‘Learning convolutional Neural Network for Face Anti-Spoofing”

Beatriz Gómez Ayllón

May 28, 2017

Abstract

This thesis proposes an algorithm based on deep learning whose finality is being able to detect face spoofing attacks.

CASIA, FRAV and MFSD-MSU databases would be 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.

With the purpose of getting the main objective, a Convolutional Neural Network (CNN) would be designed and build for this specific problem. Furthermore, the best suitable classifier would be discussed and selected.

The Convolutional Neural Network architecture would be developed from LeNet-5 architecture. The steps and decision made to get the final architecture of the Convolutional Neural Network are explained likewise the personalization of each classifier.

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

Classifiers and dimensionality reduction algorithms would be personalized for each database and each experiment.

Python has been used to succeed the thesis object as the programming language. Theano has been preferred as the main library to develop the convolutional neural network architecture and training and testing processes.

From the results obtained, could be concluded that the most correctly classified database is RGB+NIR (added in feature level) FRAV database by reason of the perfectly classification result.

Acknowledgements

In first place I would like to share my sincere gratitude to the director of this thesis, Dr. Aristeidis Tsitiridismy, for the continuous support, his extensive patience, and the advice that have guide me during the development of the thesis and the research work and have 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.

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

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

Contents **vii**

1 Introduction **1**
1.1 Motivation 1
1.2 Objectives and contributions 2
1.3 Thesis structure 3

2 Literature Review **5**
2.1 Related Works 5

3 Background Theory **7**
3.1 Anti-spoofing and biometrics systems 7
3.1.1 Historical introduction 7
3.1.2 Biometrics 8
3.1.3 Biometric system 8
3.1.4 Spoofing 9
3.1.5 Face anti-spoofing 10
3.2 Neural Network background theory 12
3.2.1 Historical introduction 12
3.2.2 Introduction to ANN 13
3.2.3 Biological Neural Networks 14
3.2.4 Artificial Neural Networks (ANN) 14
3.2.5 Convolutional Neural Network (CNN) 19

4 Methodology **21**
4.1 Programming language and frameworks 21
4.2 Databases 22
4.2.1 MNIST digit database 22
4.2.2 FRAV dataset 23
4.2.3 CASIA dataset 25
4.2.4 MSU - MFSD database 27
4.3 Classifiers, Reduction of the dimensionality algorithms and Cross Validation 28

4.3.1 Classifiers . . . 28
4.3.2 Dimensionality reduction algorithms . . . 31
4.3.3 Cross Validation . . . 32
4.4 Metrics . . . 32
4.4.1 Cost and Error rate . . . 33
4.4.2 True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN) . . . 33
4.4.3 ROC curve and Precision and Recall curve . . . 34
4.4.4 APCER and BPCER . . . 36

5 Experiments **37**
5.1 LeNet-5 . . . 37
5.1.1 LeNet-5 specifications . . . 38
5.1.2 LeNet-5 Results . . . 39
5.1.3 Modifying LeNet . . . 40
5.1.4 LeNet-5 and RGB FRAV faces database . . . 47
5.2 Adapting LeNet-5 architecture . . . 48
5.2.1 New Architecture . . . 49
5.2.2 Databases . . . 50
5.2.3 Experiments Description . . . 50
5.3 Final architecture . . . 52
5.3.1 Architecture . . . 52
5.3.2 Database . . . 52
5.3.3 Classification . . . 53

6 Results **59**
6.1 Training and validating process . . . 59
6.1.1 CASIA Image database . . . 59
6.1.2 CASIA Video database . . . 59
6.1.3 FRAV database . . . 60
6.1.4 FRAV RGB+NIR (feature level) database . . . 61
6.1.5 FRAV RGB+NIR (classification level) database . . . 62
6.1.6 MFSD database . . . 62
6.2 Testing process . . . 63
6.2.1 CASIA Image database . . . 64
6.2.2 CASIA Video database . . . 65
6.2.3 FRAV database . . . 66
6.2.4 FRAV RGB+NIR (feature level) database . . . 67
6.2.5 FRAV RGB+NIR (classification level) database . . . 68
6.2.6 MFSD-MSU database . . . 69
6.3 Comparative among databases . . . 70
6.3.1 FRAVs databases . . . 70

6.3.2 CASIAs databases 70
6.4 Executing time 71

7 Conclusions and future work **73**
7.1 Conclusions 73
7.2 Future work 74

Chapter 1

Introduction

In this chapter, the reasons that have inspired to effort in developing this work are exposed. The thesis objectives and this document structure are described as well.

1.1 Motivation

Artificial intelligence (AI) is a remarkably researched field which in recent years is having a lot of importance and it is expanding. Because of my interest in AI, I decided to study an Artificial Vision Master and the most that I would like to learn is Machine Learning, more exactly, Deep Learning.

How bio-inspired algorithms (as Artificial Neural Networks) learn from the input data and classify with a magnificent accuracy the test samples or how Neural Networks are able to create music and pictures impressed me and is a knowledge that I would like to get.

The Human body as an identification key is fast, private identification and recognition systems. Biometrics is a technology that is currently being implemented in society and its presence is being increasing. At the same time as biometric systems increase its potential, attacks are more powerful. From the acquired knowledge along 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 caught my interest.

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

FRAV group are currently working on *ABC4EU European Project*, where face biometric is utilized. Learning from a project that is currently being developed with technology

that interested me, is a generous opportunity. And developing a thesis with them and with biometrics and Deep Learning is an excellent chance that has encouraged me.

1.2 Objectives and contributions

Objectives

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

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

Next objective is to build a Convolutional Neural Network architecture which is adapted for this thesis and would use to extract the feature of databases images.

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

Next objective is selecting the most correctly database from the compared and analyzed results.

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

Contributions

This thesis contribute with a Convolutional Neural Network architecture and a classifier which optimize the classification.

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

The next contribution is the argument among the results of each database with the selection of the database 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 this thesis main objectives are described.

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

Before go ahead to the development of the main part, in chapter 3, a short introduction to neural network is presented as well as anti-spoofing definitions.

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

Experiments that have been realized are specified in chapter **??**.

In chapter 6, the results obtained in previous chapter 4 are presented and discussed.

In the last chapter 7, the conclusions obtained along the thesis are explained and the future work as well

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.

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 as technology advance, capture systems and processing algorithms have proceeded too. This open an immense quantity of options and one of them is 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 characteristic or psychological behavior in order to identify a person.

3.1.1 Historical introduction

There were in the 1870s when the necessity of identifying people getting physical characteristics appeared. Alphonse Bertillon had the desire of identifying jail prisoners, therefore, to satisfy this need, the skull diameter, arm and foot length were utilized in the USA until the 1920s [**?**].

There were in the 1880s when fingerprint and facial identification were proposed. With the appearance of digital signals processing systems in 1960, the voice and the fingerprint biometric systems were started to be investigated and researchers started to think to use this system to people identification in access control security [**?**].

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

the 80s and after a short time the face systems [**?**].

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

3.1.2 Biometrics

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

Biometrics could be distinguished as physical or behavioral biometrics. Physical biometrics are characteristics which every human are born with and are genetic, fingerprint, face, DNA, ear and iris are physical biometrics; behavioral biometrics are characteristics which have been matured with person, are psychological, signature and gait are behavioral biometrics [**?**].

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

3.1.3 Biometric system

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

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

Recognition or Identification system: user data is compared with the all the template database to find user's identity. The identity is not claimed by the user and is given by the system in function of features. Is a one-to-all comparative.

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

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

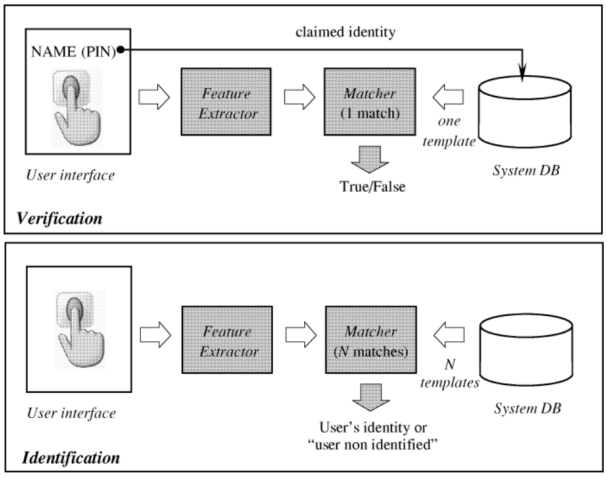


Figure 3.1: Verification and Identification system diagram. Image obtained from [?]

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

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

Database: the database is formed by templates which are going used as true samples to compare the obtained with 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, adding the biometrics in feature or score level. It is labeled as multimodal and it is used to complement vulnerabilities of biometrics [**?**].

3.1.4 Spoofing

As the same time as biometrics authentication systems appeared, the attacks to trick systems emerge and they are called *spoofing attacks*. For example making plaster molds for geometric hand biometrics or fingerprints would be an attack as the presented in figure 3.2 (image obtained from [**?**]).

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

So as to get the tricks and do not allow spoofing attacks, *anti-spoofing* tries to minimize the attacks or prevent them.

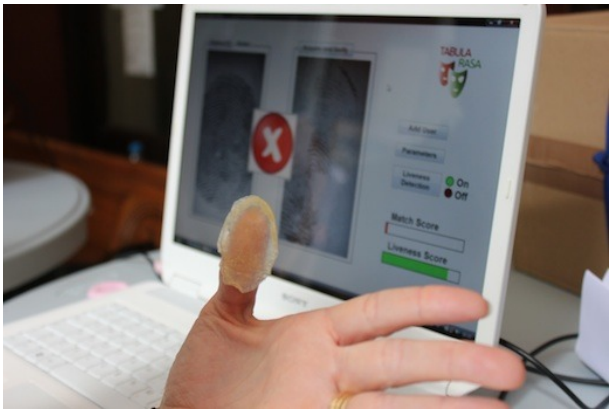


Figure 3.2: Spoofing fingerprint. Image obtained from [?]

Anti-spoofing methods could be distinguish depending on the biometric system module which is combined [**?**]:

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 software decided if it is a genuine or fake user.

Score level: after the feature extraction, when the scores are obtained, the biometric system decide 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 is evaluated two methodologies are distinguished [**?**]:

Algorithm-based or technology evaluation: Evaluate algorithms that prevents anti-spoofing.

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

In this project, it is going to be developed the algorithm to detect anti-spoofing attacks when the face is using 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 very used biometric system because of its attributes since is not intrusive, is easy capturing images and it is considerably accepted by people as biometrics.

The principal or main used acquire system are visible light cameras, in spite of other cameras as infra-red or depth camera could be used too. Moreover, using different types of camera could help to detect anti-spoofing attacks.

The disadvantages of cameras as sensor are that the illumination need to be controlled and images from different angles could not be suitable, in addition, people expressions or face occlusions are complicated to work with [**?**, **?**]. On the contrary, the main advantage of using a visible light camera as sensor is the cost, that is very low.

The location of verification or identification system could be in a static and specific place as the office entry or could be a wide place as a subway or an airport [**?**].

Attack to this biometric system is also easy to generate, people face images are accessible in social networks. The spoofing attacks could be 3D attacks if 3D masks are used; printed photos, 2D mask and displaying an image or video in a smartphone or tablet are considered 2D attacks [**?**]. 2D attacks cost cost is not high [**?**] and lighter that 3D spoofing attacks cost.

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

In figure 3.3 (image obtained from [**?**]) are represented 3D resin faces mask. This masks are used for a 3D face mask database, but are an example of face spoofing attack. 3D mask could be made of silicone too and are cheaper that resin mask.

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

To argue spoofing attacks, feature level is the most studied method. Literature separates into two groups, dynamic and static approach [**?**]:

Dynamic: is based on motion to detect anti-spoofing, for instance, blink eyes or optical flow assessment, 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, are being researched and could be used with image and video sequences [**?**].

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 [**?**, **?**].

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

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

In general, the explained methods works with 2D face anti-spoofing although 3D anti-spoofing is present too using as a sensor depth cameras [**?**]. 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 by first time in the years 40 by Warren McCulloch and Walter Pitts, more specifically, simple models of neural networks (switches based on neurons) were able to calculate almost every arithmetic problems and logic operations. A few years later, the recognition of spatial patterns field was defined as an interesting scope for neural networks [**?**].

In 1949 the 'Hebbian rule' was postulated by Donald O. Hebb. The rule describes the generalized learning basis of neural networks. This rule explain 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 [**?**].

From 1951, the golden age of neural networks started. 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'* as was developed between 1957 and 1958 [**?**].

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, this was 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, researched figured out that the XOR function was not able to be solved with just one perceptron [**?**].

After 1960, neural networks golden age finished and the importance of this researched field decreased until computational resources were enough, but 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 [**?**].

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

3.2.2 Introduction to ANN

Humans, along the history, have tried to reproduce nature. Evolution has turned out to be a big coordination in nature. Object recognition, associating concepts, memorize or extracting the semantic of images are competences that humans are capable. These processing information tasks are being investigated and it is now when technology is being more accurate; nevertheless, 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 [**?**].

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 effective to process information in different ways by itself [?].

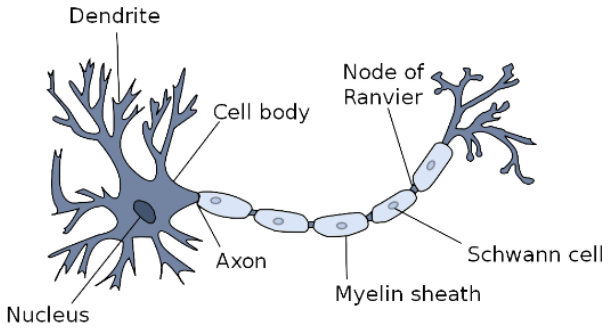


Figure 3.4: Biological Neural Network. Image obtained from [?]

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 neural is shown and its components are signalized [?].

The soma is the spherical central part of the nerve cell in whose there is a salt and potassium concentration which is covered by the neuronal membrane. Inside the soma is the neuronal nucleus and from the soma, branches are extend (dendrites). Nerve cells are connected among them by dendrites. Neurons are transmitting and receiving nervous signals continuously, communicating among them. The information transference is produced, more specifically, in the synaptic clef (space between two neurons connection). 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 [?, ?]

3.2.4 Artificial Neural Networks (ANN)

As the same way 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 [?]

Basically, there are three groups of artificial neurons:

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

Hidden Units, if 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 user desire. The connectivity and the topology of the layers define the topology of the network.

Output neurons, if belong to the output layer. these neurons gives the user the processed information.

In figure 3.5 the schematic of a general neural network it 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.

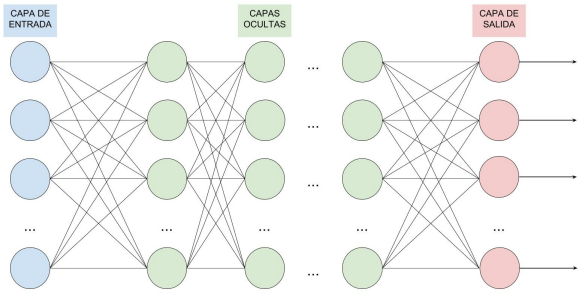


Figure 3.5: Schematic of a general neural network.

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 that the value of the bias change. Weight and bias values could be modified. The output of the neuron is associated to an activation function.

The simplest example of a neural network is formed by two inputs neurons and a output neuron as it is shown 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 sand the activation function (or transfer function) F [**?**].

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

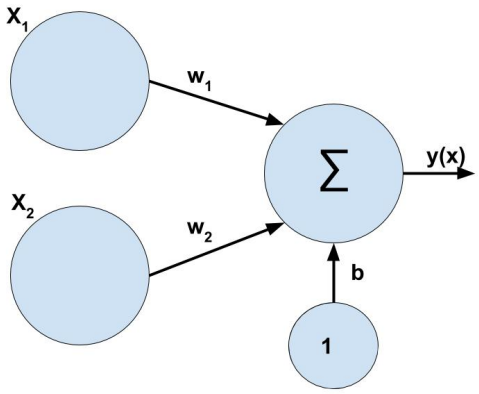


Figure 3.6: Simplest neural network architecture.

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 [**?**]) the quoted functions are shown. 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.

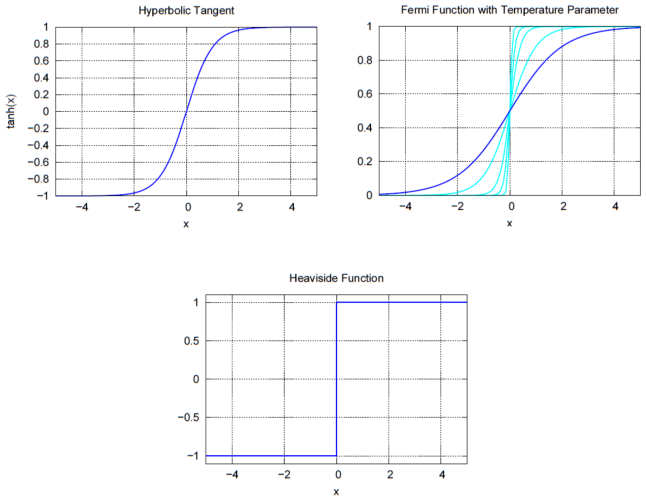


*Figure 3.7: Different activation functions. Image obtained from [**?**]*

Analogies between ANN and Biological Neural Networks

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

Biological neurons corresponds to artificial neurons is the most discernible analogy. The cell body correspond with transference function. The output to others 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 [**?**]. Artificial neural networks are inspired in Biological neural networks, and thus both work in a different (although inspired) way.

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

*Table 3.1: Analogies between artificial neural networks and biological neural networks [**?**]*

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 [**?**].

There are three main different learning procedures [**?**, **?**]:

Supervised Learning: the input training data is given with its target (the correct result of its corresponding sample), that means that the train samples has associated 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 as 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 with respect to 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.

Neural network learning procedures have in common 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 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: [**?**].

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

One of the most used supervised techniques of neural network learning is the back-propagation, this learning rule is based on the gradient descent. In general, this rule, from a randomize initialization of weights, the error would be minimized changing the weights in the direction of the gradient descent:

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

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 [**?**, **?**, **?**]. 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 elected 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, as the same time as samples are received, they are used once to train.

Type of Layers

The layers that conforms 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 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 to resume its out [**?**]. The most used pooling layer is the max-pooling layer, the maximum values are saved.

Normalization layer: This layer normalize the activities of the neurons, because of that the processing time could be reducing during training or testing. There are two types of normalization layer: Local Response Normalization (LRN) and 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 'turn 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"** [**?**].

Fully-connected layer: 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 so, it is used ahead classification.

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

3.2.5 Convolutional Neural Network (CNN)

Convolutional neural networks are a determinate typology of neural network, CNN are inspired in how the visual cortex of a cat works [**?**]. 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 used successfully in recognition and classification task including document and object recognition, face detection, robotics navigation, etc. [**?**, **?**].

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 yo 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 of 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 show 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 tree face databases has 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 with 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 a image database of human written digits. This database is frequently used to learn machine learning techniques. Because of that, this 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

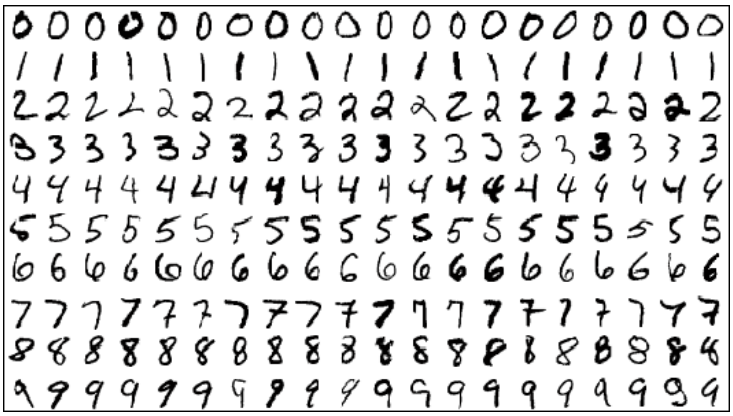


Figure 4.1: MNIST digit images database. Image obtained from [?]

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

– There are 70.000 number of unique samples.
– Despite of 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 [**?**].

One example of RGB images of FRAV database are shown in figure 4.2 and another example could be seen in 4.3. In both images, the four attacks described previously and the real user could be visualized.

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

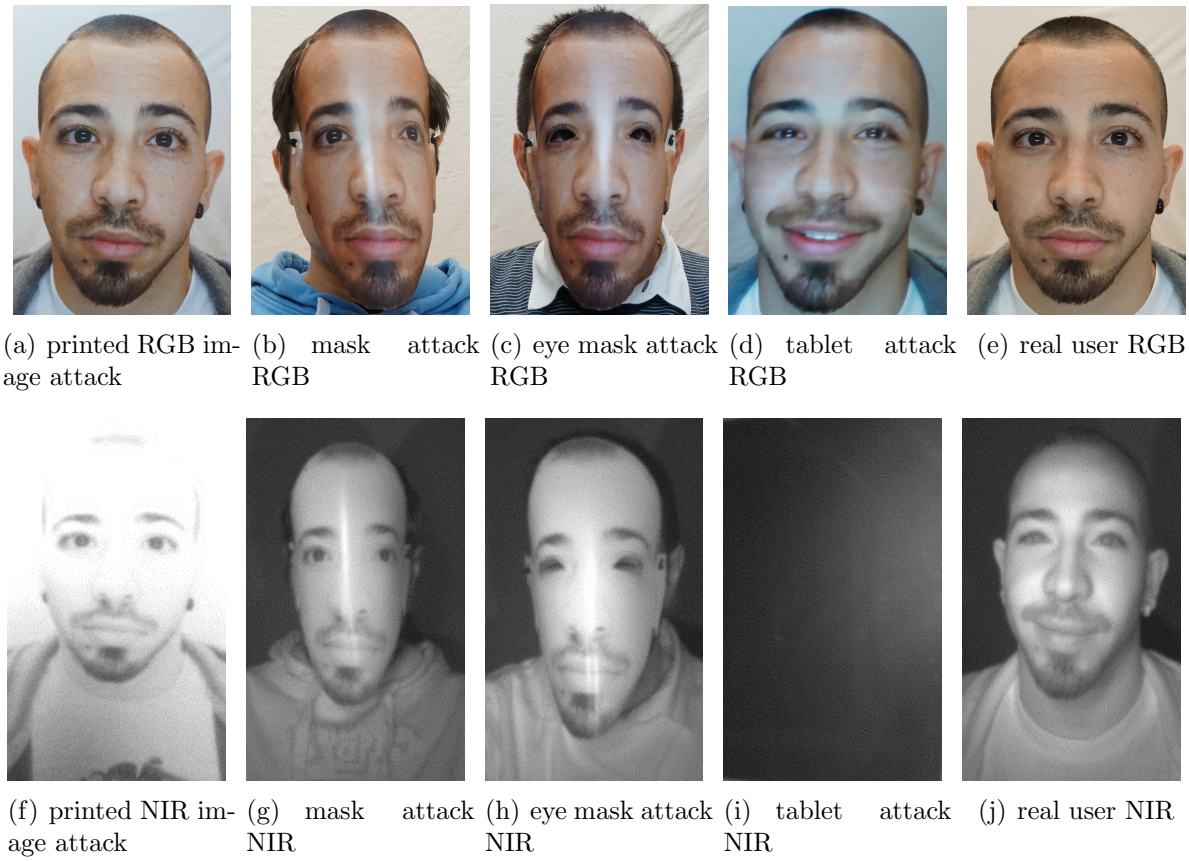
Figure 4.2: Four attacks and real user from RGB FRAV database

As could be seen in figure 4.2 and figure 4.3, five different classes composes this database. One class is the real user class and the other four classes are four spoofing attacks per user:

– 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.



(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

- 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.

Images of classes can be found in RGB and NIR (not all RGB images has its corresponding NIR image). Characteristics of FRAV images database are the following ones:

- There are 939 people in each RGB class or 195 in each NIR class.
- There is one image per person.
- Each image has its own shape.
- As it real user has all the four attack, all the classes has the same number of samples.
- The faces are centered in the image.

This database is used in two different ways:

1. Using only RGB images (figure 4.2), where there are 933 people, in which each person would have a genuine image and four attack, so 4665 samples are available.

2. Using the RGB and NIR images, where there are 195 people which correspond with NIR images and its corresponding images in RGB. 975 samples are available.

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

– 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 has three layers, one per each primary color). The network is feed with the resultant images like other times.

– Classification level: after the network training and before feeding the classifier, RGB and NIR 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 build, 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 two classes have been used; and the second way where each attacks correspond with a class, so five classes (4 attacks and 1 real) have been used.

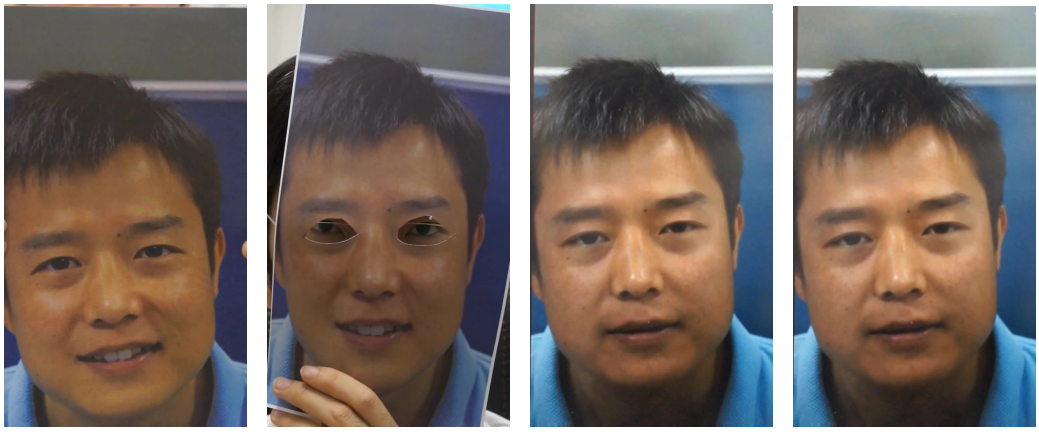
4.2.3 CASIA dataset

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

A person of CASIA database could be seen in figure 4.4, where three attacks types could be seen (figure 4.4(a), 4.4(b) and 4.4(c)) with the real user (figure 4.4(d)).

In the same way as FRAV dataset, this 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 a 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 a independent class, it would be four different classes, the real user class and three attack classes.

This database has been used in different articles [?, ?, ?, ?] it is a very used databased for face biometrics.

4.2.4 MSU - MFSD database

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

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 attack where a video is Replayed represented in figure 4.5(a).
- Smartphone 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 one, but just one image 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 centred in images.
- The size of each image are not equal. Approximately images are 300 pixel heigh and 335 pixel width.

4.3 Classifiers, Reduction of the dimensionality algorithms and Cross Validation

Classifying is called to the task of sign a a category to a object. The classification task is based in the features of the object obtained of the feature extractor [**?**], that in the particular case of this thesis is the output of a convolutional neural network.

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 [**?**], 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 [**?**]:

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

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:

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

That is that the 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 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 [**?**]. 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.

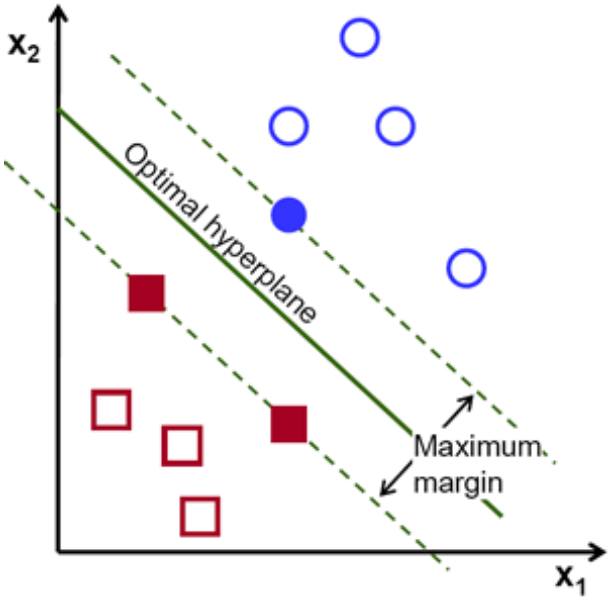


*Figure 4.6: Optimal hyperplane and the decision boundary. Image obtained from [**?**]*

In figure 4.6 the optimal hyperplane between two classes are represented with its corresponding margin. In this example the two classes are well differentiated. This image has been obtained from [**?**].

Based on estimate the hyperplane that the distance between classes, the closest vectors of each class, is maximized [?, ?]. 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 [?, ?]. There are many kernels (polynomial, sigmoid, etc.), but the two used are described:

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

K Nearest Neighbours

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

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

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 classifiers 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 [?]:

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

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, 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 [?]. It could be used as a binary classifier or with k classes.

The input data is split to maximize its separation, resulting a tree structure [?] as is described in figure 4.7 (image obtained from [?]). 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.

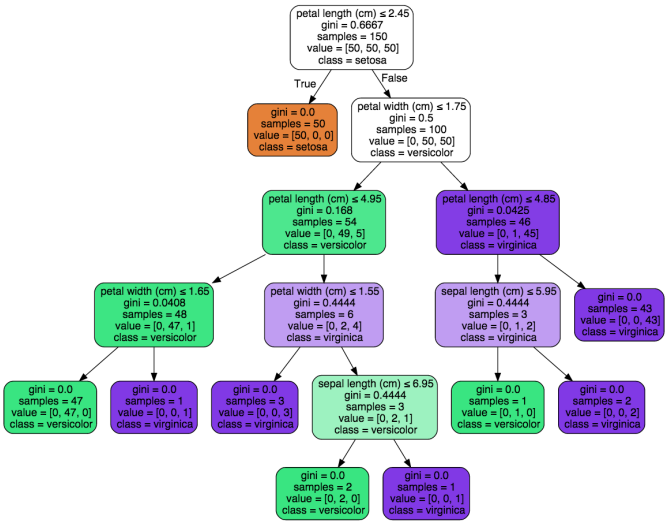


Figure 4.7: Decision Tree Classifier. Image obtained from [?]

4.3.2 Dimensionality reduction algorithms

The objective of those algorithms is transform 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 [**?**], 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, that is that LDA pretend 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 [**?**].

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

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

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 [**?**]. PCA faces the problem of reducing the n dimensional samples vector to a single vector X_0. The X_0 vector would be the result of the sum of the squared distances between X_0 and various features X_k is the smallest [**?**].

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

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 use 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 [**?**].

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 a positive or negative samples is correctly or incorrectly classified [**?**]. The classified sample or predicted is compared with it 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 [?, ?]:

Real / Classified	Positive	Negative
Positve	TP	FN
Negative	FP	TN

Table 4.1: Confusion Matrix

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

4.4.3 ROC curve and Precision and Recall curve

From the confusion matrix, it is possible calculate others parameters [?]: precision, recall, specificity, accuracy 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 [?]:

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

  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 [?]. This parameter could be known as Recall too:

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

  Where:

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

- Precision is defined as the proportion of real positive samples that has been classified as positive [?, ?]:

$$precision = \frac{TP}{TP + FP} \tag{4.15}$$

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

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4.16}$$

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) [**?**].

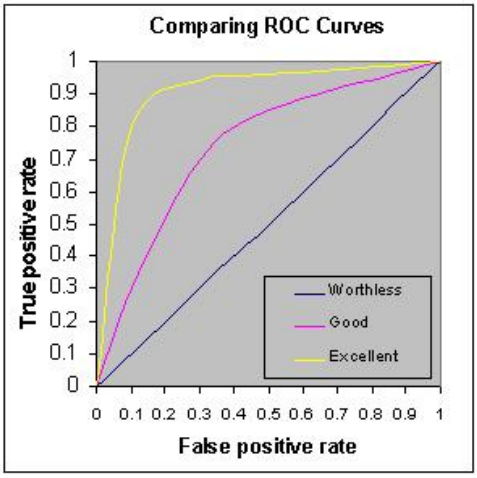


Figure 4.8: ROC curves. Image obtained from [?]

The figure 4.8, obtained from [**?**], it is shown a ROC graph in which three curves are shown. 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 should be improved significantly.

The Precision and Recall curve is the representation of the Precision and the Recall in the same graph. The desired behaviour of a classification system is a high recall (1) and a high precision (1) because that would mean that predictions made by the classifier are correct.

4.4.4 APCER and BPCER

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

This 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.)

$$APCER_{PAIS} = \frac{1}{N_{PAIS}} \sum_{i=1}^{N_{PAIS}} (1 - Res_i) \tag{4.17}$$

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

$$BPCER = \frac{\sum_{i=1}^{N_{BF}} Res_i}{N_{BF}} \tag{4.18}$$

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 shows 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.

Chapter 5

Experiments

In this chapter the start and the evolution of the built convolutional neural network used is described.

5.1 LeNet-5

LeNet-5 [?] 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.*

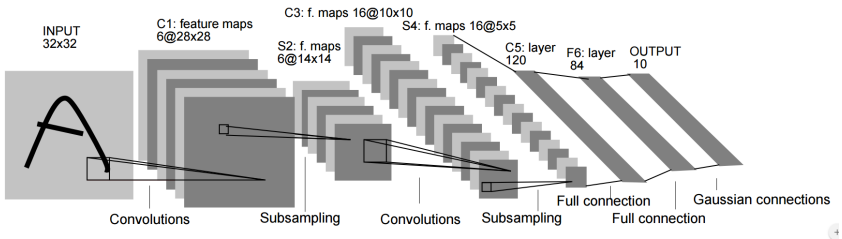


Figure 5.1: LeNet-5 Arquitecture

The basic architecture of LeNet-5 is two convolutional layer, followed each one 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 beginners users to learn deep learning matter because of its short architecture and it is implemented in lots of deep learning framework using it to explain the framework. Because of this, LeNet-5 has been used as basis of the project and to learn Theano and convolutional

neural networks theory and implementation.

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

5.1.1 LeNet-5 specifications

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

The classifier which has been used is the logistic regression which is trained at the same time as the convolutional neural network. The activation function of the convolutional layers and the logistic regression is 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 comes split in three subsets: training, testing and validating. Each subset is used for training, testing or validating respectively.

The data is not fed to the network in one go, each subset is grouped in smalls subsets called batches and whose size is chosen by 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 batches, if there are some 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 of using batches is to define the size of the network and because, usually, the quantity of samples used in deep learning is big (thousand, millions..) and too much memory would be need 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 realized for too many epochs as users has selected and returns the cost of the procedure. While the training is being 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 behavior 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 realized and the results are the best obtained in the whole process until that iteration. The result that is used to compared with others classifiers or with others articles.

In addition to the number of epoch, other way to stop the training procedure is early-stopping, this method is used to avoid over fitting tracking the validation process [**?**]. The decision of stopping the training depends on *the patience* and is chosen by user.

5.1.2 LeNet-5 Results

While the training is being calculate, the weights are being update in each iteration. When A model is selected or saved, weights are actually 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 represented. So 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 at training obtained 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, because it is not oscillate abruptly and converges in a very low value logarithmically.

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

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

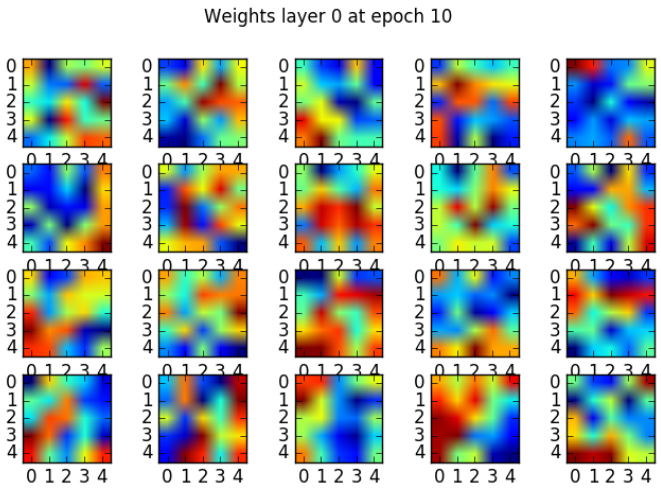


Figure 5.2: Weights at epoch 10 of the first convolutional layer

5.1.3 Modifying LeNet

Modifications have been made to LeNet-5 architecture. First the batch size has been changed and then the activation function, a normalization layer has been added and the weight initialization has been changed too. The database used for those 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 change too with respect to LeNet-5 with the original batch size (500).

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

The validation error for both experiments and the original LeNet are represented in figure 5.5. From images 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

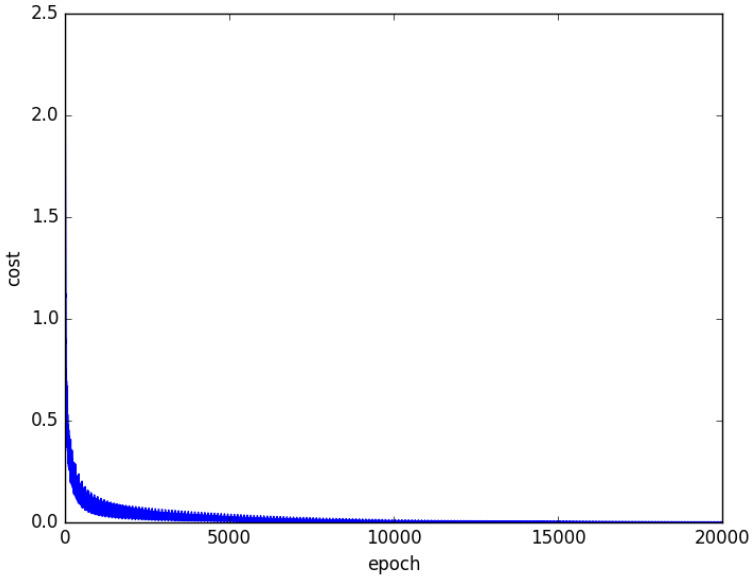


Figure 5.3: Cost function at training running LeNet-5 with MNIST digit database.

the batch size is equal to 20, the optimal test error rate has been obtained in the first 15^{th} epochs, the same error rate than in the original case (0.92%), but 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.

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

In figure 5.5 the error in each epoch is represented for 500 batch size, the original size, for a value of 20 and 100. In the original case, the error starts with a value of 9% approx. with the batch size = 20 the error in the first iteration is about 2.4%, and with a bunch 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, the difference between those examples is that each one get to that conclusion into different epochs. With a batch size equal to 100, the code stopped because of the early-stop with a patience of 10000; it stop in epoch 33,while those epochs, it has been possible to get to a test error of 1.04% in iteration 8500 when the validation error was 1.01%, it has been running with put getting a better validation score for 17 epoch. With a batch size = 20, the code also has stopped earlier because of the same reason, but in that case it has been possible to get to the same minimum that with the original size; the epoch in which has

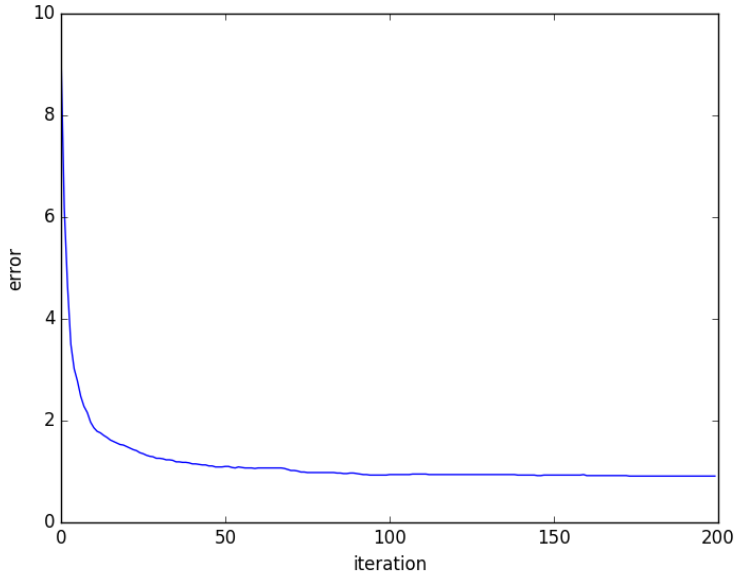


Figure 5.4: Validation error obtained with LeNet-5 with MNIST digit database

stopped is 31, it has been running without getting a better validation score for 15 epochs.

Changing activation function, normalization and weights initialization

As the same way as the batch has been changed and its results has been compared in the previous subsection, the activation function has been changed, a normalization layer has been added and weights initialization has been changed.

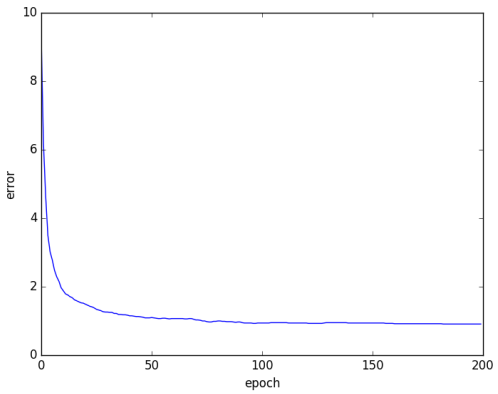
LeNet-5 does not use any normalization layer, but in this experiment from the different normalization availables (batch normalization, local normalization, etc.) Local Response normalization has been added after the max-pooling layers.

The activation function used in LeNet-5 is tanh, it has been changed to rectified linear unit (ReLu) activation function.

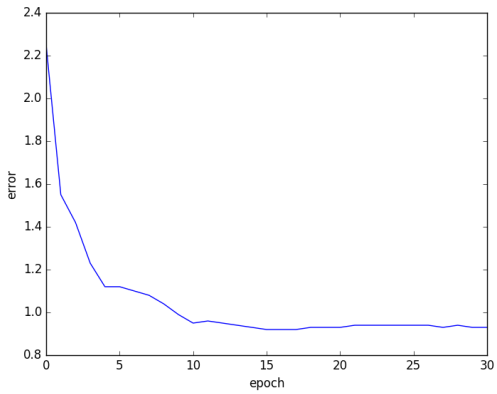
With respect to the weight initialization, in LeNet, for the convolutional layers and the fully connected layer, a normalized initialization [**?**] is used:

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

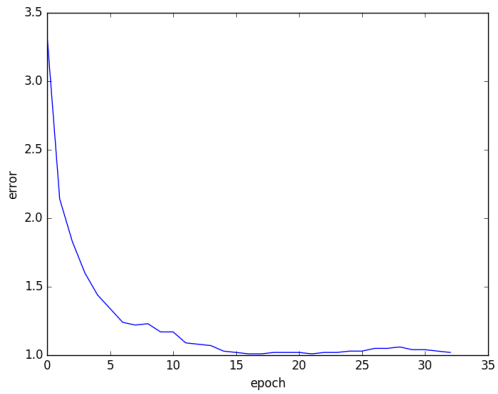
Where n_j is the number of neuron of the current layer and n_j is number of neurons of



(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.

the following layers. In this experiment, this initialization has been changed to a weight initialization with a Gaussian distribution.

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

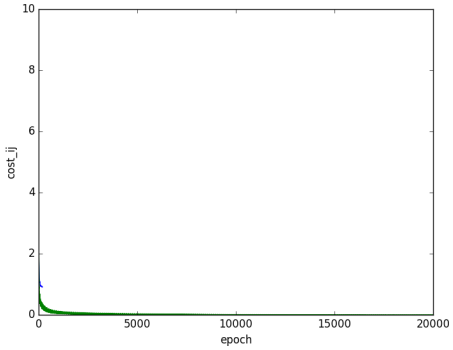
LRN has been used as normalization layer.

– Experiment 4: changing weights initialization: Weights initialization has been changed by Gaussian. In which mean value that has been used is 0 and std is 0.01. Weights initialization has been changed in convolutional and fully connected layers. Also, bias initialization has been changed by ones.

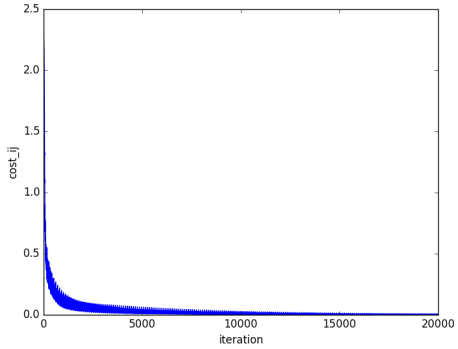
First, the cost of training process are going to be visualized with the original cost train, without modifying LeNet). In figure 5.6 is represented. Also the validation error ($validationError = validationCost * 100$) could be visualized in figure 5.6. The network has been running for 200 epochs.

Visualizing the loss during the training, could be affirmed that the network with Gaussian initialization is in a local minimum because the loss has converged as could be seen in figure 5.6(e). The loss of LeNet-5 without being modified (figure 5.6(a)) is the one whose oscillation at training is less than others.

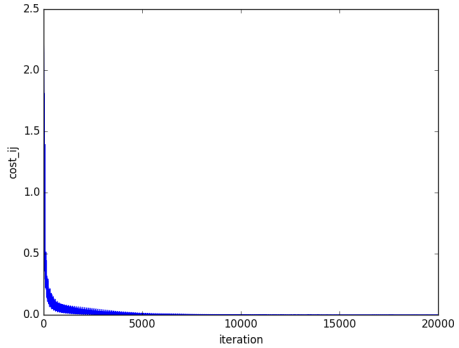
About the error at validation, visualizing the graphs in figure 5.6, it is very similar the curve for original LeNet-5, LeNet-5 with ReLu, LeNet-5 with LRN and LeNet-5 with LRN and ReLu.



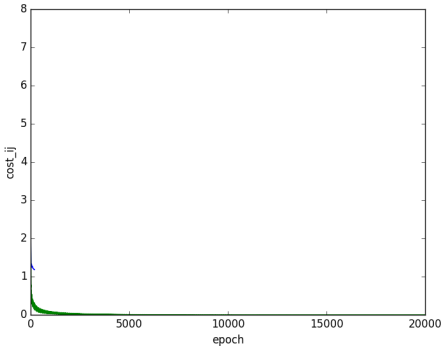
(a) Original LeNet



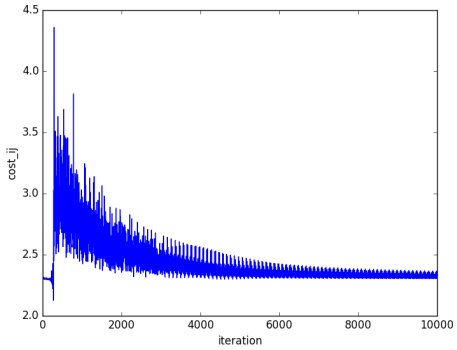
(b) Experiment 1: using LRN



(c) Experiment 2: using ReLu

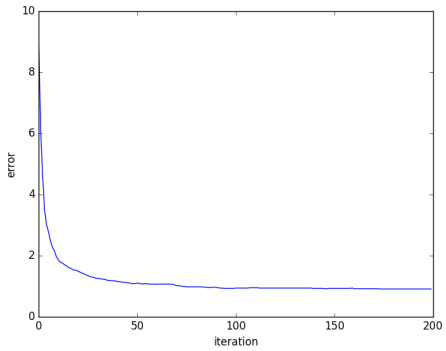


(d) Experiment 3: using ReLu and LRN

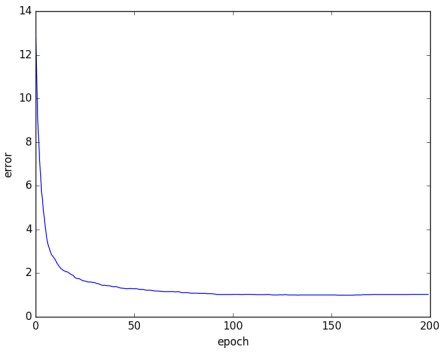


(e) Experiment 4: using Gaussian weights init.

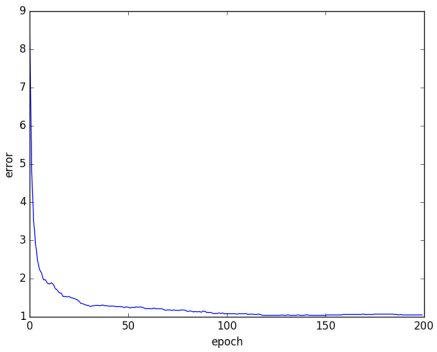
Figure 5.6: Cost function of Lenet and Lenet Modified.



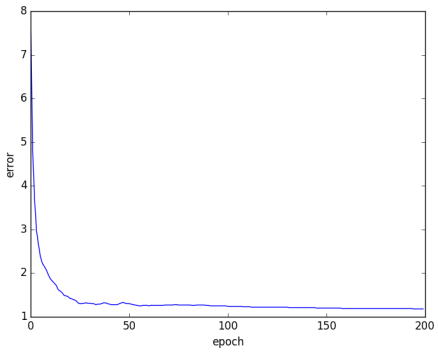
(a) Original LeNet



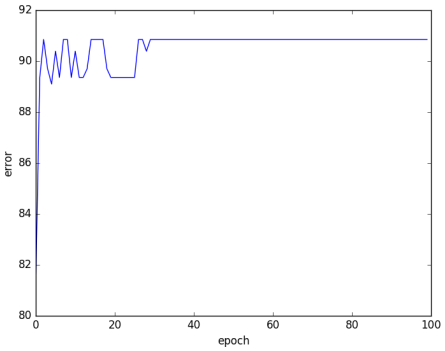
(b) Experiment 1: using LRN



(c) Experiment 2: using ReLu



(d) Experiment 3: using ReLu and LRN



(e) Experiment 4: using Gaussian weights init.

Figure 5.7: Valid error of Lenet and Lenet Modified.

The results obtained, at testing, have been the following ones:

- Original LeNet: Best validation score of 0.91 % obtained at iteration 17400, with test performance 0.92%.
- Experiment 1: using Local Response Normalization: Best validation score of 0.99 % obtained at iteration 13400, with test performance 1.6 %.
- Experiment 2: using ReLu as activation function:Best validation score of 1.04 % obtained at iteration 11900, with test performance 2.4%.
- Experiment 3: using ReLu and LRN: Best validation score of 1.18 % obtained at iteration 19500, with test performance 1.08 %.
- Experiment 4: Gaussian weight initialization: Best validation score of 81.22% obtained at iteration 100, with test performance 80.90%.

The best configuration for the network is the original one. With Gaussian initialization, the network does not find a local minimum in such a sort of time. Using LRN and ReLu, test result is closer to the obtained with LeNet-5 original, but not as good as the last one. Changing the activation function has not been a good change. Not taking into account 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 values are close of the modifications, but the modification of Gaussian weight initialization.

5.1.4 LeNet-5 and RGB FRAV faces database

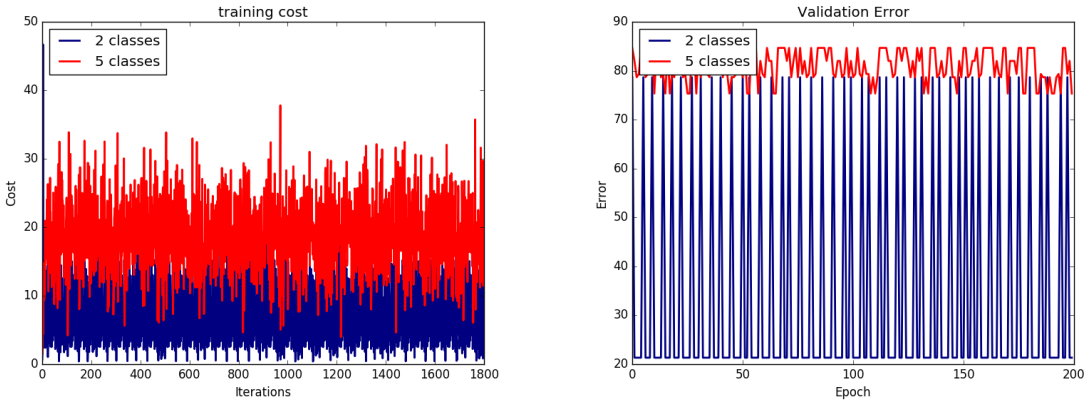
The goal of this thesis is train a convolutional neural network for face anti-spoofing, so the following step is feed LeNet-5 with one face databases, RGB FRAV database, the one that would be used in the final architecture.

One of the databases used with the final architecture and configuration of the network is FRAV database. LeNet-5 is tested with this database in this current section.

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

The network has been tested with this databases in the two ways of 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 much samples as in the MNIST database.



(a) Cost (b) Error

Figure 5.8: Cost at training (a) and Error at validation (b) when LeNet has been used with RGB FRAV database.

In figure 5.8 is represented the training cost 5.8(a) and the validation error 5.8(b) of the training process. In the figure, is represented when 2 classes are used to classify in blue 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 should 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 [**?**]. LeNet-5 architecture is simple to use it for this project purpose.

In [**?**] authors develop a convolutional neural network with similar objectives as the described in this thesis. The CNN is based on *Imagenet* architecture [**?**], this is a determined architecture used for classifying objects which is well known and very used in the deep learning community. Its architecture is described in figure 5.9.

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

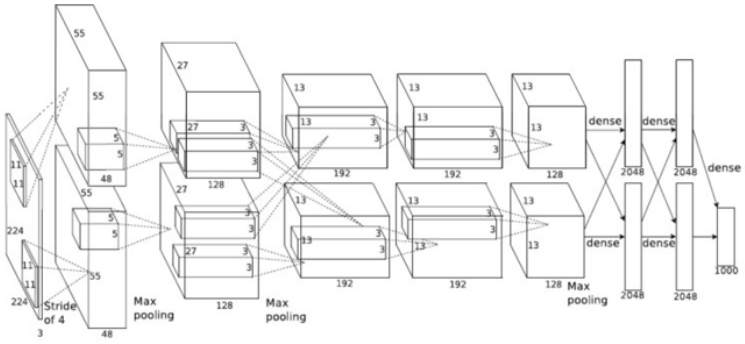


Figure 5.9: Imagenet architecture.

Authors from [**?**] describe lightly the architecture, the parameters and the changes made from Imagenet, so it is not possible to develop the same architecture.

5.2.1 New Architecture

The new architecture would be formed by 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 activation function. Logistic regression has been used in the training process. Weights have been initialized with Gaussian initialization and bias to 1.

Some experiments are going to be developed, and parameters like the learning rate would be changed to obtain the optimal one and those would be explained for each one.

5.2.2 Databases

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

Samples distribution are described in table 5.1 where the number of samples per class and subset (train, test and validation) are summarized for each database. It is possible to realize that 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 5.1: Samples distribution for each database

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

5.2.3 Experiments Description

With that databases. Some experiments has been carried out. To test, the used classifier is SVM with RBF, the parameter C has been searched for each experiment.:

- General experiment, with Gaussian weight initialization, classification with SVM RBF and SOFTMAX.
- General experiment but with a small database in order to make over-fitting in the network and check it. It has been used 20 samples for training, testing and validation images in all databases but MFSD that has been used 14 images for each subset.
- The same experiment that above but the test has been realized with the same subset that in training, this is to check that the network has over-fit or should have over-fitted.

– The same as above but decreasing the learning rate from 0.01 to 0.001.

In figure 5.10 are represented the experiments with RGB FRAV database. RGB+NIR FRAV database results are represented in figure 5.11. CASIA image database experiments are represented in figure 5.12, CASIA video database is represented in figure 5.13. In figure 5.14 are represented results of MFSD-MSU database.

From images, could be concluded that in general, decreasing the learning rate from 0.01 to 0.001, for this experiment, does not affect in any database.

In table 5.2 could be seen the positive and negatives rates obtained for each database when the general experiment (the fist one described) has been tested. The result obtained with RGB+NIR FRAV database is the best because just 3 samples have been misclassified from 140 images. In general, classification is not bag. The database whose samples are misclassified in a big amount is CASIA videos but the total number of samples is much bigger too.

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

Looking the graphs where a minidataset has been used (20 images or 14), if the cost (training) is visualized, could be expected that the train is learning the images because the cost decreases to 0 (almost zero) so that means that if it is tested with itself the error should be 0, but that does not happen.

The conclusion is the needed of a balanced database, at least to do this experiment. In which the number of class 0 samples are the same that the number of class 1 samples, because the network would not learn in the same way if in some cases the number of samples of attack class is four times than the number of samples of class 1, just predicting 0 would have 25% accuracy, and having less than 5 samples in validation or testing is not a good generalizer (2 samples in class 0 MFSD database).

5.3 Final architecture

The architecture utilized in the final, and the most important experiment is described in this section. This architecture is like the presented in section 5.2.1 and it is going to be explained in detail.

5.3.1 Architecture

The neural network is composed by five convolutional layers: the first and second convolutional layers (CL1 and CL2) , whose kernel sizes are 11x11 and 3x3 respectively, are followed by a local response normalization layer and a max pool layer whose size is 2x2. The third convolutional layer (CL3), with a kernel size of 3x3 , the next layer is a convolutional one (CL4) of size 3x3 followed by a max-pool layer whose size is 2x2. The next two layers are Dropouts layers (DL1 and DL2) with 4096 neurons. The next layer us a Fully-connected layer (FL) with 4096 neurons at the input and 2000 layers at the output.

The activation function used in each layer is the ReLu. The weights have been initialized pseudo-randomly (a random initialization that could be repeated selecting the same seed) with a Gaussian distribution and the bias has been initialized with 1.

It has been used the minibatch Stochastic Gradient Descend and in the training, the used classifier, is the logistic regression.

The learning rate is fixed at a 0.01 value and a bath size of 20, except when the MFSD database is used, in this case, the batch size is 14.

5.3.2 Database

The dataset used in this experiment is the same as used in the previous architecture, the described in section 5.2.2, but also, RGB+NIR FRAV database added in classification level is going to be used.

The sample distribution is the same, which is summarized in table table 5.1. There is just one row for RGB+NIR FRAV database because the sample distribution is the same independently of when the are concatenate.

Two classes has been used, class 0: the real users class and class 1: the attacks class.

5.3.3 Classification

For testing some classifiers has been utilized, and they are fed by the output of the convolutional neural network last layer, the Fully-connected layer.

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

The classifiers, before use them, has been personalize to each and particular time (for each database and if LDA or PCA is used). For that, cross validation has been used, more concretely, the *cros_val_Score()* function from sklearn has used with 10 folders.

- For 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 KNN classifier, the number of neighbors (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 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 PCA, the number of components has been found in a range of 3 to 5000, in 3 to 3 steps.
- For 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.

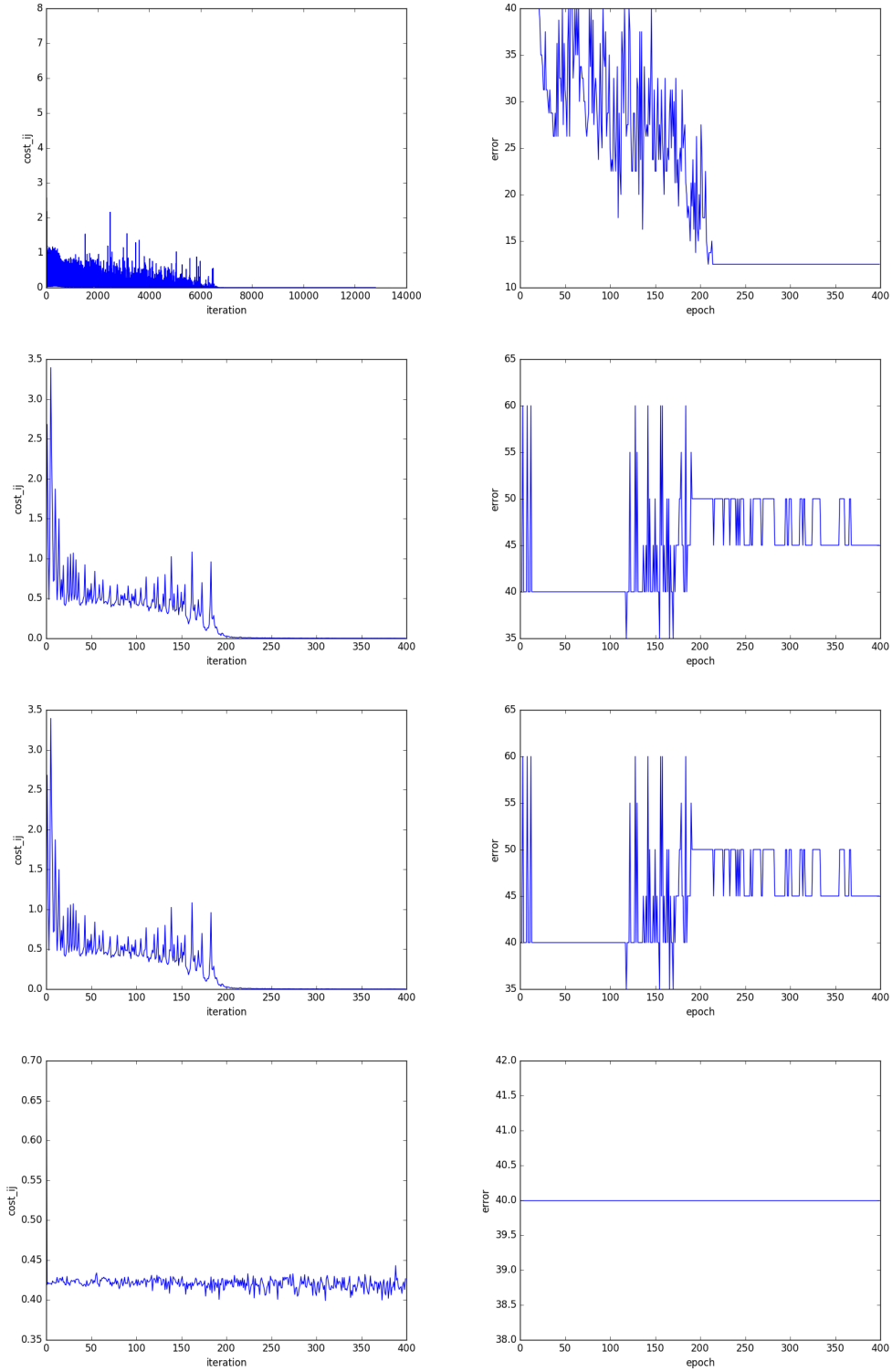


Figure 5.10: cost and error of the tree experiments with frav.

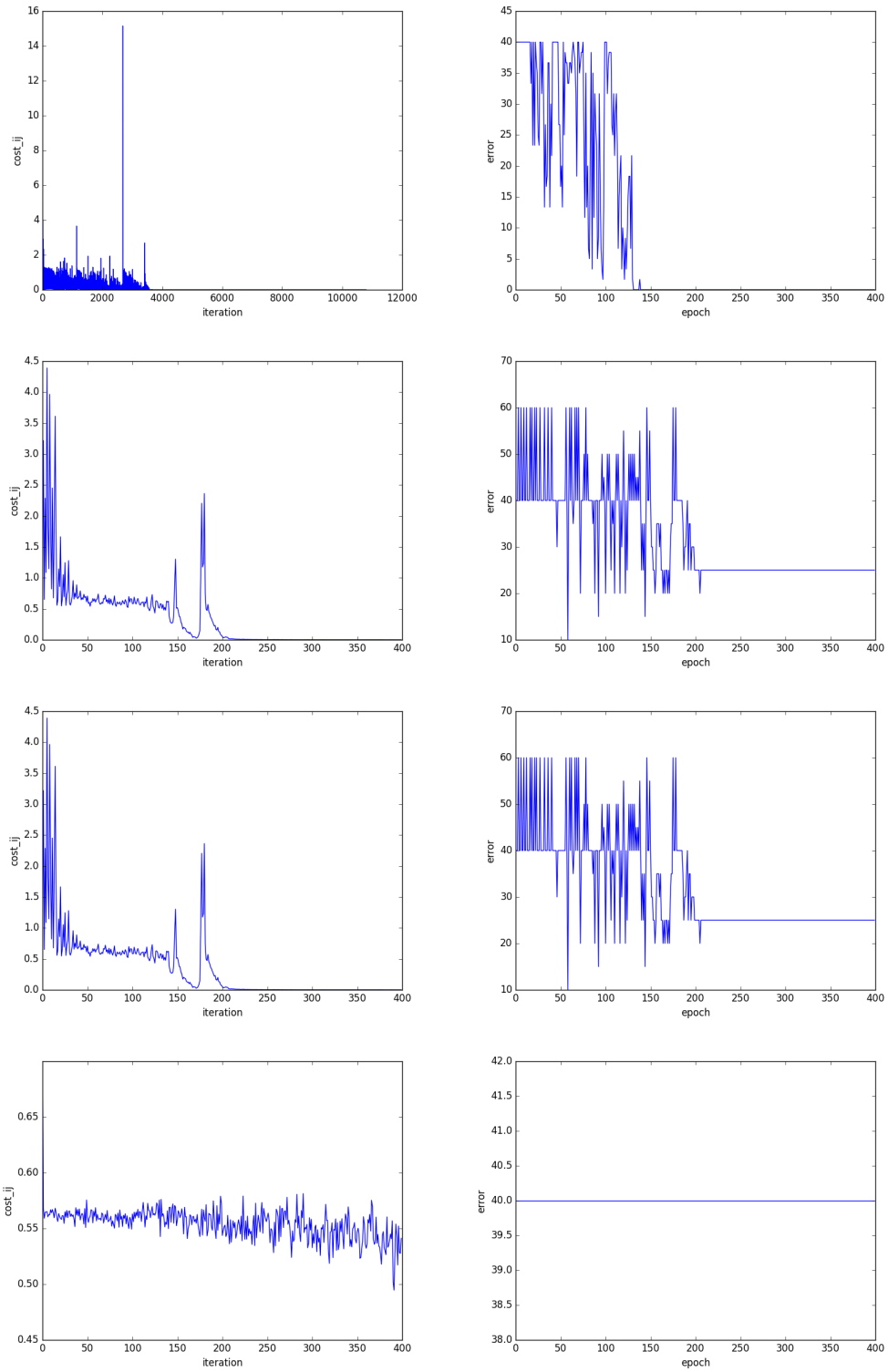


Figure 5.11: cost and error of the tree experiments with FRAV (rgb + nir) image level images.

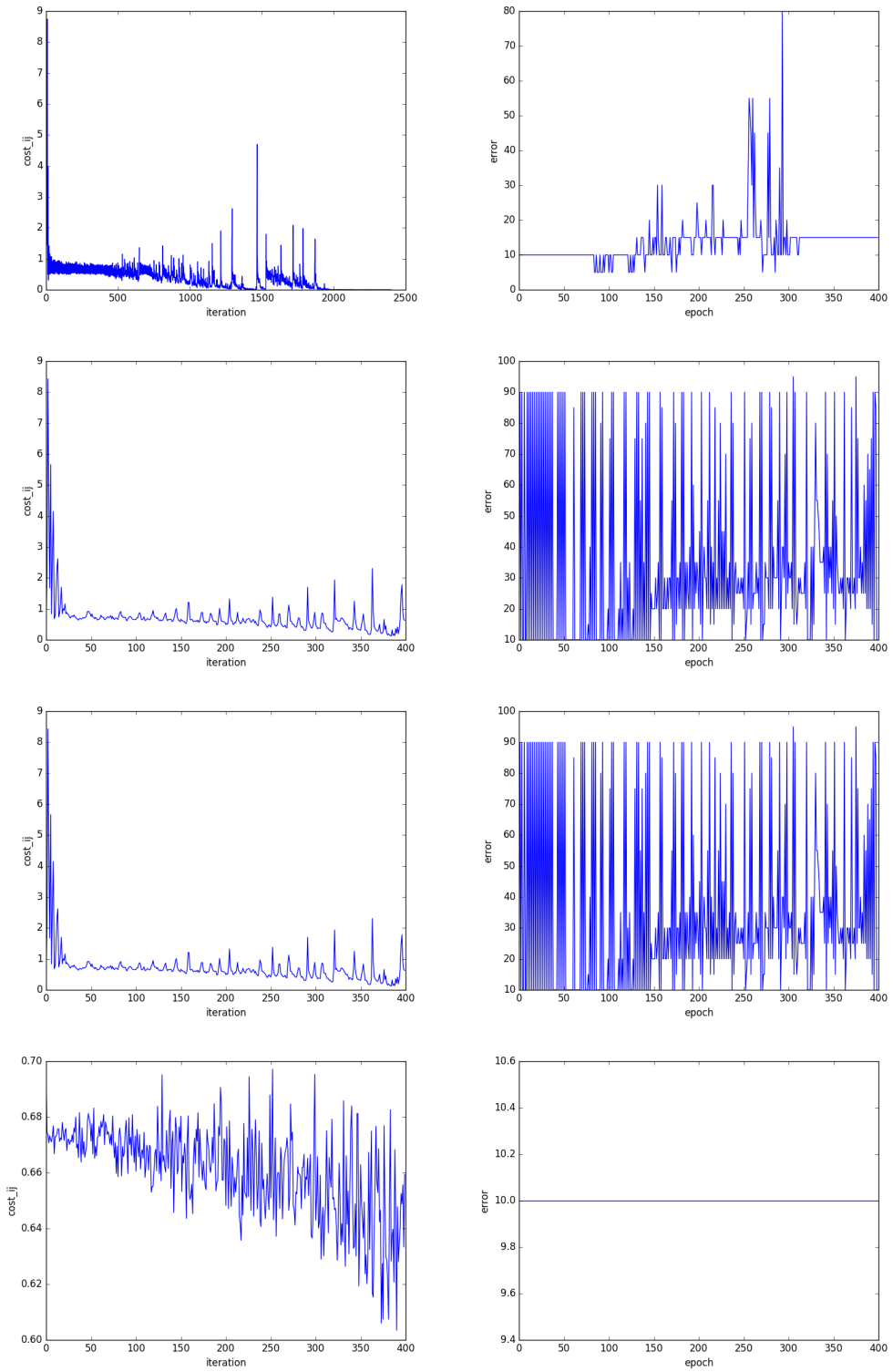


Figure 5.12: cost and error of the tree experiments with CASIA images.

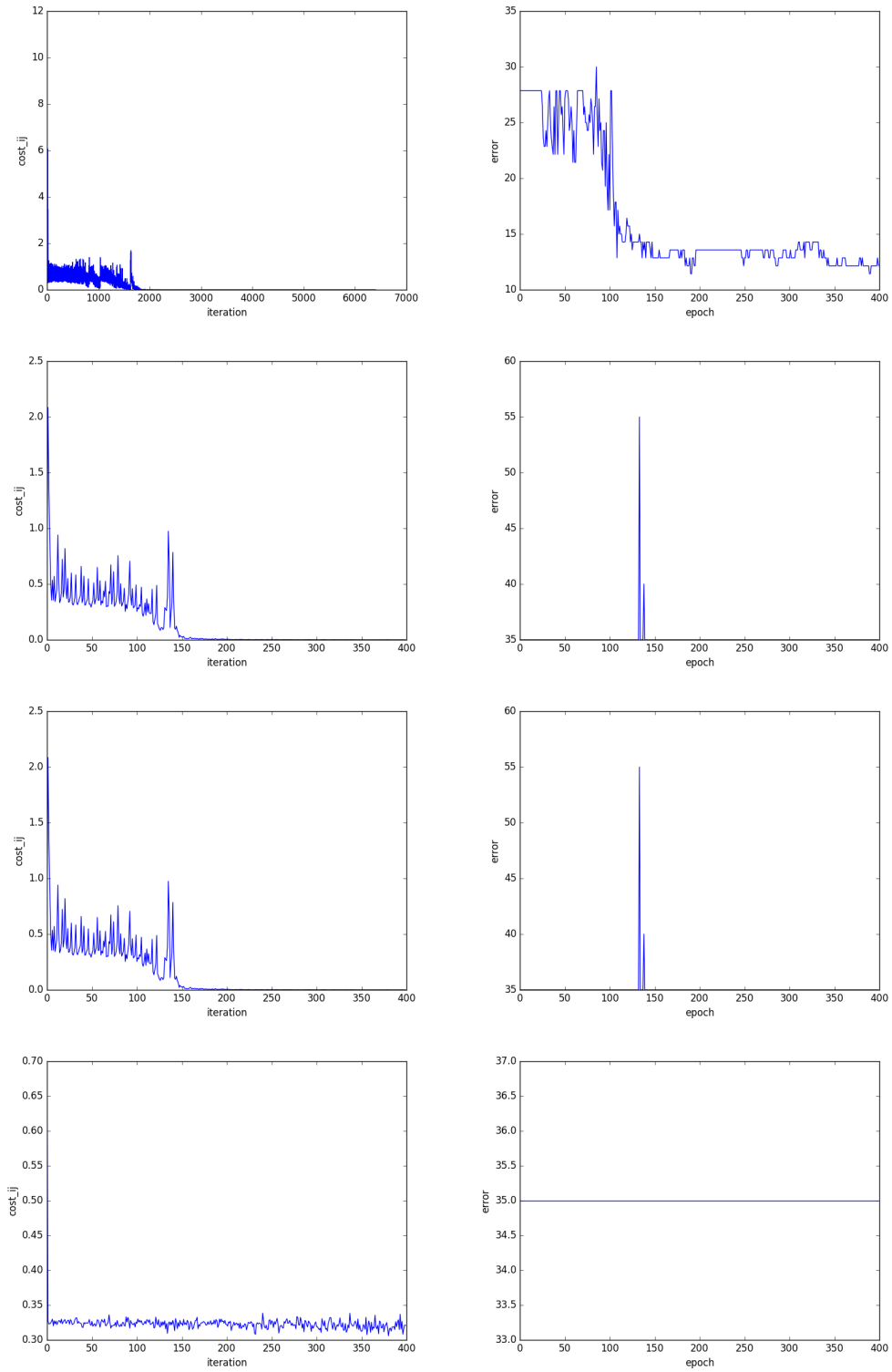


Figure 5.13: cost and error of the tree experiments with CASIA videos.

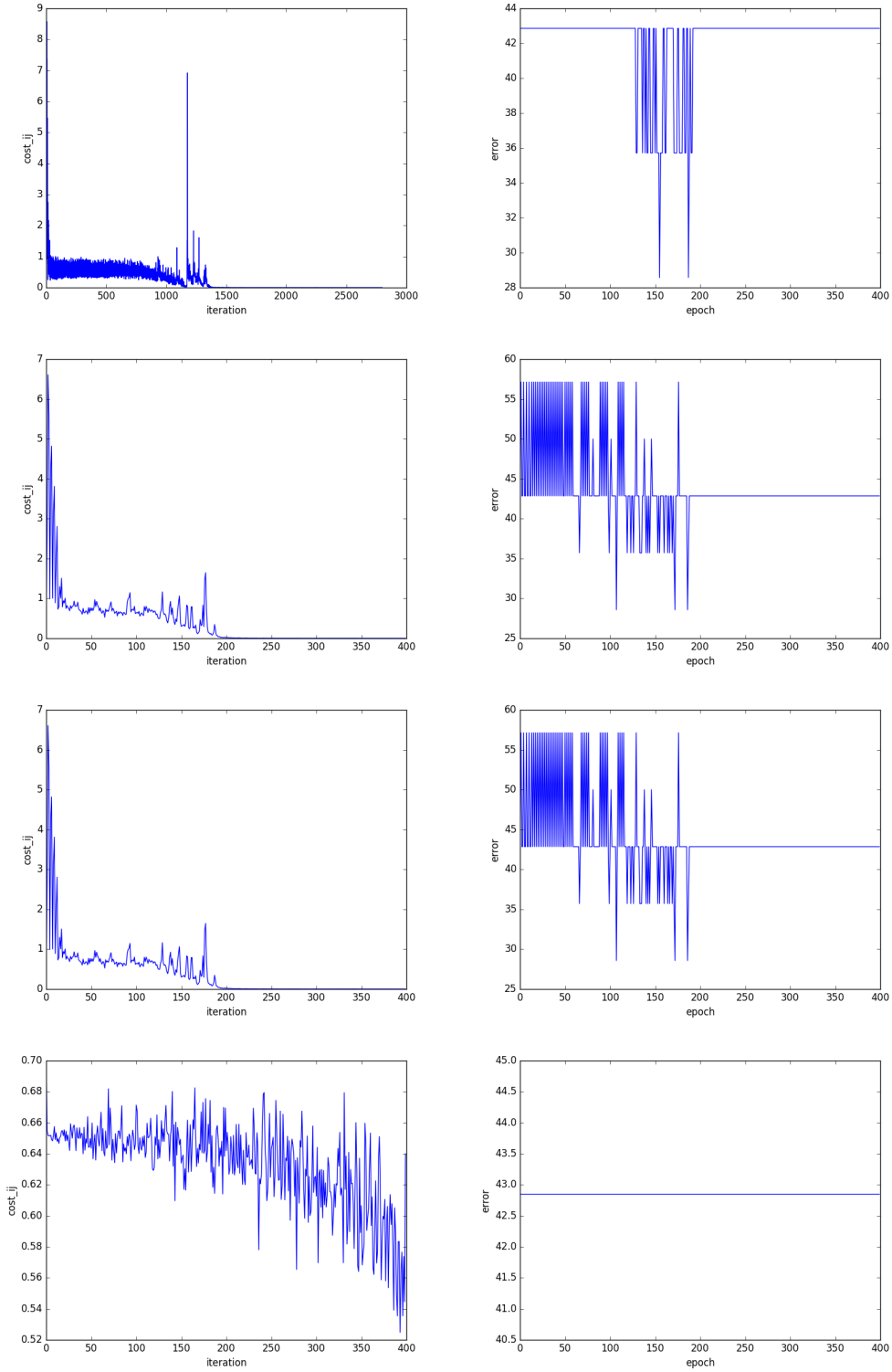


Figure 5.14: cost and error of the tree experiments with MFSD images.

Chapter 6

Results

In this chapter the results obtained are presented as well as the training process that is going to be discussed.

6.1 Training and validating process

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

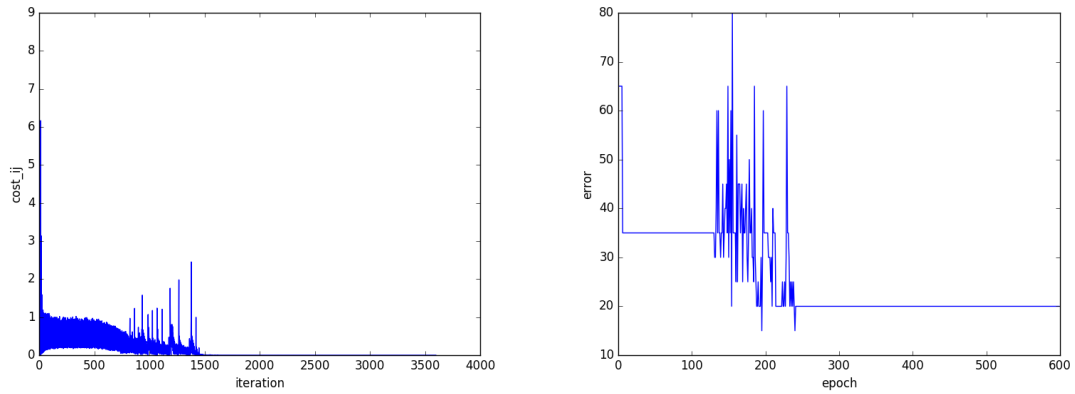
6.1.1 CASIA Image database

In figure 6.1 the cost and training are represented (6.1(a) and 6.1(a) respectively). The training process converge in a low values, smaller than 0.0001; while the validation process converges in 20% error from the 250th epoch. The iterations where the training process oscillates sharply agree with the epochs where the cost oscillates sharply too. After those oscillations, the validation gets constant and the training error variates in its lowest values.

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

6.1.2 CASIA Video database

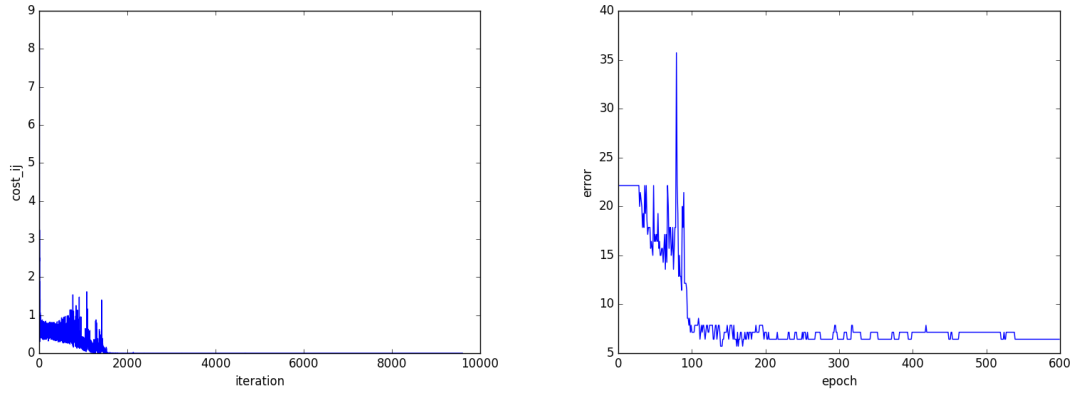
The cost and error obtained at training and validation processed when CASIA video database is used, are represented in figure 6.2. The minimum cost is almost 0 and is obtained before 2000th iteration, as is represented in figure 6.2(a). The validation converges



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

Figure 6.1: Cost (a) and error (b) at training CASIA image database

at the same time as the training, where oscillate between 7% and 6%.



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

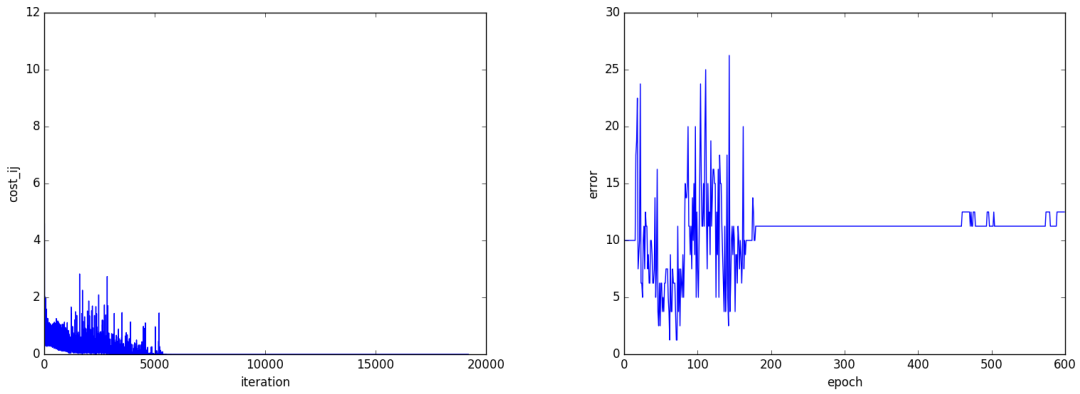
Figure 6.2: Cost (a) and error (b) at training CASIA video database

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

6.1.3 FRAV database

The cost and error obtained at training and validation processes respectively are represented in figure 6.3. The cost, that could be seen in figure 6.3(a) get a low value, when

it converges before the 5000th iteration, the cost obtained is 0.000001. The validation error, represented in figure 6.3(a), fluctuate a lot, the curve should decrease, but it does not and in 200 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.



(a) Cost at training

(b) Error at validation

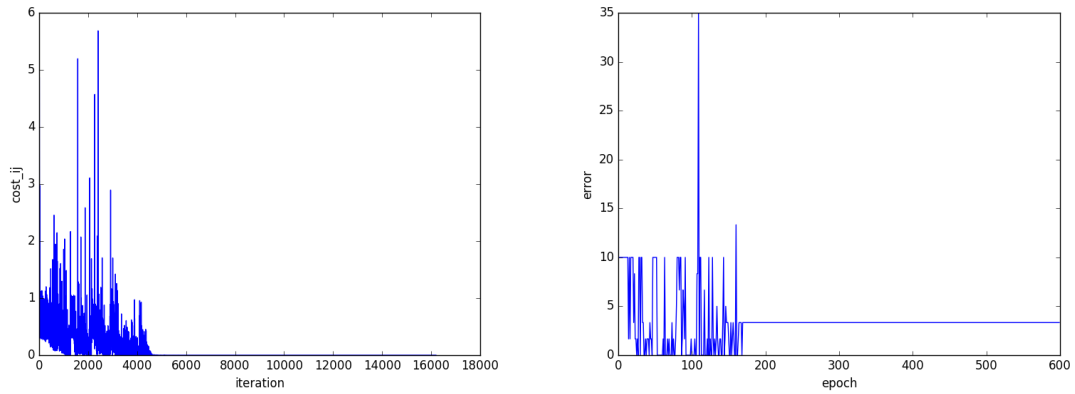
Figure 6.3: Cost (a) and error (b) at training RGB FRAV image database

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

6.1.4 FRAV RGB+NIR (feature level) database

The RGB+NIR FRAV database, whose images has been added before the training process, has a cost and error which is represented in figure 6.4. The cost at training, that could be seen in 6.4(a) oscillates in the 3500 first iterations until decrease its value to 0 in the last 100 iterations. The cost at validation oscillates in the same oscillation cost period, where the value gets 0% in different times. After this, the value gets in 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.

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



(a) Cost at training

(b) Error at validation

Figure 6.4: Cost (a) and error (b) at training RGB and NIR FRAV (feature level) image database

6.1.5 FRAV RGB+NIR (classification level) database

In figure 6.5 is represented the training cost and the validation error when the database has been trained, first the RGB subset and then NIR subset. This two subsets are trained independently and features of each best model are concatenated to test performance.

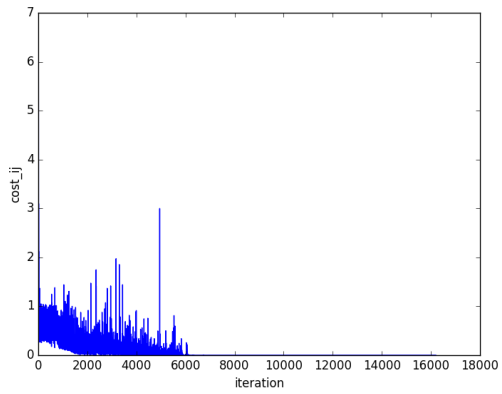
In RGB and NIR cost training, after decreasing its value got a very low value as in others databases. About the cost in validation, in both cases gets 0%, the difference is that in RGB, this lowest value is gotten twice at epoch 124 and 125 and then the value is increased until get constant at 10%, what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

The best RGB model is the 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%.

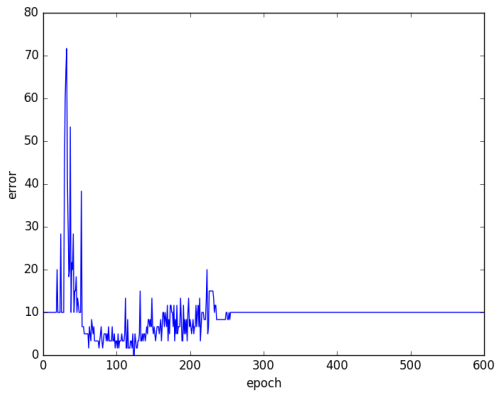
6.1.6 MFSD database

The training results obtained for MFSD database are represented in figure 6.6. The cost decreases slowly until get a desirable cost value, near to 0%. The validation error curve graph (figure 6.6(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 is increased until 35%.

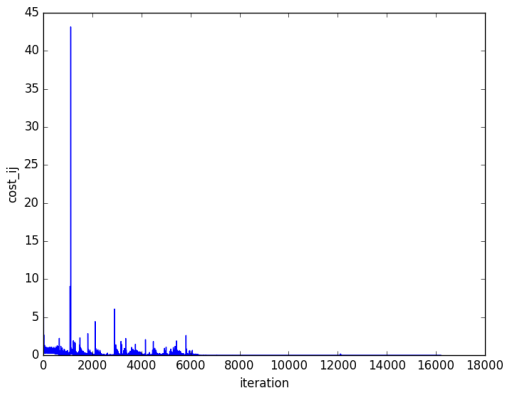
The best model is the obtained at iteration 7, in the first epoch, where the validation error is 14,28%.



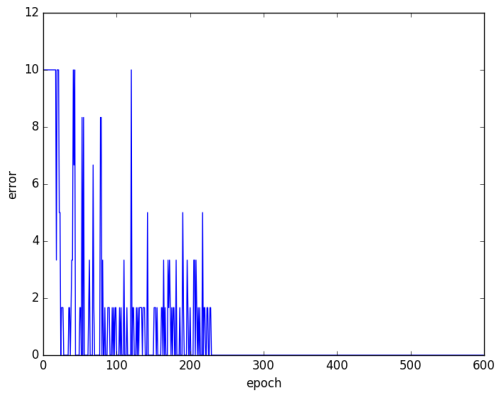
(a) Cost at training RGB subset



(b) Error at validation RGB subset



(c) Cost at training NIR subset



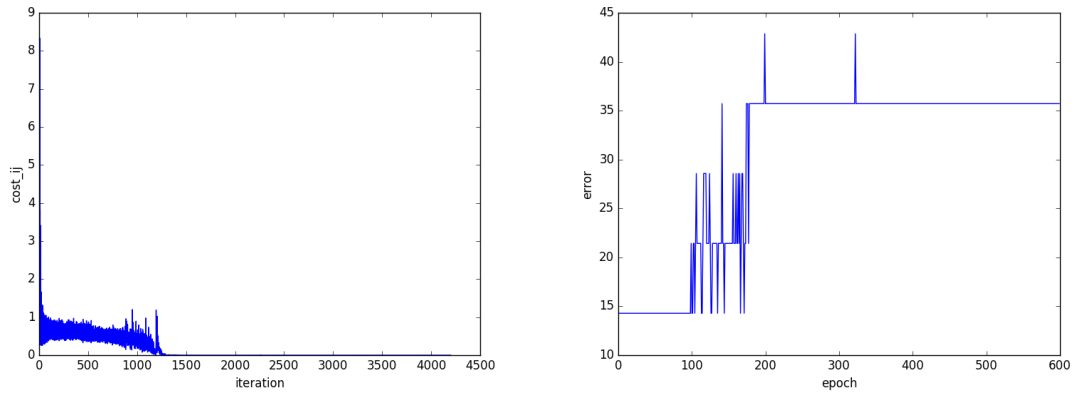
(d) Error at validation NIR subset

Figure 6.5: 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

6.2 Testing process

From the best CNN model, for each database, A SVM (with lineal and RBF kernel), KNN, Decision Tree and logistic regression has been trained. Also the testing has been realized 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 realized after chosen the best classifier model. The results are going to be exposed independently of each database and then



(a) Cost at training

(b) Error at validation

Figure 6.6: Cost (a) and error (b) at training MFSD image database

compared among all databases.

Best obtained results are highlighted in bold in each table.

6.2.1 CASIA Image database

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

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 6.1: APCR and BPCR classifying results using CASIA image database

In table 6.1 are exposed the APCR and BPCR parameters for each classifier. All the attacks samples are rightly classified because APCR value is 0. The number of genuine samples misclassified is what changes, it gets 1 for SVM with kernel lineal, all the positives samples have been misclassified, and decision tree whose number of misclassified real user are 4 (the half of the total real samples of the test subset). The rest of the classifiers

takes a 0.125 APCR value, just one sample is misclassified, the same sample which could be seen in figure 6.7.

With respect to APCR and BPCR it is not possible to get the best combination with which the results are better, because most of classifiers works correctly.

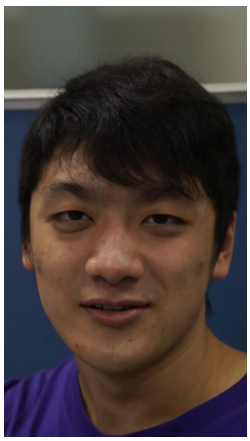


Figure 6.7: Casia image misclassified sample.

6.2.2 CASIA Video database

Next analyzed results are from CASIA video database.

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 6.2: APCR and BPCR classifying results using CASIA video database

In table 6.2 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, no matters if PCA

or LDA have been used, because all the positives samples have been classified as negatives.

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 improve or worsened.

Also The last conclusion that could be gotten from the table is that Attacks samples are better classified that genuine samples, it is due to the fact that has been trained with more negative samples.

6.2.3 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 6.3: APCR and BPCR classifying results using FRAV RGB database

APCR and BPCR results are in table 6.3. 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, that results are as good as the 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 has been used, although the improved is not too much.

The misclassified samples are repeated among the classifiers. The two samples shown in figure 6.8 represents the two most misclassified images, almost every classifier have classified it incorrectly.

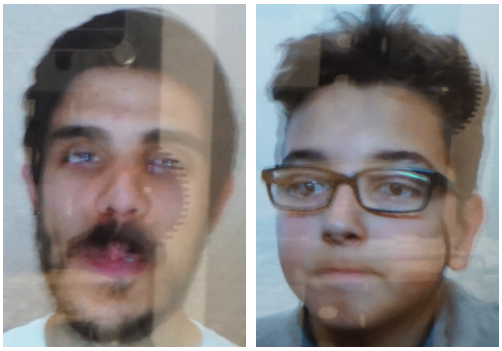


Figure 6.8: RGB FRAV misclassified samples.

6.2.4 FRAV RGB+NIR (feature level) database

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

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

Table 6.4 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 when just the classifier has been used or if PCA has been applied too, classifications have produced good results, the APCR and BPCR values are very low, close to 0. In fact, Decision Tree and logistic regression classify correctly all the samples because APCR and BPCR are equal to 0.

PCA and LDA have not improved results significantly. 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 6.9.

Figure 6.9: RGB+NIR FRAV (feature level) misclassified samples.

6.2.5 FRAV RGB+NIR (classification level) database

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

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

APCER and BPCER results are summarized in table 6.5. Results are closer to 0, but none classifier classifies perfectly. There is possible to see that with LDA that APCR results are better and BPCR results are perfect.

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

In most of cases, classifiers just mis-classify one sample, the same sample represented in figure 6.10. Classifiers that mis-classify more than one sample, the sample 6.10 is one of the misclassified.

Figure 6.10: RGB+NIR FRAV (classification level) misclassified samples.

6.2.6 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 = 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 6.6: APCR and BPCR classifying results using MFSD-MSU database

The classification with this database is not good because all the samples are classified as negatives, as could be seen in table 6.6 where , independently of the classifier or if LDA and PCA are used, the BPCR value is always 1 and the APCR value is 0.

6.3 Comparative among databases

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

6.3.1 FRAVs databases

Three FRAVs databases are compared in order to discuss easier if NIR images, in general, help in order to detect anti-spoofing attacks. In order to do that, an due to the fact to the big amount of data, one classifier is chosen to do this comparative: SVM with RBF kernel.

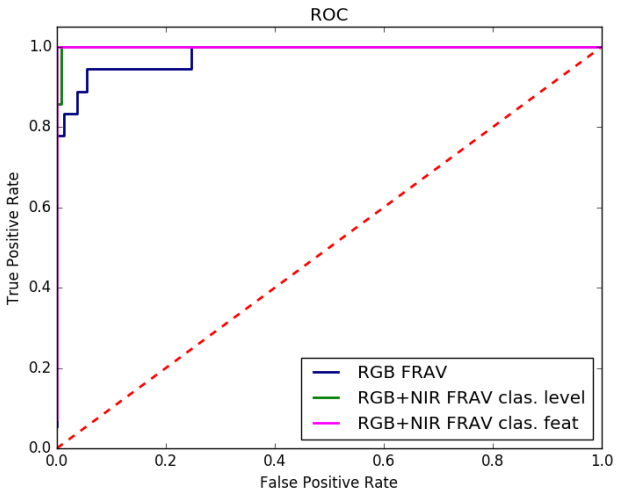


Figure 6.11: Comparative among FRAV databases with SVM RBF.

In figure 6.11 are represented the three databases: In blue is represented RGB database, in green RGB+NIR FRAV database classification level and in magenta RGB+NIR FRAV database feature level. And the ROC curve with NIR work better, in fact, is almost perfect.

This image and the perfect APCR and BPCR result obtained with logistic regression and decision tree in the RGB+NIR feature level tends to conduce that NIR database are improved the results.

6.3.2 CASIAs databases

As the same as FRAVs databases, the comparative is going to be made with CASIAs databases: Image and Videos. In order to do that, KNN classifier is going to be used.

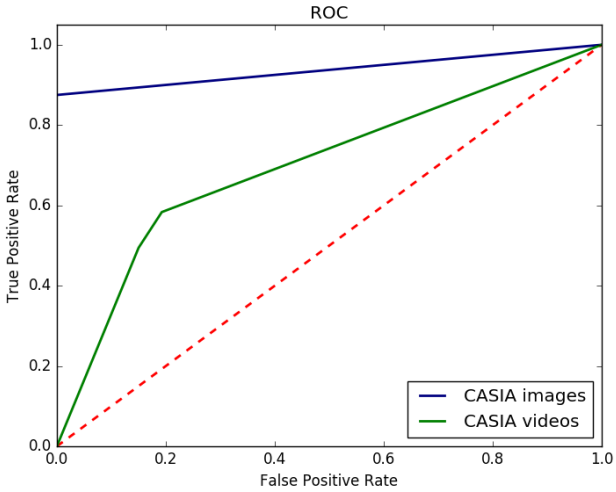


Figure 6.12: Comparative among CASIA databases with KNN.

In figure 6.12 is represented the comparative between image CASIA in blue and video CASIA database in green. As it is possible to see, the result is much better when just CASIA images databases are used, although CASIA video results are not bad.

6.4 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 of the optimal parameter for each classifier, and the metrics calculation. There are six general experiment, one per used databased:

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.

In table 6.7 the time (minutes) is exposed for each process. All the processes differs

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

Table 6.7: Executing time

(generally) just in the database, so the differences in time is because one database is bigger than other or because it is need to train two different neural networks as in RGB+NIR FRAV classification level.

Chapter 7

Conclusions and future work

In this section the conclusions obtained along the methodology are presented as well as the future work.

7.1 Conclusions

The importance of having a representative and well-build database is essential to train any classifier or neural network, because it is the main tool which classifiers or neural networks have to use. In the used databases, it has been proved that the quality of images is important to extract the features better and minimize the error. The results used with the FRAV database were better than CASIA database (for example) and one main reason is the quality of images. FRAV database has been build with a professional camera.

Continuing with databases, the need of having a balanced database has been raised. If the database is composed by 70% class 1 samples and 30% class 2 samples, just if the classifier classify all the samples as class 1, the error will be the 30%, so the classify would learn better to classify one class samples. This problem has appeared in experiments, mainly with MFSD database. There are different ways to try to minimize unbalanced databases, the first one would be get more samples from the class which has less samples. There not work if more samples of the less-samples class are obtained from the existing one, because if samples are repeated the classifier would learn them and over-fitting would be obtained.

So, FRAV database is the database whose samples are the most correctly classified. The worst results obtained has been with MFSD-MSU database, this database has been built with the camera of a smartphone and a laptop. To get better results with MFSD-MSU database, the CNN needs to be modified, for example, trying to change the learning rate or trying to build a balanced database, because all the training and testing processes get stuck classifying all the samples as negatives.

It has been conclude that using NIR images with RGB have improved the results. NIR images could detect attacks because printed photos or tablets are similar to real user in RGB space but not in NIR space.

With respect with CASIA image and CASIA video, results are much better with CASIA images, one sample is misclassified in most of the cases, but in CASIA videos the quantity is much bigger. But there are much samples in CASIA video database.

The best model combination of CNN + classifier is obtained when RGB+NIR FRAV database feature level. This best result is a perfect classification with logistic regression and with Decision Tree too. RGB+NIR FRAV database classification level results are also outstanding, especially when LDA has been used for each classify because just one sample has been misclassified, the same sample and two samples has been misclassified with LDA + logistic regression.

SVM with RBF kernel is, in general, the classifier that work best along the databases, although with this classifier has not gotten the best result. SVM is a classifier that works good with bi-class problems. However, it has not been possible to classify correctly, most of the time, when SVM with linear kernel has been used.

With respect to using LDA or PCA reduction dimensionality methods none 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, because results could be improved.

7.2 Future work

First trying to minimize the unbalanced database, like it is not possible to get more samples, change the error in that way that the network does not be penalized as much as will be during training if a sample of the less-samples class is misclassified.

It would be interesting test the neural network and the classifier which works the best in a database with other databases.

If further work would like to do with the databases, it should be necessary to modify the convolutional neural network if working with MFSD-MSU database is necessary.

About improve the convolutional neural network, an adaptive learning rate would help. This kind of learning rate change its value, decreasing it, with the epochs. So, big steps

are made while a minimum is searched and small steps are made to get in the minimum.