to be used to compare them.

In figure 5.31 what is represented is the comparison between image CASIA (in blue) and video CASIA database (in green). As it can be seen, the result is much better when just CASIA images databases are used, although CASIA video results are not bad either.

FRAVs databases

Three FRAVs databases are compared in order to discuss if NIR images, in general, help to detect anti-spoofing attacks. In order to do so, and due to the large amount of data,

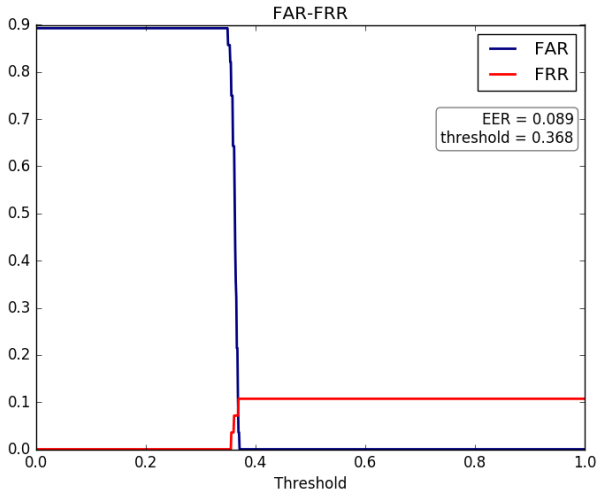


Figure 5.30: FAR-FRR curve of MFSD database using SVM with RBF kernel.

one classifier is chosen to do the comparative: SVM with RBF kernel.

In figure 5.32 the three databases are represented: the RGB database in blue, the RGB+NIR FRAV database classification level in green and the RGB+NIR FRAV database feature level in magenta. The ROC curve with NIR work better, in fact, it is almost perfect.

Figure 5.32, along with the perfect APCR and the BPCR result obtained with logistic regression and the decision tree in the RGB+NIR feature level tends to conduce that NIR database improve the results.

5.4.4 Results comparative with related works

In order to compare results with related work, Equal Error rate (ERR) metric is going to be utilized. The comparative it is going to be made with CASIA database and MFSD-MSU database, because FRAV database results with others methods are not available.

In table 5.11 are exposed EER(%) of the results obtained with CASIA images database and CASIA video database with the literature review; In addition, the EER (%) obtained with the proposed model is presented in the table with the literature review.

The researched work with MFSD-MSU is small and there is not researched work with Deep Learning.

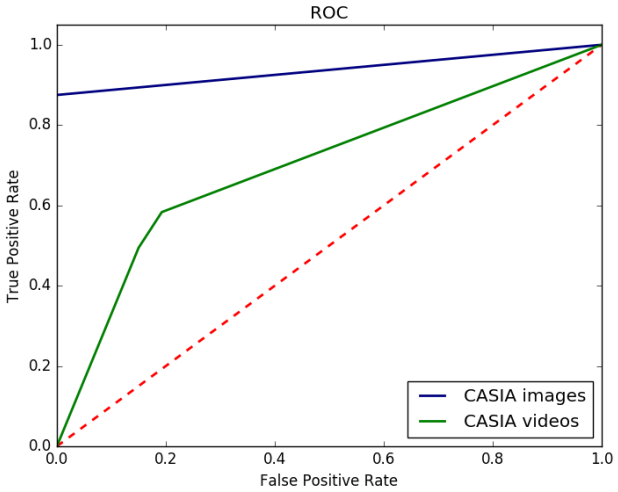


Figure 5.31: Comparative between CASIA databases with KNN classifier.

CASIA database		**MFSD - MSU database**	
CNN model	**EER (%)**	**CNN model**	**EER (%)**
Own CASIA image proposed model	2.5	Own MFSD-MSU proposed model	5.4
Own CASIA video proposed model	11.5	LDA+SVM proposed model in [7]	5.82
CNN model proposed model in [2]	6.2		
LSTM-CNN proposed model in [2]	5.17		
CNN proposed model in [1]	4.64		
LDA+SVM proposed model in [7]	12.9		

Table 5.11: Comparative among proposed models and literature work.

5.4.5 Executing time

In this section, the time that has been necessary to carry out the training and testing processing described in this chapter are exposed.

Each general experiment is formed by the CNN training and the classification, which involves the search for the optimal parameter for each classifier, and the calculation of metrics. There are six general experiments, one per database used :

Process 1: when CASIA image database has been used.
Process 2: when CASIA video database has been used.
Process 3: when RGB FRAV database has been used.
Process 4: when RGB+NIR FRAV database (feature level) has been used.
Process 5: when RGB+NIR FRAV database (classification level) has been used.
Process 6: when MFSD database has been used.

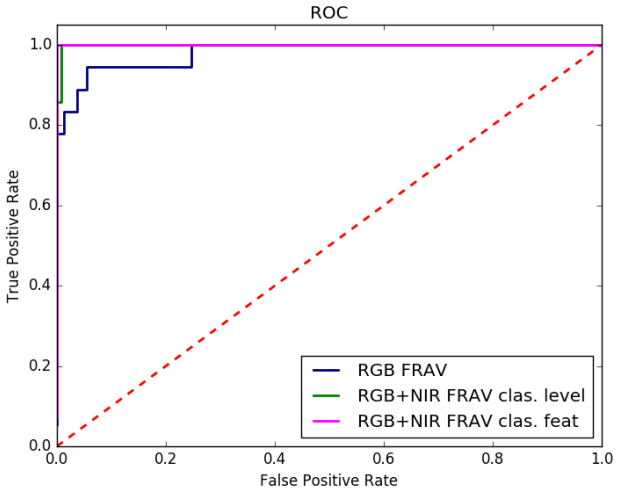


Figure 5.32: Comparative among FRAV databases with SVM RBF.

Process	1	2	3	4	5	6
Executing Time(min)	51.93	152.40	274.87	240.09	432.78	43.95

Table 5.12: Executing time.

In table 5.12 the time (minutes) is exposed for each process. All the processes differ (generally) just in the database, so the differences in time is due to the fact that one database is bigger than other or because it is needed to train two different neural networks as in the RGB+NIR FRAV classification level.

Chapter 6

Discussion on experiments

In this chapter a discussion from the results obtained in experiment chapter is proposed with the purpose of achieving the conclusion of the following chapter.

6.1 Discussion of LeNet-5 and its results

The final architecture has been developed from LeNet-5 architecture whose training, validation and test performance is exemplary as it has been explained in 5.1.2.

The large size of the batch size used in LeNet-5 architecture has been changed to test how the learning process is modified because, in the experiments made with others databases, the number of samples is not enough to preserve the batch size. And results obtained are improved (when batch size is 100) and descend when batch size used is 20.

Usually, in literature, a 32 size is used. And has been concluded that the batch size is fundamentally used to avoid computational resources problems, therefore, the difference of using a smaller batch size is the number of epoch should be increased.

Due to the fact that the Lenet-5 architecture is optimized for MNIST Digit classification, when changes have been made such as modifying activation function or/and adding ReLu layer, results have not been improved, although the difference with respect to the original result is not relevant .

The worst result is obtained when Gaussian weight initialization is used, the performance of the training process suggest that the learning is stuck in a local minimum because the training performance is correct; however, the validation and the test processes return a high error rate.

Regarding using RGB FRAV database with LeNet-5 architecture (detailed in sec-

tion 5.1.4), the purpose of the experiment is testing LeNet-5 with an anti-spoofing face database. The results obtained are not satisfying. In that experiment, RGB FRAV database has been testing using two classes (attacks and genuine users) and five classes (genuine users and each attack correspond with one different class).

In both cases the results are poor and the reason is the architecture is simple and is not able to extract relevant features from images in order to classify them correctly. There is not a learning curve process, the architecture required to be modified; indeed, the learning curve suggest that the learning rate is small because the cost does not decrease, there is no learning.

6.2 Discussion of the own architecture developed experiments

In consequence of the poor LeNet-5 (with the anti-spoofing database) performance, the architecture has been modified (described in section 5.2). Previous works are based on *Imagenet* architecture and its results are respectable, therefore an own architecture based on Imagenet is developed.

The four experiments described to test the architecture behavior, have been realized with the three anti-spoofing databases: CASIA, FRAV and MFSD-MSU databases.

The training process of the general experiments are acceptable because each one converges into a minimum and close to 0, meaning that the training procedure is correct, although there are oscillations and the cost curve is not as perfect as the obtained with original LeNet-5.

The validation process should decrease its value in each epoch converging to a low value (not as lower as the obtained with the cost performance). The validation performance obtained with CASIA image database is not the desirable one because it oscillates changing the error value between 60% and 10%. The validation curve obtained in CASIA video, RGB FRAV database and RGB+NIR (feature level) is preferred, the value descends until converging.

The result obtained in RGB from RGB+NIR (classification level) FRAV database suggest an overfitting, because the cost descends until 0 and the error at validation descends until 0% and then improve until converge in a 10%, what means that the training samples are being learnt.

The validation result obtained with MFSD-MSU does not show a favorable perfor-

mance, the value oscillates between 30% and 40% which are large error values.

The cost results obtained with experiment two and three are as expected, the cost converges in 0 for each database. Despite the fact that the validation error obtained in experiment 2 and experiment 3 is not a desirable result: for each database the error oscillates sharply, experiments 3 converges in 0% error for all databases except for FRAV one (overfitting has been provoked). Experiment 2 result does not converge to low values.

The results obtained in experiment four are not acceptable for any database. The training cost and the validation error get constant values. Decreasing the learning rate is not an option for any database. The training cost curve should decrease slower and converge.

6.3 Discussion of the final experiment

The final architecture built is tested with the all the databases (as is detailed in 5.3).

First, the acquired results, for each database, at the training process are going to be discussed. Training cost and validation error are different from the obtained in section 5.2.3 because weights are initialized (with a Gaussian distribution) randomly, but the seed provided to the random function does not provide the same weight initialization. Also, the number of epoch at training in this last experiment has been increased until 600.

The obtained cost at the training process for each database is correct, because converges in low values. The error at the validation process is not as desirable as the cost. The oscillation is hardly, the convergence is not the expected one in some databases (RGB FRAV and RGB+NIR (feature level) FRAV databases and MFSD-MSU). When CASIA image and video databases have been used, the performance of the validation error curve is more desirable.

The cause of the oscillation problem is the unbalanced databases, more specifically, training with the inequality number of samples in each class would produce the tendency to classify all the samples in the same class.

The consequences of training with unbalance databases is represented by almost each database: all the negative samples are correctly classified but some of the positive samples are incorrectly classified.

CASIA image database just misclassified one sample with all the classifiers, the same positive sample. SVM linear does not work correctly with the features obtained in the

database, so all the samples are classified as negatives. PCA and LDA methods does not improve the performance, the same sample is misclassified.

When CASIA video database is used, the proportion of incorrectly classified samples is increased, BPCR takes higher values than with CASIA image database. Moreover, a small proportion of negative samples is misclassified. When SVM (RBF) and LDA has been used, the results are considered the best, but rest of the models works in a similar way. The worst is obtained when Decision Tree and PCA are used.

With RGB FRAV database, there are the same proportion of misclassified samples as negative samples. The SVM (linear classifier) does not work properly. The rest of the cases, the incorrectly classified samples are repeated in almost every classifier.

When NIR database has been used, results have been improved if images has been added in feature level. When images are added in classification level, results are similar than the obtained in RGB FRAV database.

When RGB+NIR (feature level) FRAV database is used, a perfect classification task has been obtained when Decision Tree and Logistic Regression. In this experiment, using LDA worsen results.

However, when RGB+NIR (classification level) FRAV database is used, using LDA improve results and just one sample is misclassified. The most repeated misclassified sample matches with one of the most incorrectly repeated samples in RGB+NIR (feature level) FRAV database.

Best result obtained for RGB+NIR (classification level) FRAV database is any of the classifiers used with LDA.

Despite of the fact that APCR result is equal to 0, BPCR result is equal to 1 when MFSD-MSU database has been used. That means that each sample has been classified as attack. No matters the classifier or even LDA and PCA has been used. This performance is the worst with respect others databases.

The Equal Error Rate value obtained for all the databases is lower than 1, except for CASIA video database that is 1,15 value; with MFSD-MSU the EER value is the second higher: 0.089. The lowest EER value is obtained when RGB+NIR (feature level) FRAV database is used, because is 0 when Decision Tree and logistic regression classifiers have been used.

6.3.1 Comparative among databases

To compare the results among databases, ROC curve has been used.

Comparing CASIA image and video results, CASIA image outperforms CASIA video results. The performance of the CASIA video ROC curve is poor.

With regards to FRAV databases, the perform of the three databases is outstanding; however, the best performance is obtained with FRAV (RGB+NIR) feature level.

6.3.2 Results comparative with related works

Literature results are summarize in section 5.4.4. CASIA and MFSD-MSU databases are researched in literature; however, MFSD-MSU has been used in one article [7].

The metric used to compare the results is Equal Error Rate, because is the used in literature.

Own CASIA result is the best obtained (with respect literature) when CASIA images have been used, 2.5%, the best result in literature is 4.65% obtained in [1]. However, when CASIA video database is used, the EER is the second worst compared with literature (11.5% EER).

With regard to MFSD-MSU, despite the bad classification performance, the EER obtained in the own model (5.4%) is sligly better that the obtained in [7] (5.84%).

Chapter 7

Conclusions and future work

In this section, the conclusions obtained along the methodology are presented and the future work will be explained.

7.1 Conclusions

As a first conclusion, we can point out the importance of the database. Having a representative and well-built database is essential in order to train any classifier or neural network.

Classifiers or Neural Network training procedure is based on the input training samples, databases are the elementary tool.

In the used databases, it has been proved that the quality of images is important to extract the features better and to minimize the error. The results obtained with FRAV database were better than with the CASIA database and MSU-MFSD database. One main reason is the quality of images. FRAV database has been built with a professional camera, whereas CASIA and MFSD databases are built with a laptop or smartphone cameras.

Apart from the quality of images, a database should be balanced. In other words, the amount of samples in each class should be equivalent or close to it since having a 70%-30% distribution of samples would mean that the classifier starts with 30% error if all samples are classified in the same class. This would mean that training would not be performed as well as if the database were balanced.

Balancing problems occur in used databases and, specially with MSU-MFSD database, the training procedure is not favourable; samples are classified in the same class regardless of the classifier.

To solve balancing issues, the first recommendation would be to add more samples to the class with fewer samples. If this is not possible, the classifier should strongly be penalized when classifying a sample from a class with fewer samples. This way; the training procedure would be more balanced.

The total number of samples affects the training process due to the fact that in Deep Learning techniques, a large quantity of samples (thousands) are required. In the used databases, more specifically the MFSD-MSU database, is composed by only 35 class-0 samples in validation subset and only 2 class-0 samples (12 class-1 samples) in test subset. Consequently, bad results are obtained. Bigger databases should be used.

The quantity of FRAV and CASIA samples is greater than MFSD-MSU, but usually, thousands of samples are used to train neural networks.

rom the results obtained, the FRAV database is the database whose samples are the most correctly classified. The worst results obtained have been with the MFSD-MSU database.

With the purpose of getting better results with MFSD-MSU database, the CNN requires to be modified by trying to change the learning rate or trying to build a balanced database, for example. This is because all the training and testing processes get stuck classifying all the samples as negatives.

It has been concluded that using NIR images with RGB have improved the results until getting a perfect classification at testing. NIR images could detect attacks because printed photos or tablets are similar to real user in a RGB space but not in a NIR space. It has been proved that using complementary devices enhances results.

Regarding the CASIA image and CASIA video databases, results are better when the CASIA images database is utilized. One sample is misclassified in most cases, on the contrary, CASIA videos database is formed by a higher quantity of samples and more samples are erroneously classified.

The best model combination of CNN + classifier is obtained when using the RGB+NIR FRAV database feature level. The result from this is a perfect classification obtained with the Logistic Regression and the Decision Tree.

The results obtained from the RGB+NIR FRAV database classification level are also outstanding, especially when using LDA since just one sample is misclassified, the same one for all classifiers.

SVM with RBF kernel is, in general, the classifier that works best along the databases. Nevertheless, with this classifier the best performance has not been obtained. SVM is a classifier that works well with bi-class problems. Nonetheless, for most of the times, it has not been possible to classify correctly when using SVM with linear kernel. Therefore, it is recommended to use various classifiers as some may work better than others due to the fact that , the performance of each classifier depends on distribution of the feature vector.

Regarding the use of LDA or PCA reduction dimensionality methods, no fundamental conclusion has been obtained. In some cases, one works better than others, so it is not a waste of time trying to use it if the resources are available as results could be improved. Consequently, it is recommended to use it when possible.

For each database, misclassified samples are repeated along the different classifiers used, but a pattern has not been found.

When some samples are misclassified for each classifier, the incorrectly classifier samples are repeated in the different used classifiers, but a pattern of misclassified samples have not been found.

Regarding literature work, the results acquired with CASIA images and MFSD-MSU databases improve the state of the art.

7.2 Future work

For future work, foremost, trying to minimize the unbalanced database, since it is not possible to get more genuine samples, the error would increase when positive samples are incorrectly classified in order to compensate the unbalancing

After modifying the database, if results do not improve, the network architecture should be changed in order to achieve better results.

With the aim of improving the performance of the Convolutional Neural Network, the learning rate will change to a dynamic one, whose value would decrease as the number of epochs increases. Thereby, the training steps during the first epochs would be greater and when the minimum is found, the steps are smaller. It would be necessary to try with different values until obtaining the most suitable one.

Inter-test experiments are important, in future work, training with one database and validating and testing with another is recommended in order to know the adaptation of the Convolutional Neural Network. In the same way, the best model obtained with FRAV

database (the most correctly classified database) could be tested with MFSD-MSU and CASIA databases with the purpose of knowing the adaptation of the trained Neural Network and if results would improve or aggravate.

In addition, other classifiers such as Naive Bayes classifier, Random Forest or Gaussian Mixture model could be utilized in order to have more options.

REPLAY-ATTACK is also a widely used database in literature, it could be used for training and testing the CNN and compare its results with the obtained in this thesis and the literature review.

Bibliography

[1] J. Yang, Z. Lei, and S. Z. Li, "Learn convolutional neural network for face anti-spoofing," *CoRR*, vol. abs/1408.5601, 2014.

[2] Z. Xu, S. Li, and W. Deng, "Learning temporal features using LSTM-CNN architecture for face anti-spoofing," in *3rd IAPR Asian Conference on Pattern Recognition, ACPR 2015, Kuala Lumpur, Malaysia, November 3-6, 2015*, pp. 141–145, 2015.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, pp. 1097–1105, 2012.

[4] M. K. Hani and S. S. Liew, "A convolutional neural network approach for face verification," in *International Conference on High Performance Computing & Simulation, HPCS 2014, Bologna, Italy, 21-25 July, 2014*, pp. 707–714, 2014.

[5] I. Chingovska, A. Anjos, and S. Marcel, "On the effectiveness of local binary patterns in face anti-spoofing," Idiap-RR Idiap-RR-19-2012, Idiap, 2012.

[6] L. Sun, G. Pan, Z. Wu, and S. Lao, *Blinking-Based Live Face Detection Using Conditional Random Fields*, pp. 252–260. Springer Berlin Heidelberg, 2007.

[7] D. Wen, H. Han, and A. K. Jain, "Face spoof detection with image distortion analysis," *IEEE Trans. Information Forensics and Security*, vol. 10, no. 4, pp. 746–761, 2015.

[8] J. Wayman, A. Jain, D. Maltoni, and D. Maio, *An Introduction to Biometric Authentication Systems*, pp. 1–20. Springer London, 2005.

[9] M. Baca, M. Schatten, and J. Seva, "Behavioral and physical biometric characteristics modeling used for its security improvement," 2010.

[10] A. F. Abate, M. Nappi, D. Riccio, and G. Sabatino, "2d and 3d face recognition: A survey," *Pattern Recognition Letters*, vol. 28, no. 14, pp. 1885 – 1906, 2007.

[11] A. K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 4–20, 2004.

[12] J. Galbally, S. Marcel, and J. Firrez, "Biometric antispoofing methods: A survey in face recognition.," *IEEE Access*, vol. 2, pp. 1530–1552, 2014.

[13] "Take biometric security systems to the next level." http://www.rechters.pl/take-biometric-security-systems-to-the-next-level/. Accessed: 2017-05-04.

[14] K. D. I and M. Grgic, "A survey of biometric recognition methods," in *Proceedings. Elmar-2004. 46th International Symposium on Electronics in Marine*, pp. 184–193, 2004.

[15] D. Wen, H. Han, and A. K. Jain, "Face spoof detection with image distortion analysis," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 4, pp. 746–761, 2015.

[16] N. Erdogmus and S. Marcel, "Spoofing face recognition with 3d masks," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 7, pp. 1084–1097, 2014.

[17] D. Kriesel, *A Brief Introduction to Neural Networks*. 2007.

[18] R. Rojas, *Neural Networks: A Systematic Introduction*. Springer-Verlag New York, Inc, 1996.

[19] M. F. Bear, B. W.Connors, and M. A. Paradiso. Williams & Wilkins, 1996.

[20] B. J. A. Kröse and P. P. van der Smagt, *An Introduction to Neural Networks*. The University of Amsterdam, 8th ed., 1996.

[21] I. Basheer and M. N. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application," *Journal of Microbiological Methods*, vol. 43, no. 1, pp. 3 – 31, 2000.

[22] H. B. Demuth, M. H. Beale, O. De Jess, and M. T. Hagan, *Neural Network Design*. USA: Martin Hagan, 2nd ed., 2014.

[23] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000.

[24] *Analysis of Deep Convolutional Neural Network Architectures*, vol. 21, University of Twente, 2014.

[25] D. Eigen, J. T. Rolfe, R. Fergus, and Y. LeCun, "Understanding deep architectures using a recursive convolutional network," *CoRR*, vol. abs/1312.1847, 2013.

[26] C. Farabet, R. Paz, J. Perez-Carrasco, C. Zamarreos, A. Linares-Barranco, Y. LeCun, E. Culurciello, T. Serrano-Gotarredona, and B. Linares-Barranco, "Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel convnets for visual processing," *Frontiers in Neuroscience*, vol. 6, p. 32, 2012.

[27] "Sample digits of mnist handwritten digit database." https://www.researchgate.net/figure/264273647_fig1_Fig-18-0-9-Sample-digits-of-MNIST-handwritten-digit-database. Accessed: 2017-04-19.

[28] "Automated border control gates for europe." http://abc4eu.com/. Accessed: 2017-04-19.

[29] Z. Zhang, J. Yan, S. Liu, Z. Lei, D. Yi, and S. Z. Li, "A face antispoofing database with diverse attacks," in *5th IAPR International Conference on Biometrics, ICB 2012, New Delhi, India, March 29 - April 1, 2012*, pp. 26–31, 2012.

[30] A. M. Martinez and A. C. Kak, "Pca versus lda," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 23, pp. 228–233, 2001.

[31] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review," *Journal of Biomedical Informatics*, vol. 35, no. 56, pp. 352 – 359, 2002.

[32] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[33] "Opencv - undestanding svm." http://docs.opencv.org/3.1.0/d4/db1/tutorial_py_svm_basics.html. Accessed: 2017-04-24.

[34] D. Ciobanu, "Using svm for classification," *Acta Universitatis Danubius. OEconomica*, no. 5(5), pp. 209–224, 2012.

[35] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," tech. rep., Department of Computer Science, National Taiwan University, 2003.

[36] "Scikit-learn - decision tree." http://scikit-learn.org/stable/modules/tree.html. Accessed: 2017-04-28.

[37] D. Stutz, "Understanding convolutional neural networks," tech. rep., Fakultt fr Mathematik, Informatik und Naturwissenschaften, 2014.

[38] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, f-score and roc: A family of discriminant measures for performance evaluation," in *Proceedings of the 19th Australian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence*, pp. 1015–1021, 2006.

[39] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 233–240, 2006.

[40] "The area under an roc curve." http://gim.unmc.edu/dxtests/roc3.htm. Accessed: 2017-04-19.

[41] "Sc37 iso/iec jtc1," 2014.

[42] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, pp. 2278–2324, 1998.

[43] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *CoRR*, vol. abs/1206.5533, 2012.

[44] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*, 2010.

List of Figures

List of Tables