People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University of Larbi Tbessi –Tébessa.
Faculty of Science Technology
Department of Electrical Engineering

FINAL STUDY DISSERTATION
In the aim obtaining of MASTER Degree - ACADEMIC

Domain: Sciences and Technology
Option: Telecommunications
Specialty: Networks and Telecommunications

Topic:

Multispectral images classification using convolutional neural networks (Deep learning)

Presented by:

YACINE BELHOUCINE

The Examining Committee Composed From:

M. RIAD SAIDI	MCB	President
Dr. AMEL BOUCHEMHA	MCA	Supervisor
Dr. ABDALLAH MERAOUMIA	MCA	Examiner

Date of presentation Academic year: 2019/2020

﴿ يَرْفَعِ اللَّهُ الَّذِينَ آمَنُوا مِنكُمْ
وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَاتٍ وَاللَّهُ
بِمَا تَعْمَلُونَ خَبِيرٌ ﴾

صدق الله العظيم

الآية 11 من سورة المجادلة.

ملخص

ظهر التعلم العميق والشبكات العصبية التلافيفية بشكل خاص(Cnn)لحل المشكلات التي تواجه التعلم التعلم الآلي ، وكان الهدف من هذه المخطوطة دراسة تطبيق التعلم العميق في تصنيف الصور باستخدام الشبكات العصبية التلافيفية. علاوة على ذلك ، لتقييم أداء التصنيف ، استخدمنا مجموعات مختلفة من الطبقات والعهود المخفية. علاوة على ذلك ، يتم تدريب الشبكة باستخدام عدة خوارزميات مُحسِنة. تم استخدام لغةPythonمع إطار TensorFlow .

Abstract

Deep Learning and more particularly convolutional neural networks (CNN) appeared specially to solve the problems encountered with machine learning. The objective of this manuscript was to study the application of deep learning in image classification using convolutional neural networks. Moreover, for evaluating the performance of the classification, we have used various combinations of hidden layer and epochs. Further, the network is trained using several optimizer algorithms. The Python programming language with the TensorFlow framework were used.

Résumé

Le Deep Learning et plus particulièrement les réseaux de neurone convolutif (CNN) sont apparus notamment pour résoudre les problèmes rencontrés avec la machine Learning. L'objectif de ce manuscrit était d'étudier l'application du deep Learning à la classification d'images à l'aide de réseaux de neurones convolutif. De plus, pour évaluer les performances de la classification, nous avons utilisé diverses combinaisons de couches cachées et d'époques. En outre, le réseau est formé à l'aide de plusieurs algorithmes d'optimisation. Le langage de programmation Python avec le Framework TensorFlow ont été utilisés.

ACKNOWLEDGMENTS

I extend my sincere thanks, appreciation and gratitude to my supervisor Dr. Bouchemha Amel, she is one of the dearest professors who was and still teaches me and for being my supervisor in this work. She was a help and guide for me to complete this humble work.

I even want to thank the professors and workers of the Department of Electrical Engineering at the University of Larbi Tbessi –Tébessa.

And to everyone who helped us in this work, each in his name, to you all the most sincere expressions of thanks.

DEDICATION

To whom God has entrusted with prestige and dignity ... To who taught me to give without waiting ... To whom I carry his name with pride ... I hope that God extends your life to see fruits that have come to be harvested after a long wait, and your words will remain stars to be guided today, tomorrow and forever.

My dear father.

To my angel in life... To the meaning of love and to the meaning of tenderness and dedication... To the smile of life and the mystery of existence to those whose supplications were the secret of my success and affection… a surgical balm to the most precious of loved ones.

My dear mother.

To those who accompanied me since we carried small bags and walked the path step by step and still accompanied me until now ... to a burning candle illuminating the darkness of my life ..

My sister: Youssra.

To my brother and my companion in this life, with you I will be me and without you I will be like

Anything, to whom I see optimism and happiness in his laughter ... At the end of my career, I want to thank you for your noble stances to those who looked forward to my success with looks of hope.

Dear brother: Chahin.

To the brothers and sisters, to those who enjoyed generosity and were distinguished by loyalty and giving, to the springs of pure honesty, to those with whom I was happy, and with them on the sweet and sad paths of life I walked to those who were with me on the path to success and goodness.

To whom I knew how to find them and taught me not to waste them

Derby girlfriends in my workplace.

And to everyone who we contribute my pen and remember them my heart ... I dedicate this work to them.

List of Contents

List of figures

List of Tables

Acronyms

ACC	Accuracy
ADAM	Adaptativ Moment Estimation
AI	Artificial Intelligence
API	Application Programming Interface
AUC	Area under the ROC Curve
CASIA	Chinese Academy of Sciences' Institute of Automation.
CNN	Convolutif Neural Network
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DL	Deep Learning
DWT	Discrete Wavelet Transform
EER	Equal Error Rate
FA	False Acceptance
FAR	False Accept Rate
FC	Fully Connected
FN	False Negative
FP	False Positive
FR	False Rejection
FRR	False Rejection Rate
FT	Fourier Transform
GLCM	Gray Level Co-occurrence Matrix
HOG	Histogram of Oriented Gradients
K-NN	k-Nearest Neighbor
LBP	Local Binary Pattern.
ML	Machine Learning
MSE	Mean Square Error
NC	Number of Clients
NIR	Nir infrared
ONEIRIS	Open-ended Neuro-Electronic Intelligent Robot Operating System
RELU	Rectified Linear
RGB	Red, Green, Blue
RNN	Recursive Neural Network
ROC	*Receiver Operator Characteristic*
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TANH	Hyperbolic Tangent
TN	True Negative
TP	True Positive
UGC/CRC	University Grants Committee /China Research Center

General Introduction

I. General introduction

Artificial intelligence (AI) invented by a British scientist called Alan Turing in 1950. The main goal of AI is to imitate human behavior in similar cases. Machine learning (ML) is a discipline of artificial intelligence that offers computers the possibility of learning from a set of observations called dataset. The problem of ML is when it makes an error or misbehave the human responsible must take action to correct such mistakes. That problem was solved recently in 2010 by creating deep learning which is based on neural network.

Deep learning in general and neural networks in particular, have had great success in the image processing and classification. Once of the most important point of the robustness of these methods, is the availability of large amounts of training data. The Convolutional Neural Networks (CNN) is one of the most representative network structures in deep learning. It combines feature extraction and classification. The human brain does not interpret an image by pixel but it decomposes a problem into sub-problems through multiple levels of interpretations. The human brains processes visual signals through a structure of multiple layers, well represented by neural networks. In the late 1980s, Yan Le Cun developed a special type of network called Convolutional neural Networks (CNN). These networks are a special form of multi-layered neural network whose connection architecture is inspired by that of the visual cortex of mammals. For example, each element is only connected to a small number of neighboring elements in the previous layer.

Classification is an intelligent aspect related to humans. The power to group objects based on implicit or explicit criteria is a complex task. The automation of this aspect, we lead to the automatic classification which is a discipline which brings together several fields of study. Image processing was born from the idea and the need to replace the human observer with the machine, the image or signals from sensors were then digitized for processing by the computer.

In our work, we are interested in using the deep learning for the image's classification. For this, we will use three image databases: color images and multispectral images. The datasets are: cats and dogs' images; weather images; PolyU multispectral palmprint and CASIA multispectral palmprint. In our project we will use convolutional neural networks to classify the images. For this, we will create different models with different architectures and different optimization

Algorithms. Afterwards the proposed image classification framework is implemented in Python, by building a CNN to train a system to classify image.

I. Manuscript structure

For more explanation and details our work report is structured in three chapters described as follow:

- Chapter one: a definition of multispectral images and their types as well as the information that can be extracted using local and global feature extracting methods.
- Chapter two: totally reserved for AI, ML and Deep learning conception with a detailed explanation of the CNN and the parameters used in our models and how to evaluate them. In this chapter, we provide in addition a brief introduction into the important concepts of image classification and Classification performance evaluation.
- Chapter three: describes the experimental part of our work and discuss the different results obtained and at the end we end with a general conclusion.

CHAPTER I

Multispectral images and image texture analysis

I.1. Introduction

In recent years, multispectral sensors have been applied in an increasing number of application fields, such as remote sensing, astronomy, physics, medicine and biometrics. Multispectral imaging produces a series of images that contains a specific range of frequencies across the electromagnetic spectrum. Generally, the information obtained forms cubes of multispectral images which have three dimensions, two spatial dimensions representing the position of the surface and a spectral dimension which represents the spectral distribution. The latter thus makes it possible to obtain very in-depth information on the characteristics of the captured scene [1-3]. Due to the increasing popularity of this imaging technique, there is a great demand for reliable solutions which allow a user to get access to the relevant information for a large variety of application areas [1-6].

This chapter mainly gives a general overview of multispectral image processing. It provides the underlying terminology and the necessary fundamentals about multispectral image classification and how extracts the discriminant feature of images.

I.2. Multispectral and hyper spectral images

I.2.1. Core concepts

Human eyes see the energy reflected from objects in three channels: red, green and blue. We cannot see ultraviolet and infrared radiation. In order to perceive the part of the spectrum that is normally inaccessible to us, we need to use multispectral and hyper spectral cameras. The visible (red, green and blue), infrared and ultraviolet areas in the electromagnetic spectrum are conveniently classified according to their frequency and/or wavelength (λ) [1-3]:

- Visible light for humans: 380 nm to 700 nm.
- Infrared: 700 nm to 1 nm.
- Ultraviolet: 10 nm to 380 nm.

The main difference between multispectral and hyper spectral is the number of bands and the width of these bands.

- ***Multispectral imagery***: Generally, refers to groups of 3 to 10 bands which are represented in pixels. Each band is acquired using a remote sensing radiometer see fig I.1 (a)).

- The spectral density is broader and covers a smaller range of the electromagnetic spectrum, e.g. 400 - 1000 nm.

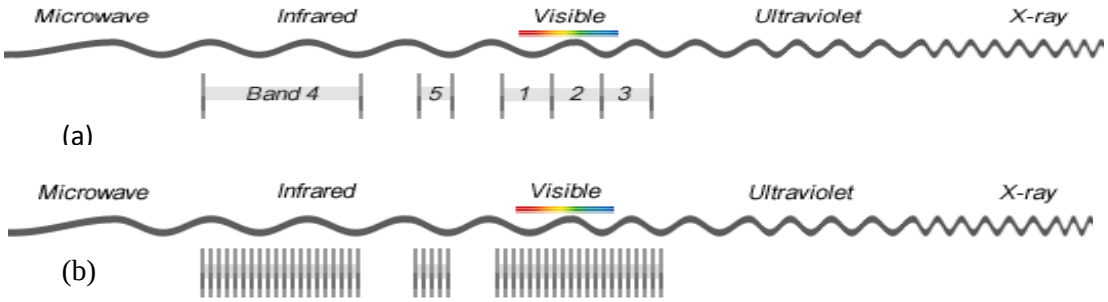


Fig I.1- Spectral bands; (a): Multispectral bands (5 wide bands); (b): Hyperspectral bands (hundreds of narrow bands) [2]

.

- ***Hyper spectral imagery***: Consists of much narrower bands (10-20 nm). A hyperspectral image could have hundreds of thousands of bands. This uses an imaging spectrometer (see fig I.1 (b)). The hundreds of narrow bands that typically cover a large range of the electromagnetic spectrum.

I.2.2. Multispectral and hyper spectral formulation

Multi and hyper spectral imaging devices acquire high dimensional datasets with high-density spectral information. Commonly, these datasets have three dimensions, two spatial dimensions and one spectral dimension, and are often denoted as multi- or hyperspectral image cubes (see. Fig. I.2). Here, each pixel (x, y) in fact has associated a spectrum $\vec{s}(x, y)$ which is usually regarded as n-dimensional vector. Where the number of spectral bands is n (Wavelengthλ). Consequently, each scalar value $\vec{s}(x, y, \lambda)$ in such a cube is the measured reflectance (or absorbance) value of a specific spectral band at(x, y).

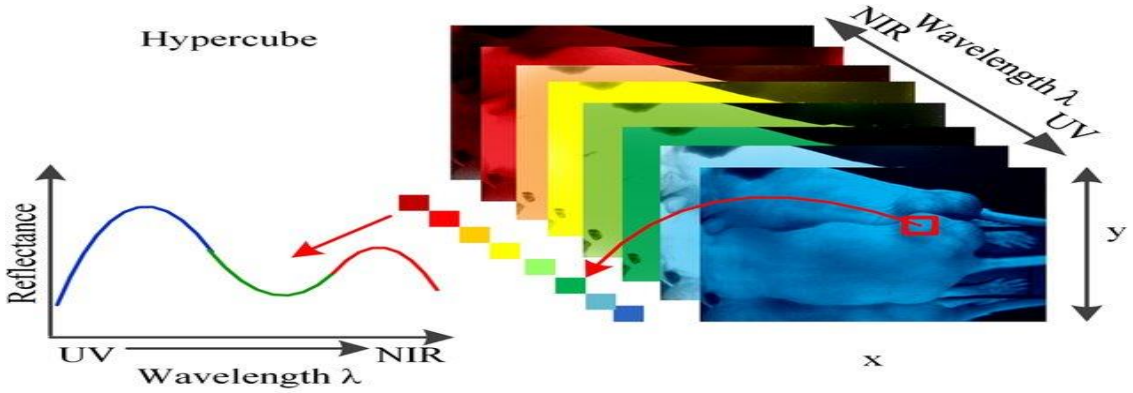


Figure I.2- Multispectral image cube illustration and a spectrum of an image spectrum with n spectral bands [3].

I.2.3. Multispectral or multi bands images [3]

Multispectral data is obtained by simultaneous recordings in a small number of spectral bands (3 to 8), these not necessarily being contiguous. Representation by combination of these digital information bands using the three primary colors (red, green, blue) makes it possible to obtain color images. The data in each band is represented as a primary color and, depending on the relative brightness (i.e., numerical value) of each pixel in each band; the primary colors will combine in different proportions to produce distinct colors.

I.2.4. Hyper spectral images (or imaging spectroscopy) [3]

The hyperspectral images are obtained by sensors capable of recording information in a multitude (often more than 200) of much narrower spectral bands (of the order of a few nm) and often contiguous, in the visible portions, near infrared and mid infrared of the electromagnetic spectrum. Hyperspectral data therefore provide more detailed information on the spectral properties (fine spectral signature) of a scene and allow more precise identification and discrimination of objects than wideband sensors. Each pixel of a hyperspectral image contains the information collected in large acquisition windows spread over the entire visible and infrared spectrum (see figure I.3).

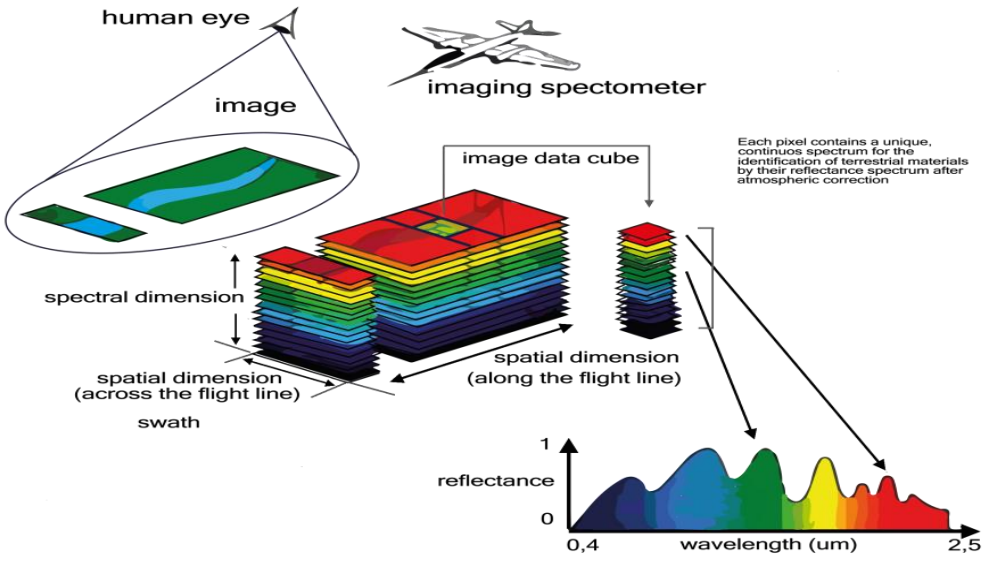


Fig I.3– Imaging Spectroscopy Concept (Source: VITO - Flemish Institute for Technological Research [3].

The amount of information to store and process is therefore enormous and requires much greater computing capacity than in the case of multispectral images. The applications of hyper spectral imaging are of course multiple. Among the most important, we can cite geology (identification of minerals, etc.), precision agriculture, forestry (health status, identification of species, etc.) or management of aquatic environments (water quality, phytoplankton composition, etc.).

I.2.5. RGB images

True color images or RGB images as in (Red, Green, Blue), is the representation of visual stuff, live or not. each pixel of the images is defined by three values, one for the red, one for the green and another one for the blue in an M X N X 3 matrix of type uint8 uint16 uint32... single or double, in uint8 the values are in range [0, 255] or in uint16 values are in range [0, 65535], All the known colors are produces by mixing the three previous colors together by deferent value for each color to see the final true image [1].

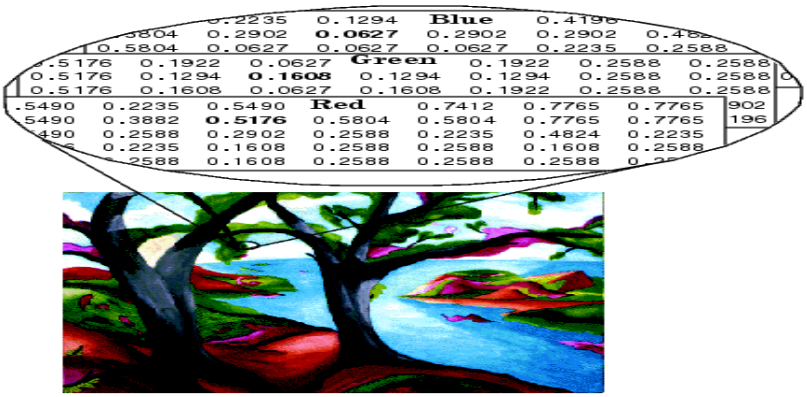


Fig I.4 – Color plan of RGB images [1].

I.2.6. Spectral Bands

Multi and hyperspectral sensors acquire information in several spectral bands. The spectral band is subdivided into several regions based on the wavelength range (e.g. the wavelength of near infrared (NIR) is between 0.7 and 1.0 µm). The wavelength is a physical quantity homogeneous to a length. It is characteristic of a monochromatic wave (i.e. of a single color).

Standard multispectral image interpretation techniques barely exploit the spectral-spatial relationships in the image. The multi-spectral image data is basically treated as a set of

independent spectral measurements at each pixel location, without taking into account their spatial relations [3].

I.2.6.1. Wavelength of visible light

Light is an electromagnetic wave whose color depends on the wavelength. Each shade is characterized by its own wavelength range and it can be difficult to establish precise limits of the different colors on the one hand and of the visible light range on the other (see figure I.5). For good reason, there is only one possible color per wavelength. The values provided vary according to the sources, the most approximate retain for visible light an interval ranging from 400 nm to 800 nm [3].

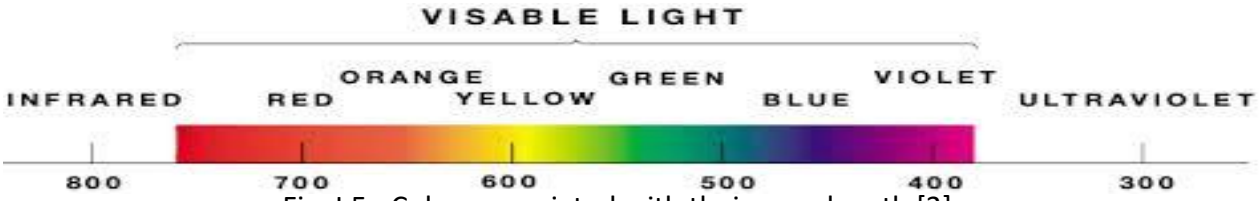


Fig. I.5 –Colors associated with their wavelength [3].

There are many applications for automatic image classification. What all these applications have in common is that they require the establishment of a processing chain from the available images composed of several steps in order to provide a decision as an output. Each step in the establishment of such a classification system requires the search for appropriate methods for optimal overall performance; namely the pre-processing phase, characteristic extraction phase and the learning phase (see figure below).

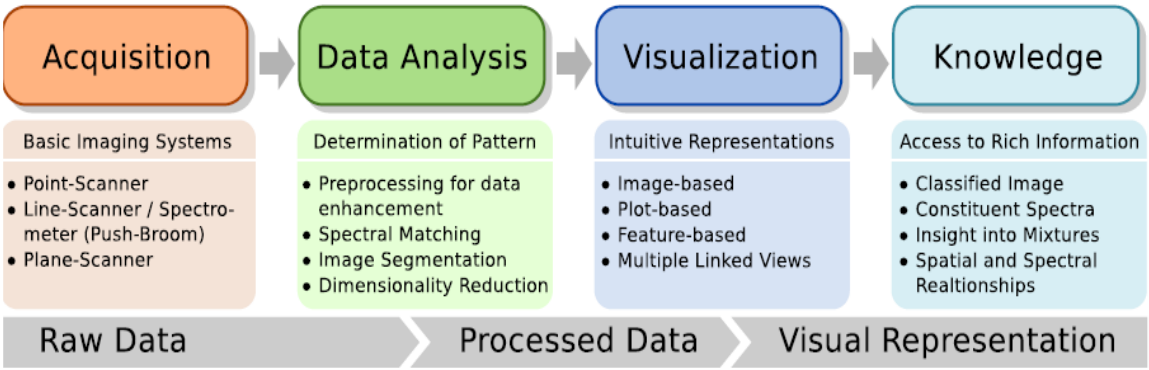


Fig I.6 - Overview of a multispectral image processing [3].

In image processing, texture analysis plays a very important role. It allows the identification of objects. It can be seen in all images from multispectral images to microscopic images of cell cultures or tissue samples.

The term texture is used to characterize the visual appearance of the surface of an object. This concept is therefore widely used in image analysis. Indeed, to distinguish, in an image, two different zones which have the same gray intensity, it is useful to look at the patterns characterizing these two zones. Texture analysis brings together a set of mathematical techniques to quantify the different levels of gray present in an image in terms of intensity or roughness and their distribution. The diversity of the images as well as the difficulty of giving a precise definition of the texture also allowed the emergence of several methods of texture analysis which can be classified essentially in three approaches [3]:

- Deterministic (structural) approach.
- Spatio-frequency approach.
- Stochastic or statistical approach.

I.3.1. Structural approaches

Structural methods seek to extract primitives from textures and their locations. These methods have therefore proven to be suitable for macrotextures, where a strong spatial structure emerges through a more or less repetitive pattern. Mainly, these methods use autocorrelation techniques to find the placement of the primitives of the textures initially extracted, in order to deduce a placement rule. The characteristic feature of these methods is that they all take place in two stages, the extraction of the primitive and then the search for the placement rule [3].

I.3.2. Spatio-frequency approaches

In spatio-frequency methods, representations preserve both global and local information; they are therefore well suited to quasi-periodic signals. Indeed, textures are quasi-periodic signals which have localized frequency energy. These methods make it possible to characterize the texture at different scales [2-6]. Among the spatio-frequency methods, we can cite:

- The Fourier transforms.
- Gabor filters considered as precursor filters in the field of spatio-frequency filtering methods.

- Discrete cosine Transform (DCT).
- The transformation into wavelets.

I.3.3. Statistical approaches

The statistical approach uses the spatial domain and takes into account the statistical aspect of the texture in the analysis procedures, since we consider that the texture does not have strong contours and does not have a basic isolatable pattern, but that it has, on the contrary, a random behavior. It then consists in evaluating, as textual information, the statistical properties of a region or a certain neighborhood around a pixel. This approach is thus adapted to micro textures and has been the subject of several studies which have given rise to a considerable number of methods [2-6] such as: Texture analysis using gray level co-occurrence matrix (GLCM).

I.4. Image Features extraction

Feature extraction phase assists the classification phase by looking for features that allow easy to distinguish between the different classes. Feature extraction is the operation of minimizing information by starting from a set of measured data and collecting the information without redundant, and facilitating the learning process. When the input data is too large to be processed, the feature extraction operation transform the data to feature vector contains relevant information to the input data [2-12].

Basically, there are two types of features are extracted from the images based on the application. They are local and global features. Thus, Feature extraction can be done by two methods:

- Global feature extraction.
- Local feature extraction.

The first type is global features extraction methods. The basic idea of this type is to represent the image in a single vector called the feature vector. Local features extraction method is the second type of our methods, it extracts relevant information from the image itself like shapes edges and color [2-9].

I.4.1. Local feature extraction methods

Extracting local features consists of describing visual information from a patch or neighborhood ($M \times M$ pixels) around a point of interest in the food item. As a result of this, for each type of local feature we form as many feature vectors as points of interest. There are several methods such as [2-6]:

I.4.1.1. Oriented gradient histogram (HOG)

The general idea of HOG is that it focuses on the structure or the shape of an object, HOG is able to provide the edge direction as well by extracting the gradient and orientation. This work is done when the complete image is broken down into smaller regions and for each region the HOG would generate a Histogram for each of these regions separately. The histograms are created using the gradients and orientations of the pixel values. We can go deeper in these steps as follow [4]:

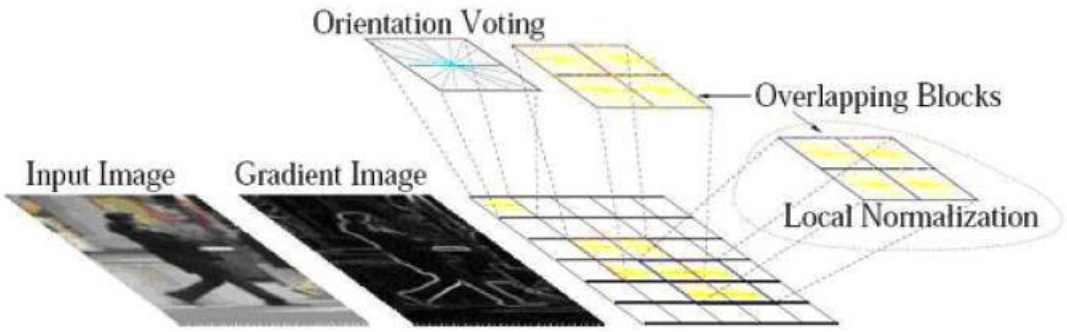


Fig I.7 –HOG features extraction process [4].

Specifically, this approach involves filtering the gray scale image with the following filter kernels:

$$D_x = [-1 \quad 0 \quad 1] \quad and \quad D_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{I.1}$$

So, given an image *I*, we obtain the x and y derivatives using a convolution operation:

$$I_x = I \times D_x \quad and \quad I_y = I \times D_y \tag{I.2}$$

Then the magnitude of the gradient is given by:

$$|G| = \left({I_x}^2 + {I_y}^2\right)^{0.5} \tag{1.3}$$

And orientation of the gradient is given by:

$$\theta = tan^{-1}\left(I_y / I_x\right) \quad (1.4)$$

I.4.1.2. Local binary patterns (LBP)

Local binary patterns were initially proposed by Ojala in 1996 to characterize the textures present in gray scale images [4, 12]. This method proved to be a powerful tool for extracting texture features. In this method, the target window is divided into a number of cells and the pixel is compared with its neighborhood pixels. If the pixel value is greater than or equal to the center pixel, the cells are labeled with either "0" or "1." After labeling the pixel values with the corresponding LBP codes, depending on the pixel values the histogram is calculated. After summing the histogram of all cells, the feature vector is obtained [4, 12].

The corresponding decimal value of the generated binary number is then used for labeling the given pixel. The derived binary numbers are referred to be the LBPs or LBP codes as shown in fig bellow:

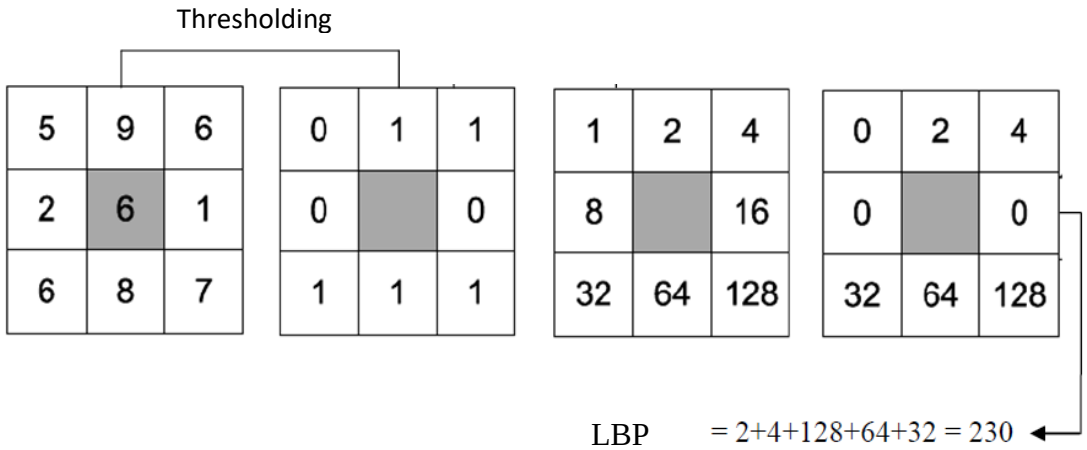


Fig I.8 – Example of the LBP working Principe [12].

Given a pixel i_c at coordinate (xc, yc), the resulting LBP can be expressed in decimal form as follows:

$$\mathrm{LBP(xc, yc)} = \sum_{P=0}^{P-1} s(i_P - i_c)2^P \quad (1.5)$$

Where i_c and i_P are, respectively, gray-level values of the central pixel and *P* surrounding pixels in the circle neighborhood ((p = 0, 1… 7) in the example), with a radius *R*, and function $s(x)$ is defined as:

$$s(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0. \end{cases} \quad (1.6)$$

I.4.2. Global feature extraction methods

I.4.2.1. Fourier transform (FT)

The Fourier Transform is an important image processing tool which is used to decompose an image into its sine and cosine components. The output of the transformation represents the image in the *Fourier* or frequency domain, while the input image is the spatial domain equivalent. In the Fourier domain image, each point represents a particular frequency contained in the spatial domain image. The Fourier Transform is used in a wide range of applications, such as image analysis, image filtering, image reconstruction and image compression [10]. Given that image processing concerns sampled data, we require the Discrete Fourier transform (DFT) as:

$$F(u,v) = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\exp(-j2\pi(\frac{mu}{M} + \frac{nv}{N}) \quad (1.7)$$

Where $f\ (m,n)$ is an image with$M\ X\ N$, $j = \sqrt{-1}$ u = 0. 1…M − 1, v =0. 1…N − 1. The inverse Transform is defined as:

$$f(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} F(u,v)\exp(j2\pi(\frac{mx}{M} + \frac{ny}{N})) \quad (1.8)$$

I.4.2.2. Discrete cosine transform (DCT)

The DCT was firstly introduced by Ahmed et al. in 1974. It helps to isolate the image into parts of differing importance based on the image's visual quality. Feature extraction by the DCT method, consists of two stages. First of all, we apply the DCT on the entire image to get a 2D coefficients matrix. In the second step only some of coefficients are selected. Those selected coefficients are the ones who determine the image features those two stages are being treated by the following steps [4, 12]:

1. Split the image into 8X8 blocks.
2. Applying dct block by block starting from top left to bottom right.
3. Each block is compressed through quantization.
4. Each block produces an array stored in reduced amount of space.

5. Building the image by decompression using inverse DCT.

The full equation of DCT [5]:

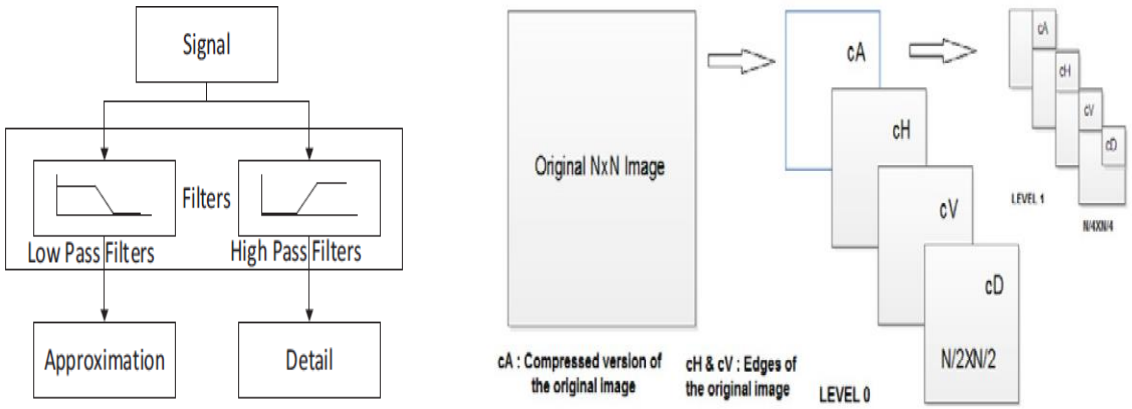
$$S(i,j) = \frac{1}{\sqrt{2N}} C(i)C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} s(x,y).\cos\left(\frac{(2x+1)i\pi}{2N}\right).\cos\left(\frac{(2y+1)j\pi}{2N}\right) \quad (1.9)$$

Where $C(u) = \begin{cases} \frac{1}{\sqrt{2}} & if\ u = 0 \\ 1 & if\ u > 0 \end{cases}$; $s(x,y)$ is the element in the image represented by the matrix and N is the size of the block. The equation calculates one entry (i, j) of the transformed image from the pixel value in the original image.

I.4.2.3. Discrete Wavelet Transform (DWT)

Discrete wavelet transform is used to extract characteristics from a signal on various scales proceeding by successive high pass and low pass filtering. The wavelet coefficients are the successive continuation of the approximation and detail coefficients. The basic feature extraction procedure consists of [6, 9]:

- Decomposing the signal using DWT into *N* levels using filtering and decimation to obtain the approximation and detailed coefficients.
- Extracting the features from the DWT coefficients.



(a) (b)

Fig I.9 – (a) Two channel signal filtering process of DWT; (b): Image decomposition using DWT at two level.

I.5. Conclusion

images are become an essential tool in data processing no matter what type of information it handles. The greater number of images the more data we can learn or extract from it. Some of the information might be invisible for the human eyes and needs a special effort to captors them and more complex features extraction methods like the one we seen. In classification issues it is simple to classify a ten or hundred images but when it comes to thousands of it. Becomes a true problem specially if there is a lot of classes to work with. This issue needs special and complex methods to do that in maximum accuracy and less time as possible. In the next chapter, we will discuss the concept of the multispectral image classification, using deep learning.

CHAPTER II

Images classification and Deep learning

II.1. Introduction

The concept of images classification is the operation of attributing the images of the dataset to an already known class (e.g. classifying Dogs vs Cats, cars vs trucks or classifying people by the skin color). This task seems to be easy in theoretical, but it's a very hard operation to do in practical especially when the amount of data is too large because it takes a lot of time. Machine learning is a branch of the artificial intelligence domain, designed in the first to imitate human learning process. Commonly used for classification problems. The Principe of the machine learning is a series of mathematical algorithms applied to an algorithm uses an initial data base with information to classify new objects none seen in the training data set. For such operation it is necessary to use a computer system capable of handling such tasks like machine learning [9-18].

Machine learning may be the best solution but with a few problems. Deep Learning can be summed up as a sub field of Machine Learning studying statically models called deep neural networks. A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. CNNs are powerful image processing, artificial intelligence (AI) that use deep learning to perform both generative and descriptive tasks, often using machine vision that includes image and video recognition.

In this chapter, our main subject is to give a general overview of the principles and approaches used in image classification. Then, we approach the concept of deep learning as well as the optimization algorithms and the choice of hyper parameters of the CNN algorithm. Finally, we end by giving the different metrics used to evaluate the performance of the classification algorithm.

II.2. Images classification

II.2.1. Definition of classification

Automatic image classification is a pattern recognition application that automatically assigns a class to an image using a classification system (see fig II.1). We thus find the classification of objects, scenes, textures, recognition of faces, fingerprints, and characters. There are two main types of learning depending on the information available on the data to be classified: supervised learning and unsupervised learning. In the supervised approach, each

image is associated with a label that describes its class of membership. In the unsupervised (or clustering) approach, the available data does not have labels; it is then up to the system to extract a membership rule from each image to a given group [12].

An automatic image classification system is made up of the following steps: the pre-processing phase; the feature extraction phase which describes the relevant information contained in the image using discriminating operators or descriptors and the learning phase which consists of building a decision rule. These three phases are essential in the construction of the classification system.

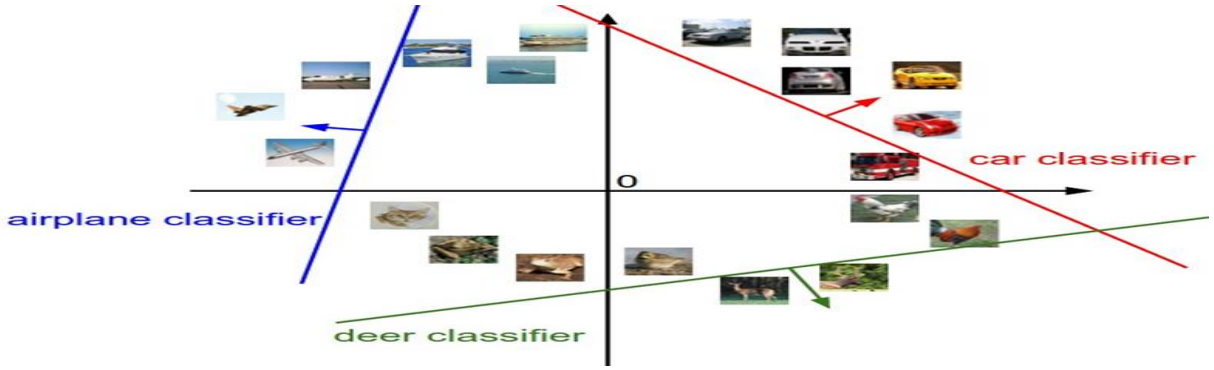


Fig. II.1 – Images classification.

II.2.2. Feature extraction

The feature extraction phase is usually one of the most important phases in building the system. We find low-level descriptors which are interested in the information contained in the image at pixel level (example: histogram, the co-occurrence matrix, etc.) and higher-level descriptors requiring an intermediate representation of the more suitable image, shapes and structures in the image, spatial relationships between pixels or these structures (example: Harris detector; Local Binary Pattern, Transformation) [7].

II.2.3. Classification methods

The purpose of the classification methods is to identify the class of belonging of objects defined by their description

II.2.3.1 Supervised classification

Supervised classification is an approach in which the computer program learns data transmitted to it by the user, and then uses this learning to classify a new observation. In this classification method, we already have examples of which the class is known and labeled.

Information on the data to be processed is available and is used to train the classification process.

This constitutes the learning phase of the model. This information, called the learning set, is generally made up of a set of individuals {characteristics, associated class}. This set is then learned by a classical supervised classification algorithm among which we cite: k-nearest neighbors (k-NN), neural networks, vector support machines (SVM), etc. Once the learning phase has been completed, the classification algorithm is then used to determine the classification of a set of test individuals, composed of a large number of samples.

This dataset can simply be bi-class (such as identifying whether the person is male or female) or it can be multi-class (voice recognition, handwriting recognition, biometric identification, document classification, etc.) [12].

II. 2.3.2 Unsupervised classification

Unsupervised classification (clustering) is a classification technique, where you do not need to supervise the model. Instead, you must allow the model to operate on its own to discover information. It mainly processes unlabeled data [12].

II.2.4. Types of classes

II.2.4.1. Binary class

The incoming data are classified into one of two possible categories. For example, for the diagnosis of a patient, we have two classes: sick {yes or no}.

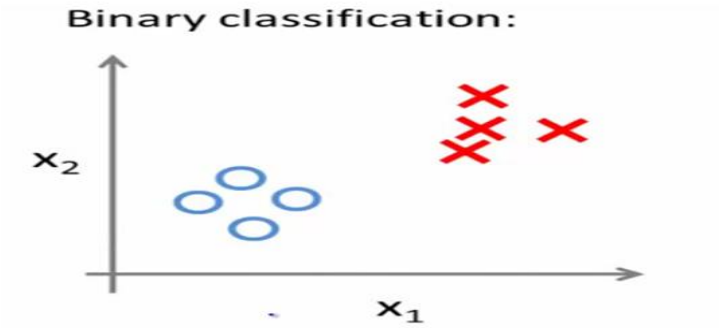


Fig II.2 – Binary classification

II.2.4.2. Multi-class

The data is divided into several possible categories (greater than 2). This type is very useful for large amounts of data. Example: the detection of emotions according to the face {happy, tired, sad ....}.

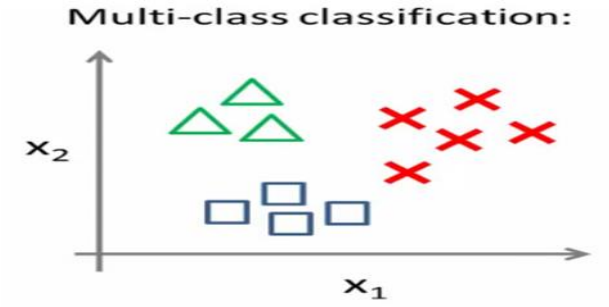


Fig II.3 – Multi-class classification.

II.3. Deep learning

II.3.1. Definition

Deep learning is a type of machine learning that aims to train and teach the computer to perform human functions such as distinguishing visual objects and identifying sound and image. Instead of organizing data, deep learning defines its own basic parameters that allow the machine to learn independently, and the current interest in deep learning is partly due to enthusiasm for artificial intelligence. Deep learning techniques have improved the ability to classify, recognize, etc. [7-18].

Fig II.4 – deep learning concept.

Deep learning is a subcategory of machine learning, it concerns the use of neural networks to improve things such as speech recognition, computer vision and natural language processing (all areas particularly complex for AI researchers) [10].

II.3.2. Difference between deep learning and machine learning

In practical terms, deep learning is just a subset of machine learning. In fact, deep learning technically is machine learning and functions in a similar way (hence why the terms are sometimes loosely interchanged). However, its capabilities are different.

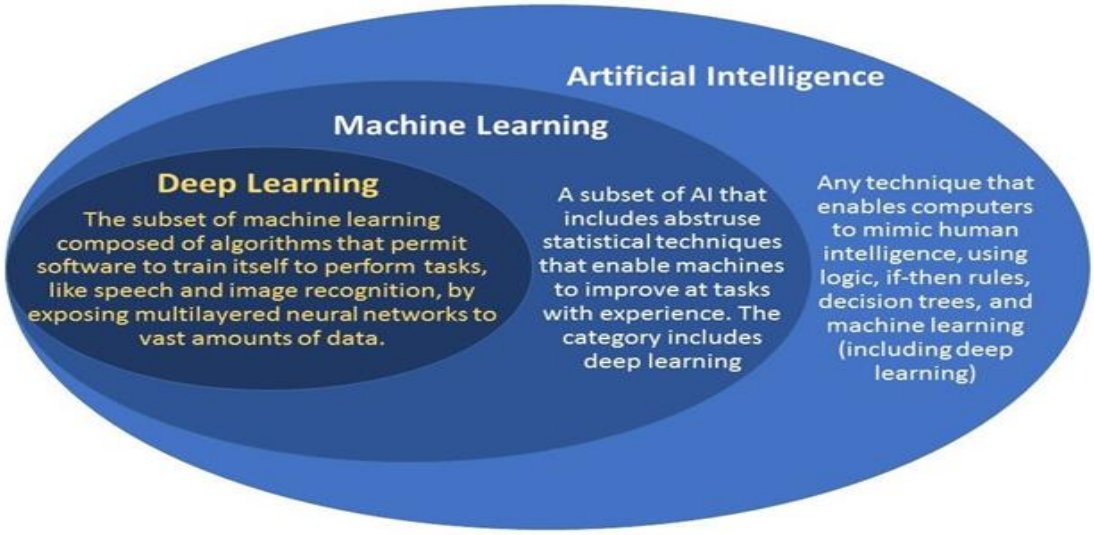


Fig II.5 –deep learning and machine learning origins [13].

While basic machine learning models do become progressively better at whatever their function is, they still need some guidance. If an AI algorithm returns an inaccurate prediction, then an engineer has to step in and make adjustments. With a deep learning model, an algorithm can determine on its own if a prediction is accurate or not through its own neural network (see figure below).

Fig II.6 Difference between machine learning and deep learning [11]

Deep Learning is used in many fields such: Image recognition, Automatic translation, Autonomous car, Medical diagnosis, personalized recommendations, Automatic moderation of social networks, Financial prediction and automated trading, Identification of defective parts, Conversational agents, Space exploration, Intelligent robots [10].

II.3.3. Why the choice of deep learning?

First of all, the different deep learning algorithms only appeared when machine learning failed to try to solve a wide variety of artificial intelligence (AI) problems:

- To enhance the development of traditional algorithms in such AI tasks.
- To develop a large amount of data such as big data.
- To adapt to any type of problem
- To extract the characteristics automatically

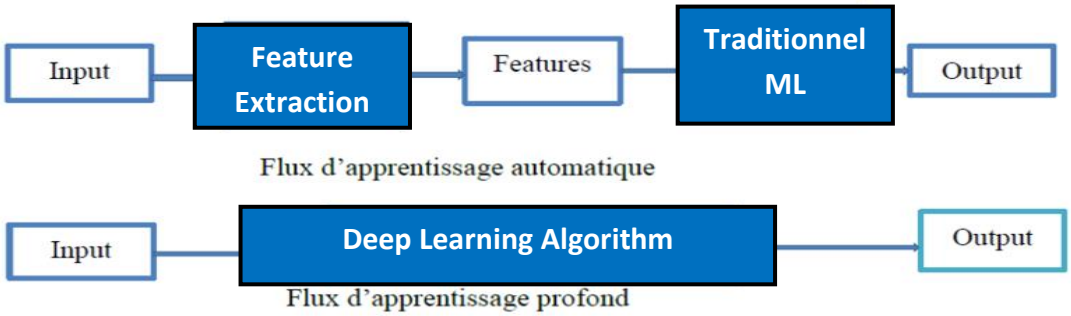


Fig II.7 Comparison between machine learning and deep

II.3.4. How deep learning works

Deep Learning is based on a network of artificial neurons inspired by the human brain. This network is made up of several "layers" of neurons, each receiving and interpreting information from the previous layer. For example, the system will learn how to determine if there is a face in a photo before discovering which person. Figure II.7 illustrates the structure of an artificial neuron. Each neuron receives values from upstream neurons through its synaptic connections and processes these values via a combining function. The result of the combination is then transformed by the activation function to produce its output. This function allows the result to be thresholded to obtain a binary output.

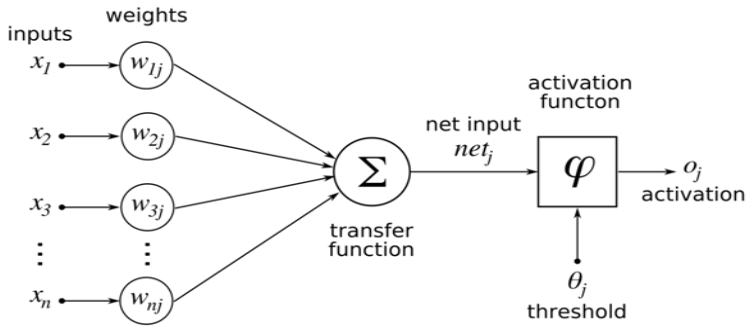


Fig II.8 Structure of artificial neurons used in the Artificial Neural

Deep Learning discovers a complex structure in large data sets using the back-propagation algorithm to indicate how a machine should modify its internal parameters which are used to calculate the representation in each layer from the representation in the previous layer. At each step, the "wrong" answer is eliminated and returned to the upstream level to adjust the mathematical model. Over time, the program reorganized the information into more complex elements. When this model is later applied to other situations, it is usually able to recognize a cat without anyone telling them that they have never learned the concept of a cat. Basic data is essential: the more experience the system has, the more efficient it will be.

The basis of deep learning is distributed representation in machine learning. "Distributed" means the assumption that the observation is the result of an interaction between different factors. Deep learning also requires that such an interaction can be divided into several layers, which means the multiple abstraction of the observed value [14].

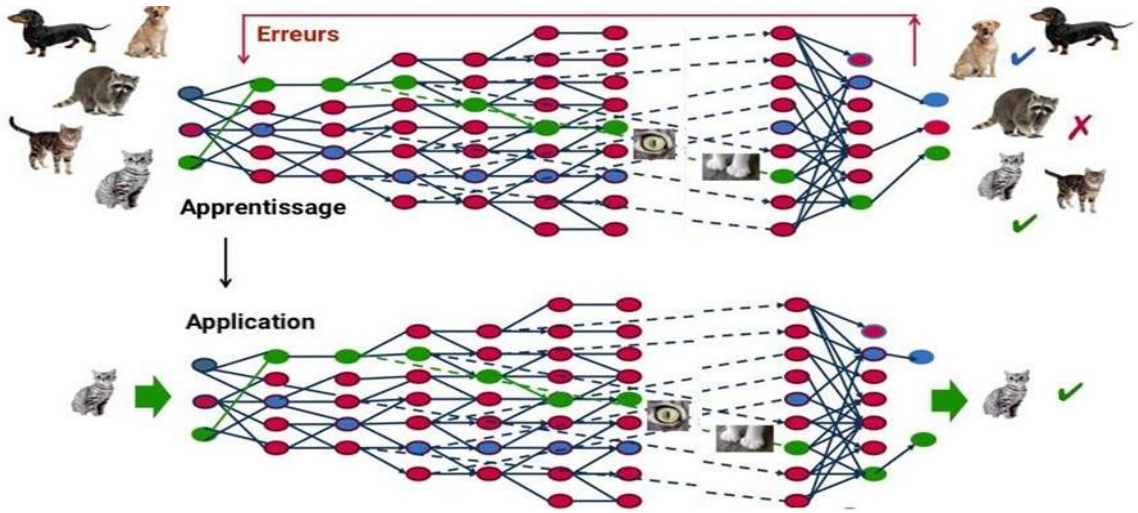


Fig II.9 –Deep learning working example [8].

II.3.5. Deep learning architecture

There are several deep learning frameworks already widely used, such as the deep neural network, the convolutional neural network (CNN) and the recursive neural network (RNN).

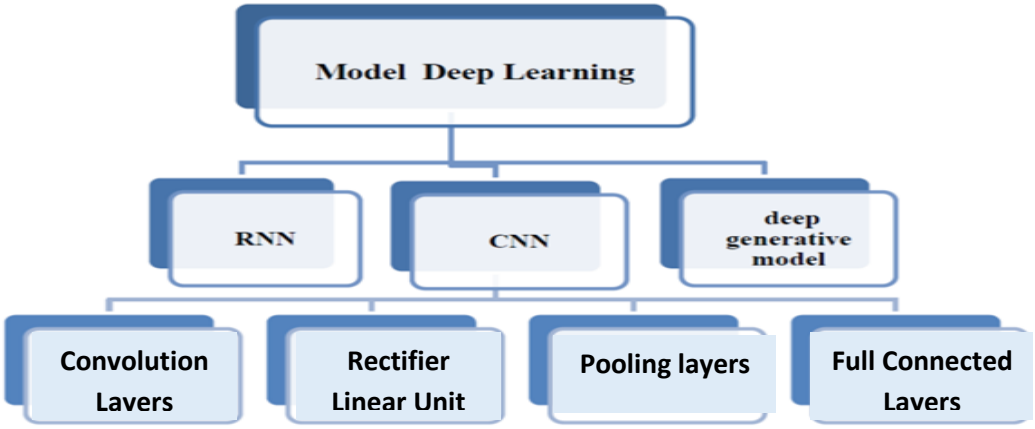


Fig II.10 –Models of deep learning and the architecture of CNN model.

II.4. Convolutional Neural Network (CNN)

II.4.1. Definition

The brain's basic computing unit called a neuron. About 86 billion neurons are found in the human nervous system and they are connected with about 10^{14}- 10^{15}synapses. Each neuron receives input signals from its dendrites and produces an output signal along its axon. The linear layer is a simplification of a group of neurons whose dendrites are connected to the same inputs (see figure II.11). The capacity of the neural networks to approximate any functions, especially non-convex, is directly the result of the non-linear activation functions (s*igmoid, Hyperbolic Tangent, Rectified Linear Unit*).

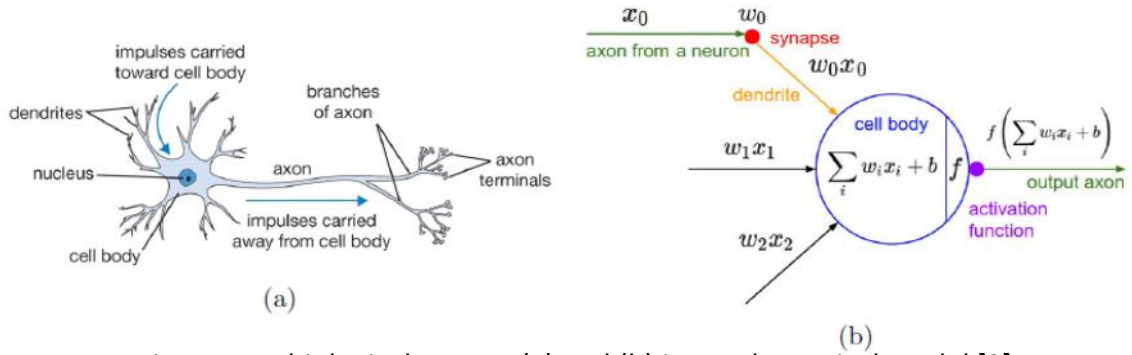


Fig II.11 –A biological neuron (a) and (b) its mathematical model [9].

A neural network is a system of hardware and/or software patterned after the operation of neurons in the human brain. Traditional neural networks are not ideal for image processing and must be fed images in reduced-resolution pieces. CNN have their “neurons” arranged more like those of the frontal lobe, the area responsible for processing visual stimuli in humans and other animals. The layers of neurons are arranged in such a way as to cover the entire visual field avoiding the piecemeal image processing problem of traditional neural networks (see Fig II.12).

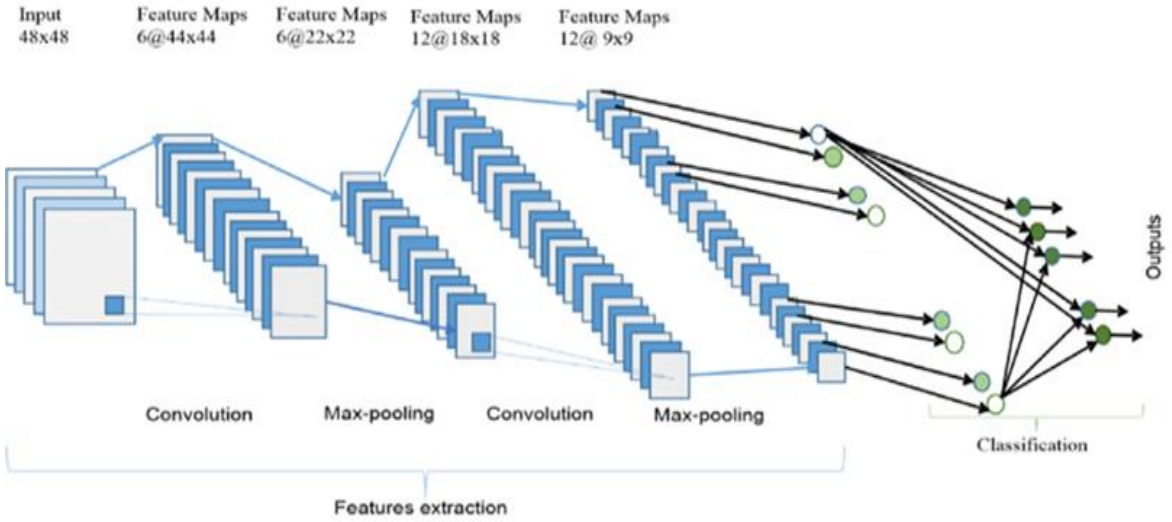


Fig II.12 –Convolutional neural network architecture [9].

A CNN uses a system much like a multilayer perceptron that has been designed for reduced processing requirements. The layers of a CNN consist of an input layer, an output layer and a hidden layer that includes multiple convolutional layers, pooling layers, fully connected layers and normalization layers (see figure II.13). The removal of limitations and increase in efficiency for image processing results in a system that is far more effective, simpler to trains limited for image processing and natural language processing [15].

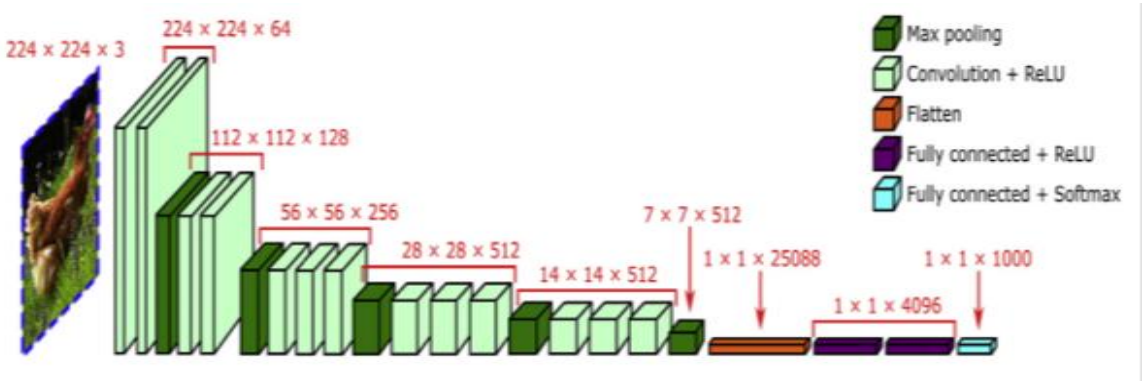


Fig II.13 –Convolutional neural network for image classification [15]

II.4.2. Convolutional neural network layers

II.4.2.1. Convolutional layer

It's the backbone of CNN's networks. Its purpose is to identify a set of features of an image received as input. For this, we perform a convolution filtering: in principle is to "drag" the window representing the feature (the filter) on the image and calculate the convolution product between the filter and each part of the scanned image. So, this layer receives several images as input and uses each filter to calculate the convolution of each image. The filters correspond exactly to the characteristics that we want to find in the image.

We obtain an activation map (feature map) for each pair (image, filter), which tells us where the feature is in the image: the higher the value, the more positions in the image that match the filter.

In explanation let's take an image of 64 by 64. The computer or the convolution layer will see it as a 64 by 64 matrix of numbers. Each number is a pixel value presenting the image. The filter size as in example will be a 3 by 3 window that scan the entire image to extract the feature map [15].

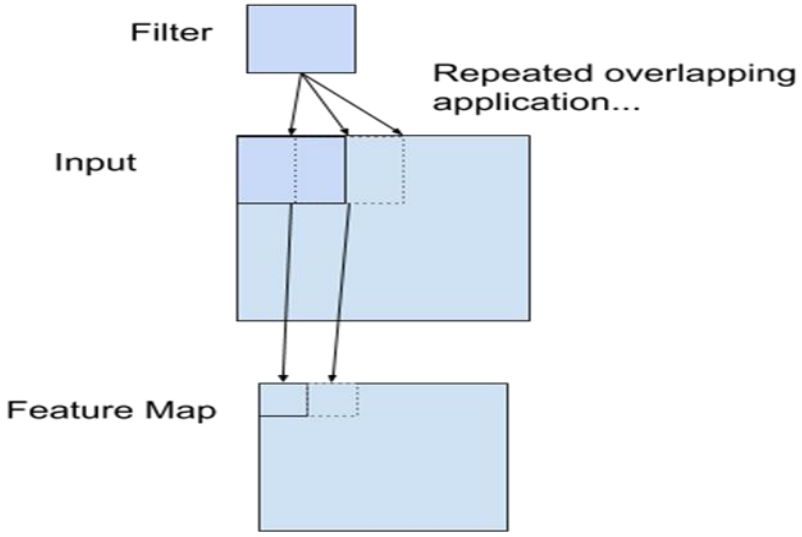


Fig II.14 – Convolutional layer [15].

II.4.2.2. Pooling layer

It receives several input feature maps and applies a pooling operation (subsampling) to each feature map. It allows the reduction of the image size taking into account its important characteristics.

The principle is to cut the image into regular cells, then we keep the maximum value in each cell relative to the filter used (2 x 2 pixels, 3x3 ...). So small square cells are often used so as not to lose too much information.

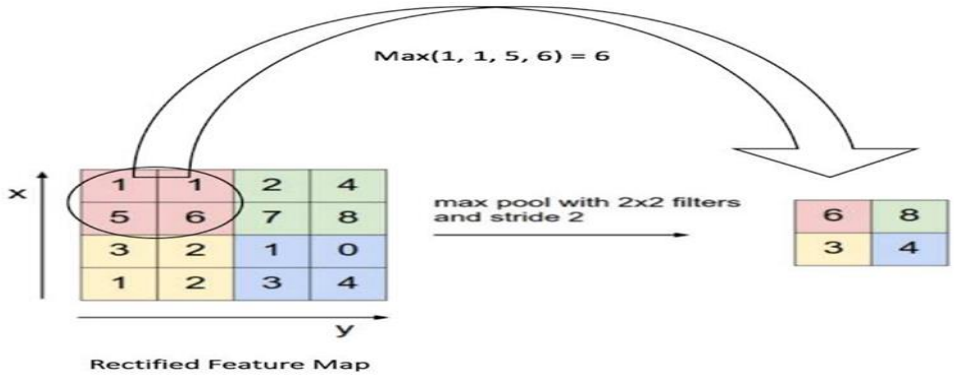


Fig II.15 –Pooling layer.

We get the same number of features map as the entry, but they are much smaller. This not only speeds up calculations, but also avoids the problem of over-fitting.

There are two main types of pooling: max and min. As the name suggests, maximum pooling is based on the detection of the maximum value in the selected region and the minimum pooling on the detection of the minimum value in the selected region [10].

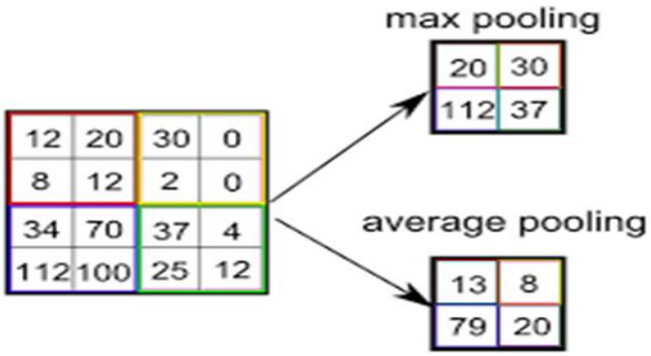


Fig II.16 –Max and min pooling layer.

II.4.2.3. Fully connected layer (FC)

Fully connected layer is a group of neurons connected all of them together from the previous layer takes the output of them, “flattens” the output shape and turns them into a single vector that can be an input for the next stage [10].

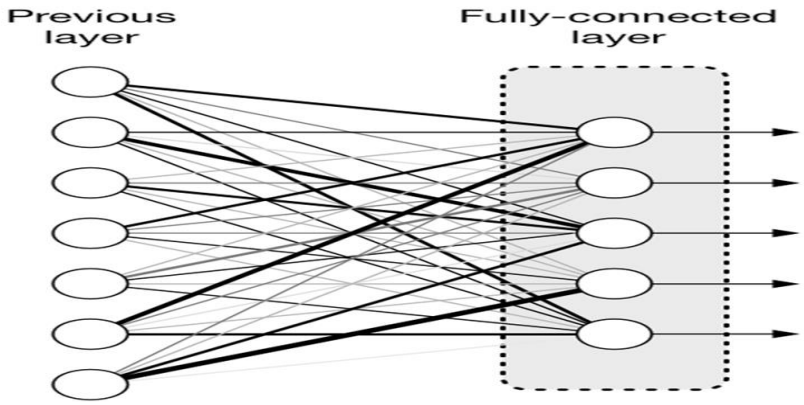


Fig II.17 – Fully connected layer.

II.5. Activation functions

Activation functions are a mathematical equation used at the end of each neuron in the neural network to compute the sum of the weights and biases. The activation function decides if the neuron fires or not. Neural network is used in plenty of domains such as recognition, classification, speech recognition, weather and medical diagnostic besides other domains.

Activation function has a linear type and a nonlinear type according to its domain of use to decide the fired neuron [16].

If the function used is a linear equation. Then we can summarize in this form of equation:

$$f(x) = w^T x + b \quad (2.1)$$

Where *x* = input, *w* = weights, *b* = biases, T=number of neurons.

In other situation linear equation can't be applied so the application of nonlinear equation is truly needed. As it's explained in this form:

$$y = \alpha(w1\,x1 + w2\,x2 + .. + wn\,xn + b) \quad (2.2)$$

Where *x* = input, *w* = weights, *b* = biases, α =Activation function.

II.5.1. Sigmoid

The main reason why we use sigmoid function is because it exists in range of (0 to 1). Therefore, it is especially used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1 [16].

$$f(x) = \frac{1}{(1 + e^{-x})} \quad (2.3)$$

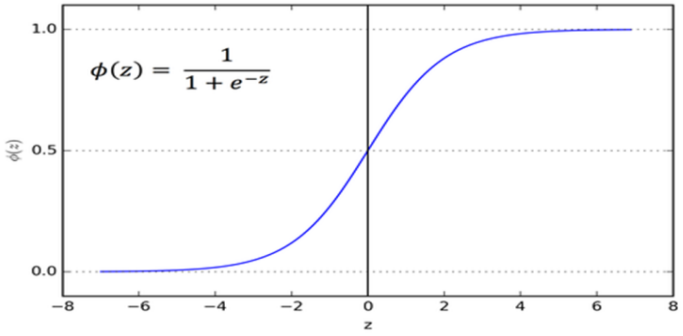


Fig II.18 – Sigmoid function plot [16].

II.5.2. Hyperbolic Tangent (*TanH*)

The TanH function is defined as follows: It is nonlinear in nature, so we can stack layers. It is bound to the range (-1, 1) the gradient is stronger for TanH than sigmoid (derivatives are steeper) Like sigmoid, TanH also has a vanishing gradient problem [16].

$$f\ (x)\ = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

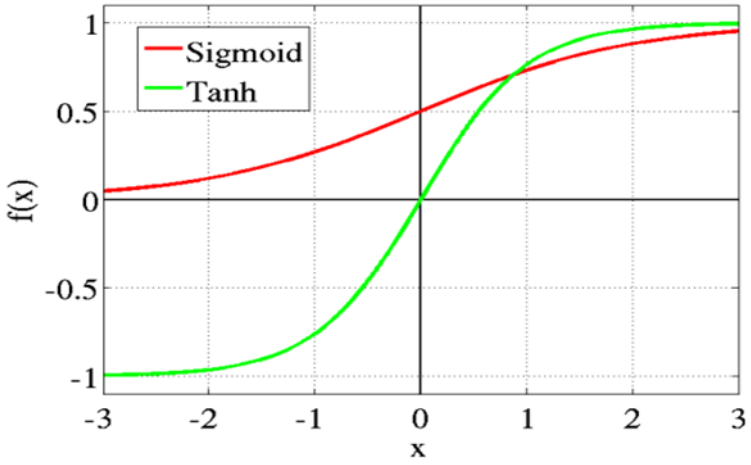


Fig II.19 – TanH function plot (comparison with sigmoid) [16].

II.5.3. Softmax

In mathematics, the Softmax function, also known as softargmax or normalized exponential function, is a function that takes as input a vector of K real numbers, and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers [16].

$$f\ (x)\ = \frac{e^{x}}{\sum_{i=0}^{n} e^{x}} \qquad (2.4)$$

Where x = input, n = number of neurons.

II.5.4. Rectified linear unit (*Relu)*

Relu stands for rectified linear unit, and is a type of activation function. Mathematically, it is defined as y = max (0, x). ... Relu is the most commonly used activation function in neural networks, especially in CNNs. If you are unsure what activation function to use in your network, Relu is usually a good first choice [16].

$$f\ (x) = \ max\ (0, x) = \begin{cases} I\ i\ f\ x\ \geq\ 0 \\ 0\ if\ \ x\ <\ 0. \end{cases} \qquad (2.5)$$

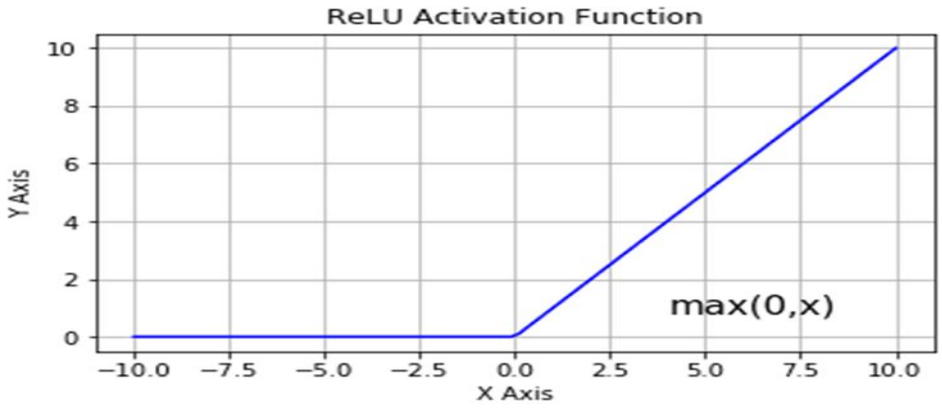


Fig II.20 – Relu function plot [16].

II.5.5. Softplus

Softplus is a newer function than sigmoid and TanH. It is firstly introduced in 2001. Softplus is an alternative of traditional functions because it is differentiable and its derivative is easy to demonstrate [16].

$$f\,(x)\;=\;log(1\;+\;e^{X}\,) \tag{2.6}$$

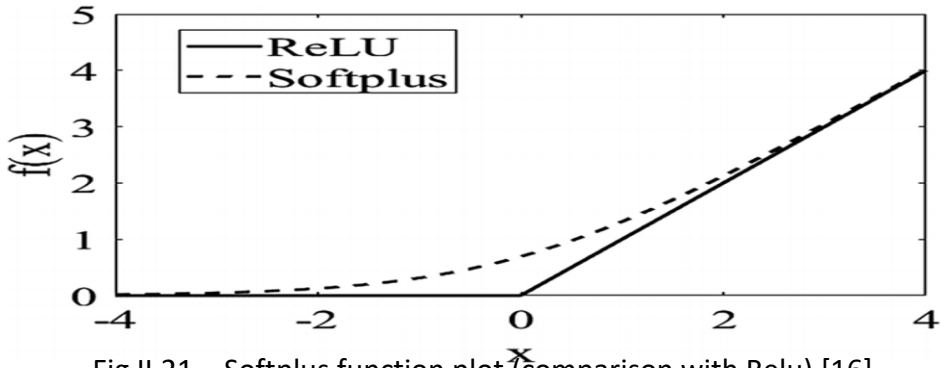


Fig II.21 – Softplus function plot (comparison with Relu) [16].

II.6. Optimizers algorithms

Optimizers update the weight parameters to minimize the loss function. They tie together the loss function and model parameters by updating the model in response to the output of the loss function. In simpler words, optimizers shape the model into its most accurate possible form by adjusting the weights [17].

II.6.1. Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) on the other hand performs a parameter update for each training example. It is usually much faster technique. It performs one update at a time [17].

$$x_{k+1} = x_k - tk\Delta f(x_k)^{(i)} \tag{2.7}$$

II.6.2. RMSprop

RMSprop is a special version of Adagrad developed by Professor Geoffrey Hinton in his neural nets class. Instead of letting all of the gradients accumulate for momentum, it only accumulates gradients in a fixed window. RMSprop is similar to Adaprop, which is another optimizer that seeks to solve some of the issues that Adagrad leaves open [17].

$$E[\Delta f(x)^2\,]k = \rho E[\Delta f(x)^2\,]k - 1 + (1 - \rho)\Delta f(x_k)^2$$
$$x_{k+1} = x_k - \frac{t}{q\,E[\Delta f(x)^2\,]k + \varepsilon}\ \Delta f(x_k) \tag{2.8}$$

Where ρ is a decay constant (e.g., 0.9).

II.6.3. Adagrad

Adagrad adapts the learning rate specifically to individual features: that means that some of the weights in your dataset will have different learning rates than others. This works really well for sparse datasets where a lot of input examples are missing. Adagrad has a major issue though: the adaptive learning rate tends to get really small over time. Some other optimizers below seek to eliminate this problem [17].

$$G_k = G_{k-1} + \Delta f(x_k)^{\,2}$$
$$x_{k+1} = x_k - \frac{t}{\sqrt{G_k + \varepsilon}}\ \Delta f(x_k) \tag{2.9}$$

Where G is the accumulation of the history gradients, and ε is a smoothing term that avoids division by zero (can be 1e − 6)

II.6.4. Adadelta

It is an extension of AdaGrad which tends to remove the *decaying learning Rate* problem of it. Instead of accumulating all previous squared gradients, *Adadelta* limits the window of accumulated past gradients to some fixed size w [17].

$$E[\Delta f(x)^2]_k = \rho E[\Delta f(x)^2]_{k-1} + (1-\rho)\Delta f(xk)^2$$
$$x'_k = -\frac{\sqrt{E[x'^2]_{k-1}+\varepsilon}}{\sqrt{E[\Delta f(x)2]_k+\varepsilon}}\Delta f(x_k) \quad (2.10)$$
$$E[x'2]_k = \rho E[x'^2]_k + (1-\rho)x'^2_k$$

$$x_{k+1} = x_k + x'_k$$

Where ρ is a decay constant (e.g., 0.95) and ε is a small value (e.g., 1e-6) for numerical stability.

II.6.5. Adam and Adamax (adaptive moment estimation)

Adam stands for adaptive moment estimation, and is another way of using past gradients to calculate current gradients. Adam also utilizes the concept of momentum by adding fractions of previous gradients to the current one. This optimizer has become pretty widespread, and is practically accepted for use in training neural nets. And Adamax It is a variant of Adam based on the infinity norm. Adamax is sometimes superior to Adam, especially in models with embeddings. An embedding is a relatively low-dimensional space into which you can translate high-dimensional vectors. Embeddings make it easier to do deep learning on large inputs like sparse vectors representing words. An embedding can be learned and reused across models [17].

$$m_k = \beta 1 m_{k-1} + (1-\beta 1)\Delta f(x_k)$$
$$v_k = \beta 2 v_{k-1} + (1-\beta 2)\Delta f(x_k)^2$$
$$m'_k = \frac{m_k}{1-\beta_1^k}, v'_k = \frac{v_k}{1-\beta_2^k} \quad (2.11)$$
$$x_{k+1} = x_k - \frac{t}{\sqrt{v'_k+\varepsilon}} m'_k$$

Where β1 can be 0.9, β2 can be 0.999, and ε can be 1e − 8.

II.7. Deep Neural Network: Hyper-Parameters

Model Hyper parameters are instead properties that govern the entire training process. They include variables which determine the network structure (for example, Number of Hidden Units) and the variables which determine how the network is trained (for example, Learning

Rate). Model hyper parameters are set before training (before optimizing the weights and bias). For example, here are some model inbuilt configuration variables [13-18]:

- Learning Rate.
- Number of Epochs.
- Hidden Layers.
- Hidden Units.
- Activations Functions.

Hyper parameters are important since they directly control behavior of the training algorithm, having important impact on performance of the model under training. Choosing appropriate hyper parameters plays a key role in the success of neural network architectures, given the impact on the learned model. Hyper parameters can be roughly divided into 2 categories:

- Optimizer hyper parameters.
- Model Specific hyper parameters.

II.7.1 Optimizer hyper parameters [13-18]

- **Learning Rate**

The learning rate, controls the step size of the optimization algorithm and is the most important hyper-parameter, the optimal learning rate is usually close to the largest learning rate which does not cause divergence of the training criterion.

- **Mini-Batch Size**

Mini batch size is the number of sub samples given to the network after which parameter update happens. A good default for batch size might be 32**.** Also try 32, 64, 128, 256, and so on.

- **Number of Epochs**

It is used to choose the right number of epochs for the training step, the metric we should pay attention to is the Validation Error. Number of Epochs determines the maximum number of update steps which are performed during training. This parameter is not set explicitly, but will be found dynamically during the training procedure.

II.7.2 Model Specific hyper parameters

They are more involved in the structure of the model:

- The number of hidden-units.
- First hidden layer.
- Number of layers.

The process of finding most optimal hyper parameters in machine learning is called hyper parameter optimization. There are several methods used to find out hyper parameters:

- Manual Search.
- Grid Search.
- Random Search.
- Bayesian Optimization.

II.8. Classification performance evaluation

Evaluations of performances are a group of parameters calculated to determine if the model or the program is a high performance. The evaluation is based on several parameters which the more perfect they were the more efficiency the model is [18-20].

II.8.1. False Acceptance Rate (FAR)

False acceptance rate is the number of the wrong clients or the wrong identification or recognition they have been accepted or classified as true clients. Can be calculated by dividing the number of the wrong clients by the total number of clients [18-20]:

$$FAR = \frac{FA}{Nc} \quad (2.12)$$

Where: FA = number of wrong clients, NC = total number of clients.

II.8.2. False Rejection Rate (FRR)

False Rejection rate are the number of the true clients who are rejected by the system. Calculated by dividing the rejected clients by the total number of clients [[18-20]].

$$FRR = \frac{FR}{Nc} \quad (2.13)$$

Where: FR = number of true rejected clients, NC = total number of clients.

II.8.3. Equal Error Rate (EER)

Equal Error rate is the point where the FAR and FRR are equal. The low EER is a proof that the model is high accurate [[18-20]].

$$FAR = FRR \tag{2.14}$$

To compare classifiers, there are various others statistics based on the confusion matrix namely: sensitivity and specificity, which are estimated respectively by the proportion true positives and the proportion of true negatives:

$$Sensibility = \frac{TP}{TP+TN} ; \tag{2.15}$$

$$Specificity = \frac{TN}{TN+FP} \tag{2.16}$$

$$with \begin{cases} TP + FN = P \\ TN + FP = N \end{cases}$$

With TP : True Positif ; FN: False Negative; TN: True Negative and FP: False positive

II.8.4. Accuracy (ACC)

Accuracy is calculated as the number of all correct predictions divided by the total number of the dataset. The best accuracy is 1.0, whereas the worst is 0.0. It can also be calculated by$1 - ERR$ [19]. Formally, accuracy has the following definition:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \tag{2.17}$$

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

$$ACC = \frac{TP+TN}{TP+TN+FN+FP} = \frac{TP+TN}{P+N} \tag{2.18}$$

II.8.5 Receiver Operator Characteristic (ROC)

A Receiver Operator Characteristic (ROC) curve is a graphical plot used to show the diagnostic ability of binary classifiers. Now used in many other areas such as medicine, radiology, natural hazards and machine learning [[18-20]]. From the ROC curve, the index to numerically evaluate the curve is the area under the ROC curve (AUC, Area under the Curve). This index can be analyzed as the probability of the model to be predicted correctly.

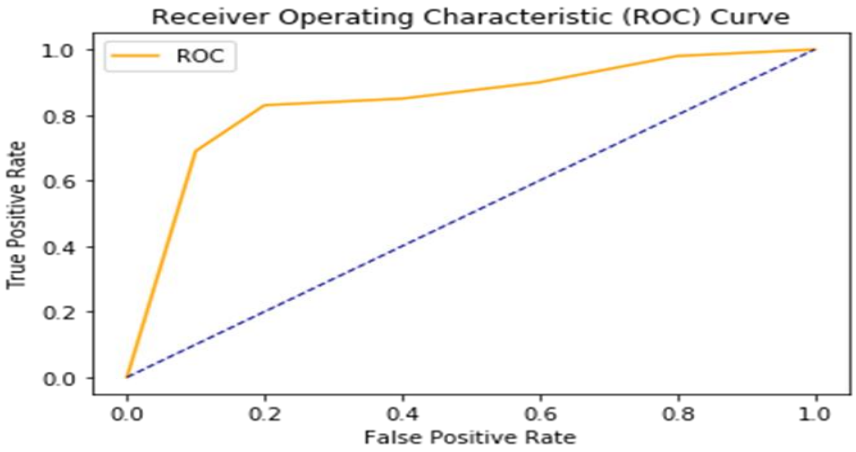


Fig II.22 - Receiver Operator Characteristic (ROC) curve.

II.8.6. Confusion Matrix

In supervised machine learning, the confusion matrix is a matrix that measures the performance of a classification model (or "classifier"), on a set of test data. Figure II.23 illustrates some of the main variables in the confusion matrix.

		Predicted class: Positive	Predicted class: Negative
Observed Class	Positive	True Positive (TP)	False positif (FP)
	Negative	False négatif (FN)	True negative (TN)

Fig II.23 – Confusion matrix.

II.8.7. Epoch parameter

In terms of artificial neural networks, an epoch refers to one cycle through the full training dataset. Usually, training a neural network takes more than a few epochs. In other words, if we feed a neural network the training data for more than one epoch in different patterns, we hope for a better generalization when given a new "unseen" input [19]. An epoch

is often mixed up with an iteration. Iterations is the number of batches or steps through partitioned packets of the training data, needed to complete one epoch.

II.8.8. Loss parameter

Loss function is used to quantify the capacity of the network to approximate the ground truth labels for all training inputs, we define a loss function which takes as inputs the weights, biases, and examples from the training set. For instance, the loss could be the number of images correctly classified. However, the most efficient way to find the weights and biases, regarding the number of parameters, is to use an algorithm similar to the Stochastic Gradient Descent (SGD). If predictions deviate too much from actual results, loss function would cough up a very large number. Gradually, with the help of some optimization function, loss function learns to reduce the error in prediction [19].

The three most used loss function to train deep neural networks for classification are:

- ***Mean Square Error (MSE)*** It is a multi-class loss formerly used to train neural networks.

$$Loss(x, y) = \frac{1}{n} \sum_i |x_i - y_i|^2 \quad (2.17)$$

With *x* a vector of n predictions, and *y* a binary vector full of 0 besides a 1 in the corresponding class dimension.

- ***Cross Entropy*** It is a multi-class loss which is nearly a better choice than MSE.
- ***Loss Multi Label*** It is the adaptation of the Cross-Entropy loss for multi-label classification. It is a multi-label one-versus-all loss based on max-entropy.

II.9. Conclusion

In this chapter, the concepts of classification as well as the methods commonly used for classification have been presented in a synthetic manner. Then, we described the deep learning algorithm as well as these optimization algorithms, these activation functions as well as the choice of hyper parameters used in the architecture of the CNN algorithm. Finally, we have described the statistics derived from the ROC curves for the quantitative evaluation of the different classification models. The following chapter describes the results obtained within the framework of the study of image classification based on deep learning, using the python language and testing on several databases (gray scale image, color images and multispectral images).

CHAPTER III

Practical work: Results and Interpretations

III.1. Introduction

After studying in the second part the different parameters in Deep Learning and the Cnn layers, we will present in this chapter an image classification approach, based on convolutional neural networks. For this, we used several models with different architectures and different optimization techniques. The results obtained will be given in terms of precision and error. A comparative study of the performance of image classification using deep learning on several image databases and a Knn classification using LBP feature extraction method will be developed in this chapter. To implement the proposed approach, we used the python language.

III.2. Image Data sets Description

III.2.1. Cats and dogs Datasets [21]

Microsoft kaggle cats and Dogs dataset are a public and open source database. It's a riche, wide and contains a lot of examples. The content of this data base is divided in two categories named cats and dogs. The total number of samples is 25000 with 12500 samples for each category. Each image has taken in different angles from different distances, in a lot of places and positions. The images are RGB images, in our work, we used them as a grayscale image. The size and the quantity of the information in each image are different from one to another (see fig III.1).

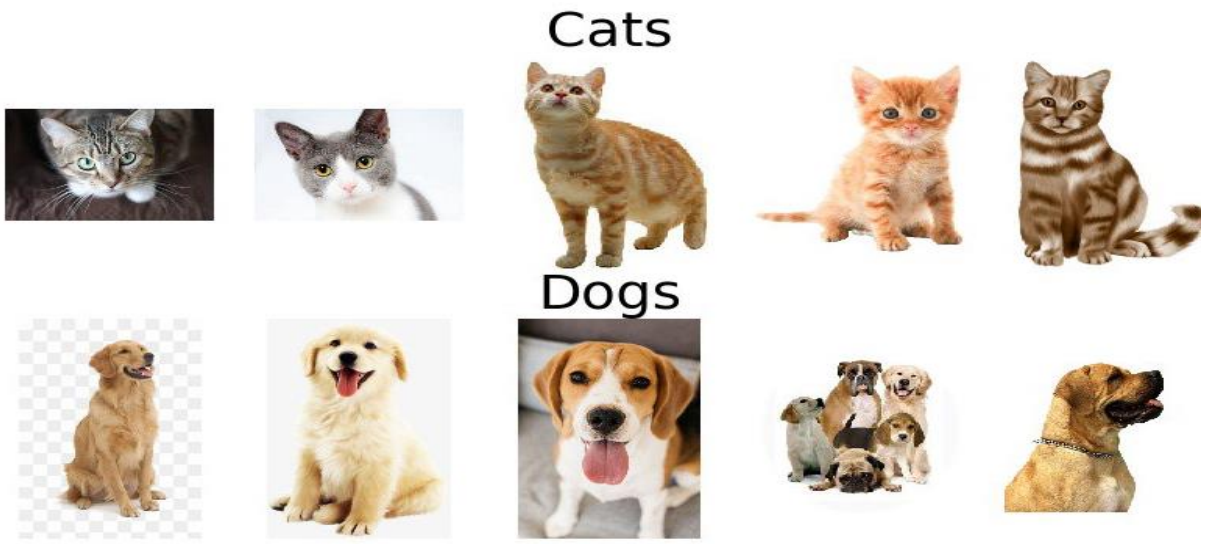


Fig III.1 – Cats and dogs data set presentation.

III.2.2. Weather Images Datasets [22]

Weather dataset is one of the multi-class datasets that contain three different categories of the weather cases: Cloudy, Rainy and the Sunshine with a 215 sample for each category. This dataset can be used as multi-class or a binary class by excluding one category. The images are in deferent sizes and angles (see fig III.2).

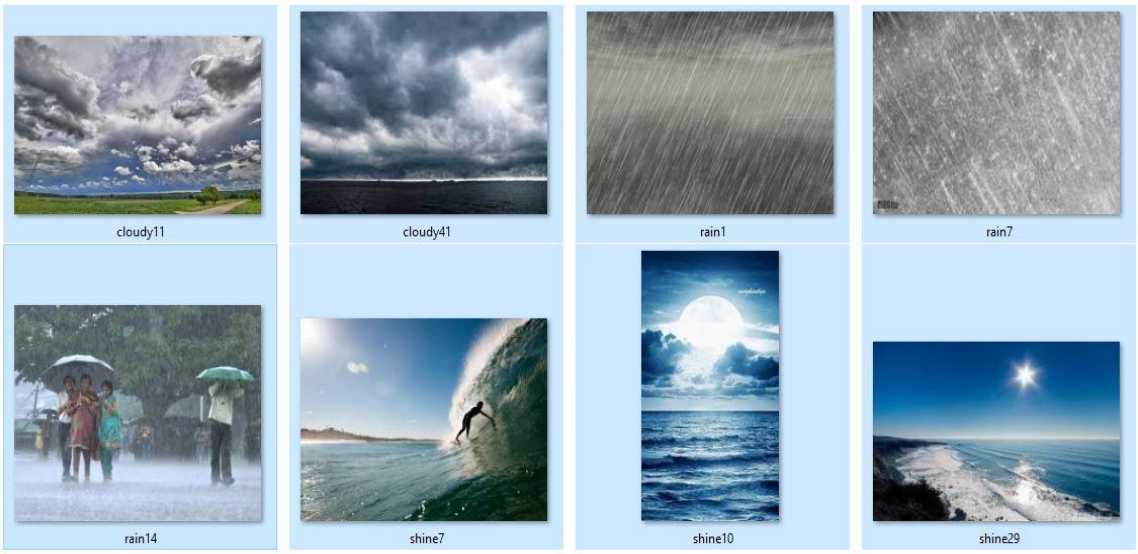
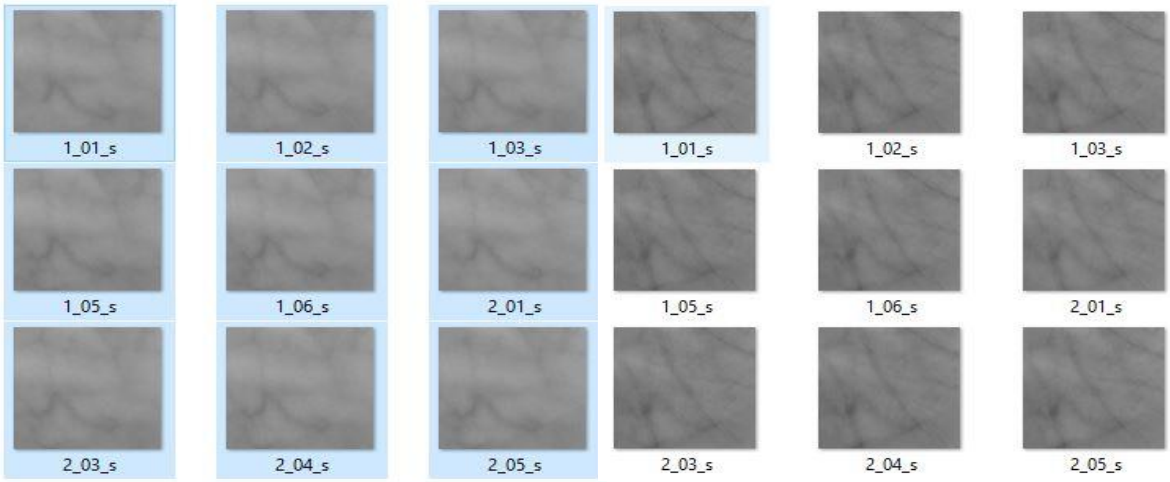


Fig III.2 – Weather data set presentation.

III.2.3. PolyU multispectral palmprint Dataset [23]

The Biometric Research Centre (UGC/CRC) at The Hong Kong Polytechnic University has developed a real time multispectral palmprint data base with blue, green, red and near-infrared (NIR) illuminations. Multispectral palmprint images were collected from 250 volunteers, including 195 males and 55 females. The age distribution is from 20 to 60 years old. We collected samples in two separate sessions. In each session, the subject was asked to provide 6 images for each palm. Therefore, 24 images of each illumination from 2 palms were collected from each subject. In total, the database contains 6,000 images from 500 different palms for one illumination (see fig III.3). In our work, we used only two bands: Red and NIR bands illumination.



(a) (b)

Fig III.3 – (a) PolyU NIR data set presentation; (b) PolyU RED data set presentation.

III.2.4. CASIA Multispectral palmprint Dataset [24]

The CASIA multispectral palmprint database is a collection of 7,200 palm images for 100 different volunteers, each subject contains 6 deferent illuminations, 12 images for illumination, for both genders in deferent ages. The 6 lights of the database are 460nm 630nm 700nm 850nm 940nm and the visible WHT. All in the same size and can be used in grayscale. Between two samples, we allow a certain degree of variations of hand postures (see fig. III.4).

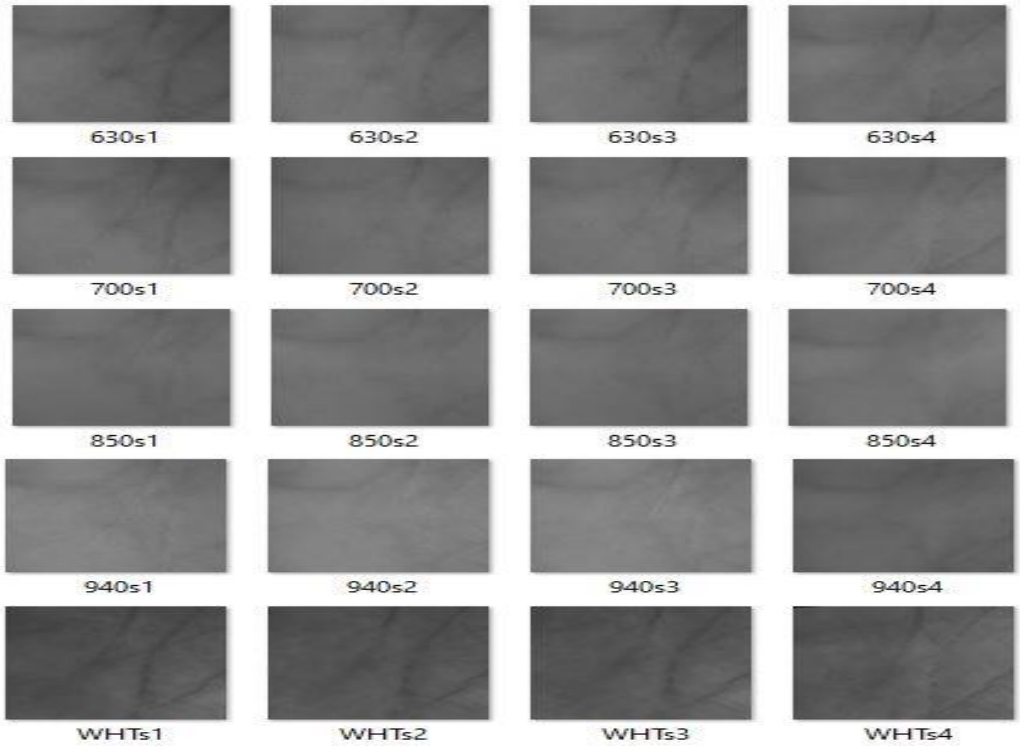


Fig IIII.4 – CASIA data set presentation in all wavelengths.

III.3. Software and libraries Used in the implementation

III.3.1. Python [8]

Python is a high-level, interpreted, object-oriented programming language with dynamic semantics. It is in high demand by a large community of developers and programmers. Python is a simple language, easy to learn and allows a good reduction in the cost of code maintenance. Python libraries encourage modularity and reusability of codes. Python and its libraries are available free of charge for the majority of platforms and can be redistributed free of charge.

III.3.2. TensorFlow [8]

TensorFlow is a programming framework for numerical computation which was made Open Source by Google in November 2015. Since its release, TensorFlow has been steadily gaining popularity, quickly becoming one of the most used frameworks for it. Deep Learning and therefore neural networks. Its name is inspired in particular by the fact that current operations on neural networks are mainly done via multi-dimensional data tables, called Tensors. A two-dimensional Tensor is the equivalent of a matrix. Today, the main Google products are based on TensorFlow: Gmail, Google Photos, Voice recognition.

III.3.3. Keras [8]

Keras is a high-level neural network API, written in Python and capable of running on TensorFlow or Theano. It was developed with an emphasis on rapid experimentation. Being able to get from idea to result with the least possible delay is the key to doing good research. It was developed as part of the research effort of the ONEIROS project (Open-ended Neuro-Electronic Intelligent Robot Operating System), in 2017, the Google TensorFlow team decided to support Keras in the main TensorFlow library. Cholet explained that Keras was designed as an interface rather than an end-to-end learning framework.

III.3.4. Scikit-learn [8]

Scikit-learn is a free Python library dedicated to machine learning. It is developed by many contributors, particularly in the academic world, by French higher education and research institutes such as Inria and Telecom ParisTech. It includes functions for estimating random forests, logistic regressions, classification algorithms, and support vector machines. It is designed to harmonize with other free Python libraries, notably NumPy and SciPy.

III.4. Problem studied and results Interpretation

We will focus on the problem of classification of images which consists in assigning to an input image x a label y from a fixed set of categories. This is one of the fundamental problems of computer vision which, despite its simplicity, has a wide variety of practical applications. Several bases have been presented in the literature; we have chosen the base of kaggle cats and dogs, weather, casia and PolyU multispectral palmprint.

III.4.1. Configuration Used in the implementation

The hardware configuration used in our implementation is:

- A Lenovo i3-4005U CPU 1.70 GHZ laptop.
- Intel Hd graphics card.
- 4 GB RAM.
- 500 GB hard drive.
- Windows 10 operating system 64-bit professional version.
- Python 3.6

III.4.2. Cats and Dogs classification results

In this classification, we used the kaggle cats and Dogs dataset that contain 2 categories, with 12500 samples for each one.

III.4.2.1. Network architecture

The first model that we present in table III.1 and fig III.5, is composed of two layers of convolution and two layers of maxpooling and two layers of fully connected. The input image is of size 50 * 50, the image passes to the first convolution layer. This layer is composed of 32 filters of size 3 * 3, Each of our convolution layers is followed by a Relu activation function this function forces the neurons to return positive values, and also Batch Normalization layer. After this convolution 32 features maps of size 32 * 32 will be created. Then we apply Maxpooling to reduce the size of the image and the number of parameters and calculation. The 32 feature maps which are obtained before they are given as input of the second convolution layer which is composed of 64 filters, an RELU activation function is applied on the convolution layer, then we apply Maxpooling to reduce the size of the image so the number of parameters and calculation. After these two convolutional layers, we use a neural network composed of two fully connected layers. The first layer contains 128 neurons.an RELU activation function is applied with dropout layer.

The output layer contains 1 neuron and with a sigmoid function which calculates the probability distribution of the 2 classes (number of classes in the database).

Layer (type)	*Output Shape*	*Param #*
conv2d_1 (Conv2D)	(None, 48, 48, 32)	320
activation_1 (Activation)	(None, 48, 48, 32)	0
batch_normalization_1	(Batch (None, 48, 48, 32)	128
max_pooling2d_1	(MaxPooling2 (None, 16, 16, 32)	0
conv2d_2 (Conv2D)	(None, 14, 14, 64)	18496
activation_2 (Activation)	(None, 18, 18, 64)	0
batch_normalization_2	(Batch (None, 14, 14, 64)	256
max_pooling2d_2	(MaxPooling2 (None, 7, 7, 64)	0
flatten_1 (Flatten)	(None, 3136)	0
dense_1 (Dense)	(None, 128)	401536
activation_3 (Activation)	(None, 128)	0
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 1)	129
activation_4 (Activation)	(None, 1)	0
Total params:		420,865
Trainable params:		420,673
Non-trainable params:		192
Total Epochs:		50

Tab III.1 – Cats and dogs network architecture parameters.

Fig III.5– cats and dogs network architecture layers.

III.4.2.2. Training details

We train the network on the popular deep learning platform Keras, all the biases and weights are initialized automatically, for the loss function we used the binary cross-entropy or the sigmoid cross entropy to calculate error, as for the optimizer we set a several optimizers one after one to find the best result after a few tests.

To start the training, we set the batch size of 50 samples. With a total of 50 epochs to complete the training process with a split of 10% for the validation.

III.4.2.3. Experiment results

In the experiment, we used 22500 grayscale images as training data and then used 2500 grayscale images as test data. After several rounds of training, the correct recognition rates and error obtained were 99.38%, 2.19%, respectively. We have also performed a comparative study with several optimizers, which are shown in table below. We can see that Adagrad optimizer exceeds the other optimizers in term of error and accuracy. Although the training time is almost the same and the same parameters used in all the tests. The following plots are the relation between accuracy, loss with number of epochs respectively.

Epochs	Optimizer	Time elapsed (s)	Loss (%)	Accuracy (%)
10	SGD	1530	42.75	80.25
	RMSprop	1580	33.67	85.84
	AdaGrad	1594	29.71	87.39
	Adadelta	1470	26.60	88.84
	Adam	1540	28.71	87.49
	Adamax	1584	29.72	87.10
30	SGD	3060	13.97	94.51
	RMSprop	3045	10.46	96.12
	AdaGrad	3100	4.79	98.56
	Adadelta	3162	6.33	98.00
	Adam	3095	7.03	97.30
	Adamax	3130	4.24	98.36
50	SGD	4600	4.59	98.85
	RMSprop	4650	6.85	97.66
	AdaGrad	4650	2.19	99.38
	Adadelta	4750	5.03	98.51
	Adam	4700	4.54	98.23
	Adamax	4720	2.35	99.16

Tab III.2 – Optimizers results comparison.

By varying the epoch parameter; fig. III.6 and fig.III.7 depict respectively, the plot of the obtained accuracy and loss for each optimized algorithm used in simulation.

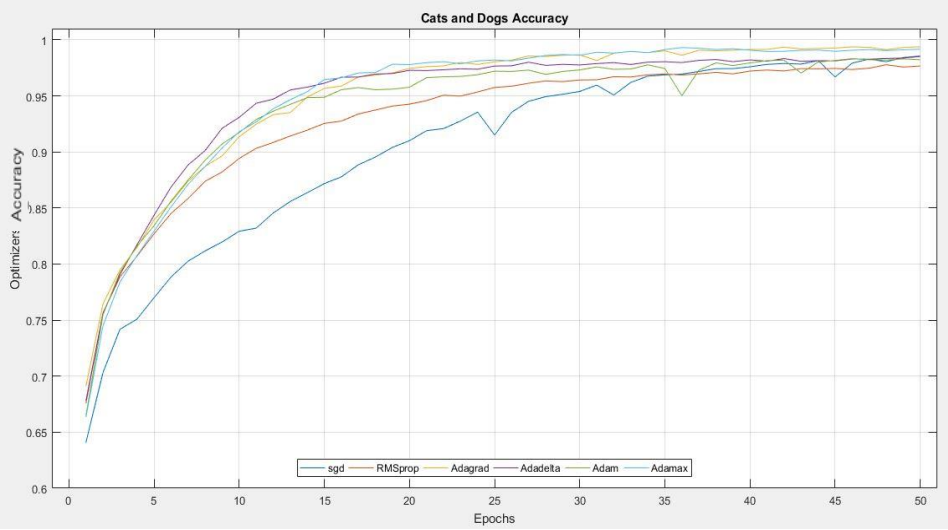


Fig III.6 – Cats and dogs Model accuracy.

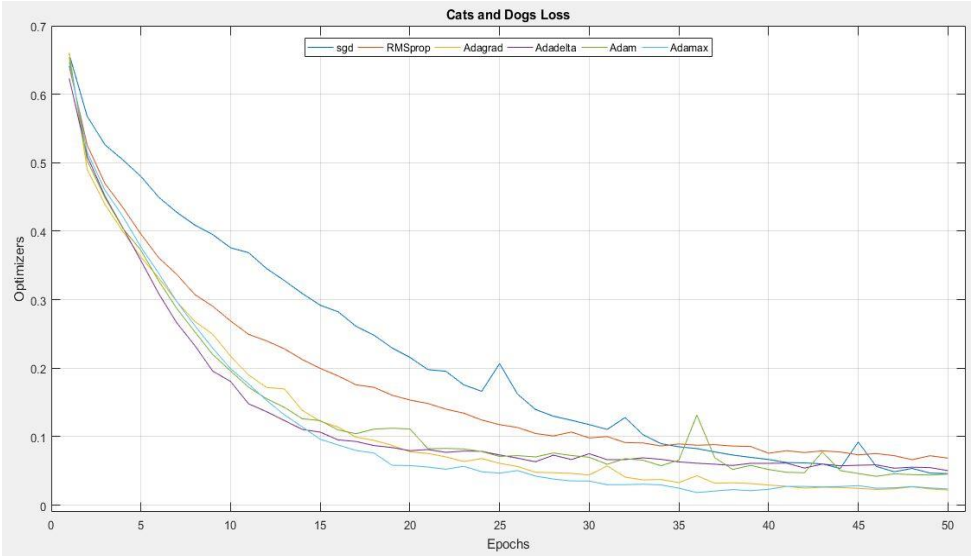


Fig III.7 – Cats and dogs Model loss.

As another performance measure we tested the model with 2000 samples unseen by our model before that makes us draw the confusion matrix as follow and calculate the model sensitivity and specificity beside the FAR and FRR curves.

Predict / Labels	Cats	Dogs
Cats	TP cats	FP cats
	978	22
Dogs	FN dogs	TN dogs
	31	969

Tab III.3 – Confusion matrix (Cats and dogs).

$$\text{Sensitivity} = \frac{978}{978+31} = 99.928\% \quad (3.1)$$

$$\text{Specificity} = \frac{969}{969+22} = 99.780\% \quad (3.2)$$

We have also calculated the FRR, FAR and then plot them by changing the value of threshold (see figure below). The error is obtained when FAR=FRR.

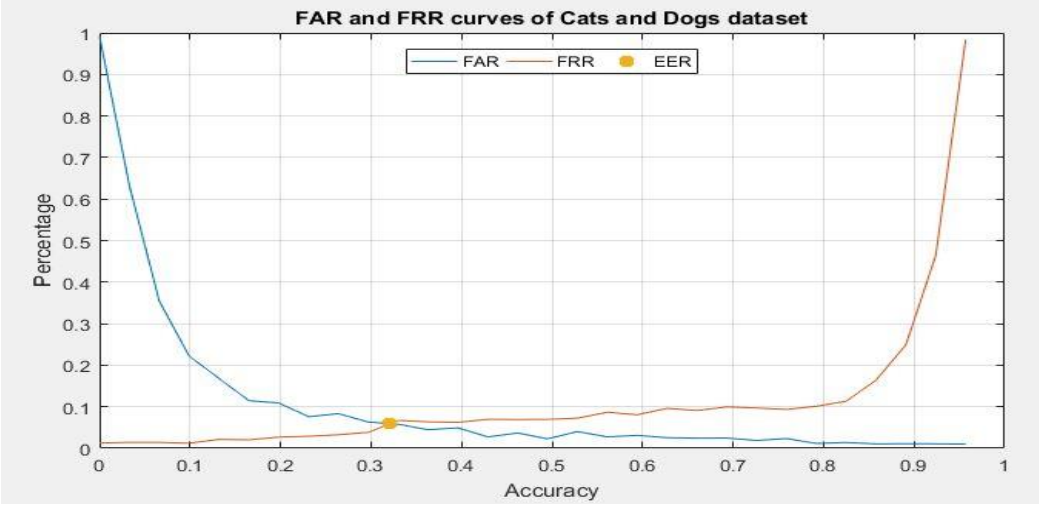


Fig III.8 – Cats and dogs Model FAR and FRR curves.

After training our model with different optimizers that shows a remarkable performance with being adagrad the best optimizer. we tested the model with an un seen data in the training process and it did very well in recognizing the two categories with a few errors related to the quality of the image and the object that it contain. But we can say that we managed to create a model with a good performance.

III.4.3. Weather classification results

In this classification I used the weather data set that contain 3 categories but we only used 2 of them that contains 215 images for each class.

III.4.3.1. Network architecture

For the network architecture there is no changes in it from the first model and we decided to use it again with this classification hopefully to get the results as shown in the previous classification. The table III.4 and fig. III.9 shows the parameters of the network architecture used in this section.

Layer (Type)	*Output Shape*	*Param #*
conv2d_1 (Conv2D)	**(None, 48, 48, 32)**	**320**
activation_1 (Activation)	**(None, 48, 48, 32)**	**0**
batch_normalization_1	**(Batch (None, 48, 48, 32)**	**128**
max_pooling2d_1	**(MaxPooling2 (None, 16, 16, 32)**	**0**
conv2d_2 (Conv2D)	**(None, 14, 14, 64)**	**18496**
activation_2 (Activation)	**(None, 18, 18, 64)**	**0**
batch_normalization_2	**(Batch (None, 14, 14, 64)**	**256**
max_pooling2d_2	**(MaxPooling2 (None, 7, 7, 64)**	**0**
flatten_1 (Flatten)	**(None, 3136)**	**0**
dense_1 (Dense)	**(None, 128)**	**401536**
activation_3 (Activation)	**(None, 128)**	**0**
dropout_1 (Dropout)	**(None, 128)**	**0**
dense_2 (Dense)	**(None, 1)**	**129**
activation_4 (Activation)	**(None, 1)**	**0**
Total params:		**420,865**
Trainable params:		**420,673**
Non-trainable params:		**192**
Total Epochs:		**50**

Tab III.4 – Weather network architecture parameters.

Fig III.9 – Weather network architecture layers.

III.4.3.2. Training details

Because the classification is a binary, as the same as the Cats and Dogs status, we used the same model with no changes so it will remain the training details as well. From the binary cross entropy is used as a loss function. The epochs, a slightly different in the batch size changes to 10 samples at once instead of 50 and an input of colored images by the size 64*64. The same test is used with all the optimizer's algorithms, to find an optimizer with relieving results.

III.4.3.3. Experiment results

After the end of the training; the model did well in recognizing the two categories. Adadelta optimizer scored an accuracy of 99.87% in a two categories classification with a loss of 0.57% (see table below). The model obtained is robust and reliable despite few errors occurred cause either to the quality of the image or to the training error. We have also performed a comparative study with several optimizers, which are shown in table (see tab III.5).

Epochs	Optimizer	Time elapsed (s)	Loss (%)	Accuracy (%)
10	SGD	1540	11.91	77.88
	RMSprop	1570	9.81	96.75
	AdaGrad	1590	12.03	96.37
	Adadelta	1440	6.31	98.64
	Adam	1550	8.32	95.27
	Adamax	1534	6.49	97.79
30	SGD	3070	2.66	97.70
	RMSprop	3055	4.34	98.57
	AdaGrad	3110	1.77	99.48
	Adadelta	3142	0.99	99.62
	Adam	3055	2.97	97.31
	Adamax	3180	1.05	99.53
50	SGD	4650	1.37	99.09
	RMSprop	4640	3.15	99.20
	AdaGrad	4680	0.96	99.55
	Adadelta	4740	0.57	99.87
	Adam	4710	3.57	97.74
	Adamax	4740	0.61	99.71

Tab III.5 – Weather network training results.

The confusion matrix obtained as follow and we calculate the model sensitivity and specificity beside the FAR and FRR curves (see table III.6 and fig. III.10).

Predict / Label	Rain	Shine
Rain	TP Rain	FP Rain
	64	1
Shine	FN Shine	TN Shine
	2	63

Tab III.6 – Confusion matrix (Weather 2 classes).

$$\text{Sensitivity} = \frac{64}{64+2} = 96.969\%. \quad (3.3)$$

$$\text{Specificity} = \frac{63}{63+1} = 98.438\%. \quad (3.4)$$

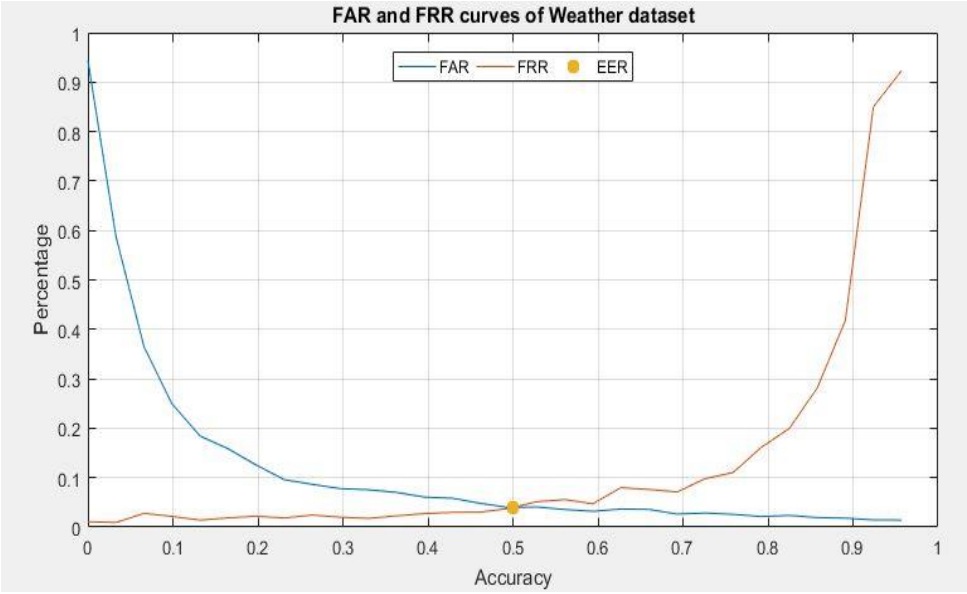


Fig III.10– Weather 2 classes Model FAR and FRR curves.

By varying the epoch parameter; fig. III.11 and fig.III.12 depict respectively, the plot of the obtained accuracy and loss for each optimized algorithm used in simulation.

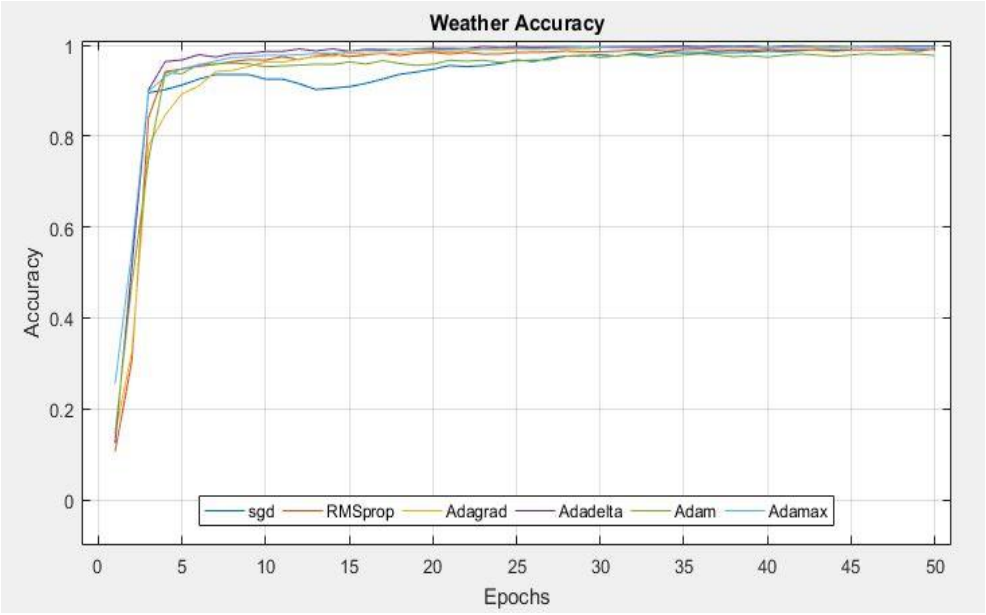


Fig III.11 – Weather (2 classes) Model accuracy.

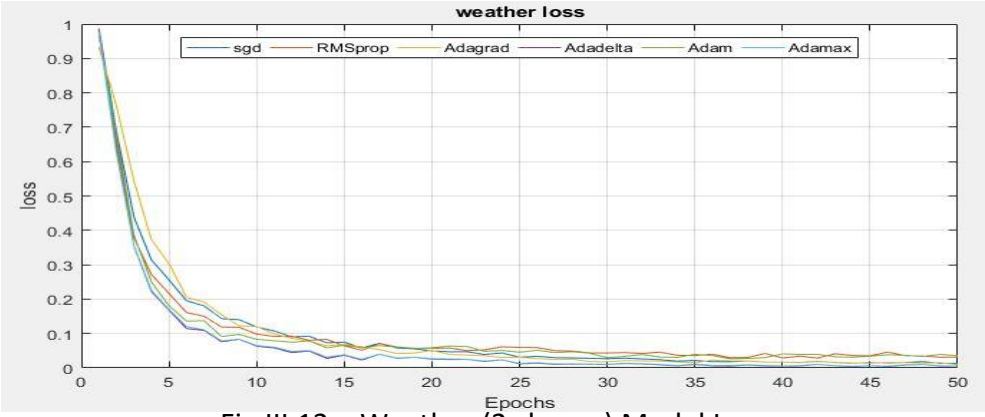


Fig III.12 – Weather (2 classes) Model Loss.

The results of the model have been very satisfying results even with two different datasets in samples number and image quality. The model itself have not seen any changes with the different dataset used, training and testing time is a variable change with the amount of data being used in the operation. Adagrad and adadelta optimizer has shown a very better accuracy and loss results in the two training processes. So, we can say with confidence after testing the model in extremely hard certain stances that it can solve any binary classification issue with promising results.

III.4.4. PolyU Multispectral palm print dataset classification

Biometrics is the technology widely used for identification and authentication for security purposes, the idea is to identifying persons by his/her physiological characteristics, such as iris, palmprint, fingerprint, and face, or using some aspect of his/her behavior, such as voice, signature, and gesture. Multispectral palmprint recognition and classification one of the best issues to apply deep learning CNN on. The best data type in extraction of features is the multispectral images by taking images in deferent wave lengths for the same palm for more features and textures. PolyU RED and NIR datasets are the base of our study in this part.

III.4.4.1. Network architecture

The model that we present in Tab III.7 is composed of four layers of convolution and three layers of fully connected. The input image is size 64* 64, the image goes first to the first convolution layer. This layer is composed of 32 filters of size 3 * 3, the activation function Relu is used, this activation function forces the neurons to return positive values, after this convolution 32 feature maps will be created of size 32 * 32.Then, the 32 feature maps that are obtained are given as input to the second convolution layer which is made up of 64 filters. The Relu activation

function is applied to this layer. Maxpooling is applied afterwards to reduce the size of the image. At the exit of this layer, we will have 32 feature maps of size 16 * 16. We repeat the same thing with convolution layers three and four which are composed of 128 filters, 256 filters respectively. The activation function Relu is always applied on each convolution. A layer of Maxpooling is applied after the both convolution layers. At the exit of this layer, we will have 64 feature maps of size 8 * 8. The vector of characteristics resulting from convolutions has a dimension of 4096.

After these four convolutional layers, we use a neural network composed of three fully connected layers. The first and second layers are composed of 1024 neurons where the activation function used is the Relu, the last layer uses the Softmax function which calculates the probability distribution of the 500 classes (number of classes in the PolyU image base). The network architecture is depicted on fig III.13.

Layer (type)	*Output Shape*	*Param #*
conv2d_1 (Conv2D)	(None, 62, 62, 32)	896
activation_1 (Activation)	(None, 62, 62, 32)	0
batch_normalization_1	(Batch (None, 62, 62, 32)	128
max_pooling2d_1	(MaxPooling2 (None, 20, 20, 32)	0
conv2d_2 (Conv2D)	(None, 18, 18, 64)	18496
activation_2 (Activation)	(None, 18, 18, 64)	0
batch_normalization_2	(Batch (None, 18, 18, 64)	256
max_pooling2d_2	(MaxPooling2 (None, 9, 9, 64)	0
conv2d_3 (Conv2D)	(None, 7, 7, 128)	73856
activation_3 (Activation)	(None, 7, 7, 128)	0
batch_normalization_3	(Batch (None, 7, 7, 128)	512
conv2d_20 (Conv2D)	(None, 5, 5, 256)	295168
activation_32 (Activation)	(None, 5, 5, 256)	0
batch_normalization_20	(Batch (None, 5, 5, 256)	1024
max_pooling2d_2	(MaxPooling2 (None, 9, 9, 64)	0
flatten_5 (Flatten)	(None, 1024)	0
dense_13 (Dense)	(None, 1024)	1049600
activation_33 (Activation)	(None, 1024)	0
dropout_9 (Dropout)	(None, 1024)	0
dense_14 (Dense)	(None, 1024)	1049600
activation_34 (Activation)	(None, 1024)	0
dropout_10 (Dropout)	(None, 1024)	0
dense_15 (Dense)	(None, 500)	512500
activation_35 (Activation)	(None, 500)	0
Total params:		3, 002,036
Trainable params:		3, 001,076
Non-trainable params:		960
Total Epochs:		200

Tab III.7 –Network architecture parameters used in PolyU Red

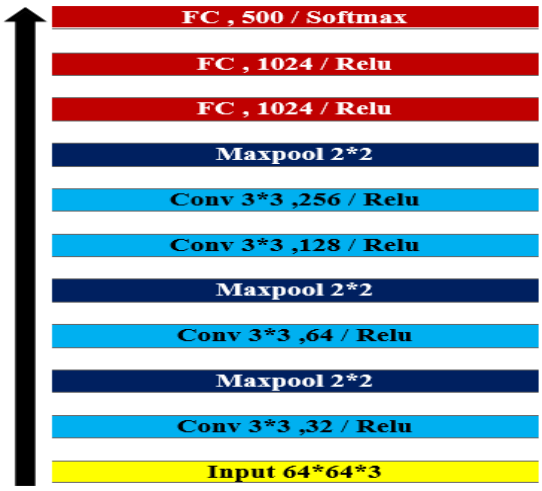


Fig III.13 –PolyU (RED and NIR) network architecture layers.

III.4.4.2. Training details

All the biases and weights are initialized automatically, for the loss function we used the categorical cross-entropy or the Softmax cross entropy to calculate error, as for the optimizer we set a several optimizers one after one to find the best result after a few tests. To start the training, we set the batch size of 64 samples. With a total of 200 epochs to complete the training process with a split of 10% for the validation.

III.4.4.3. Experiment results

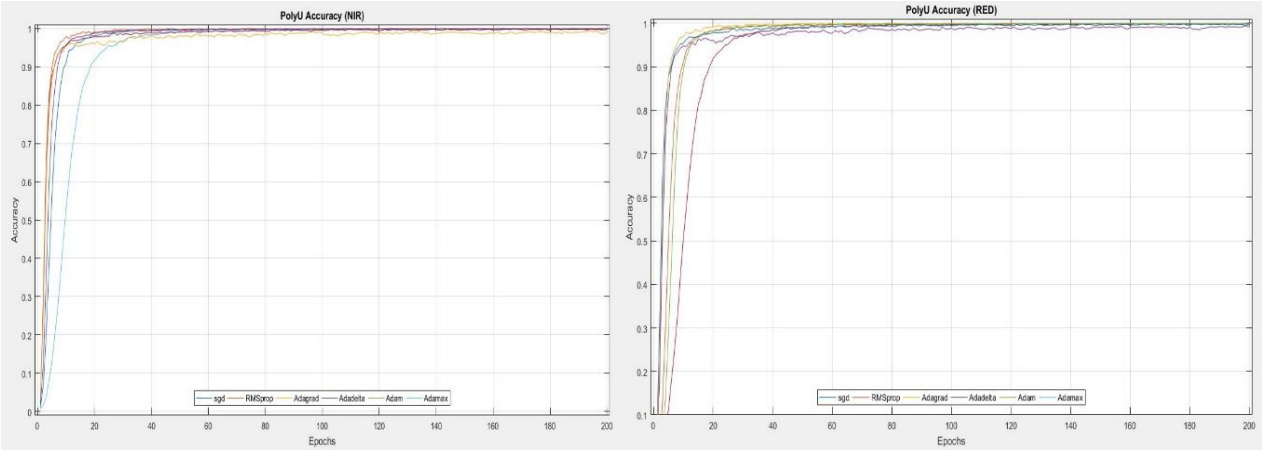
The obtained simulation results are given in the table III.8 and table III.9. Several optimization algorithms were used; we noticed that the optimizer Adadelta give for NIR et RED band respectively an accuracy of 99.98% and 99.94% with a loss of 0.03% and 0.11%. The following figures illustrate the accuracy graphs; the loss function and the FAR, FRR curve; these for both NIR and RED bands.

Epochs	Optimizer	Time elapsed (s)	Loss (%)	Accuracy (%)
10	SGD	470	222.95	52.77
	RMSprop	490	16.15	95.38
	AdaGrad	510	30.35	91.01
	Adadelta	620	7.60	97.92
	Adam	570	13.70	95.83
	Adamax	540	15.83	95.66
100	SGD	4700	1.75	99.70
	RMSprop	4900	3.08	99.59
	AdaGrad	5100	7.19	99.77
	Adadelta	6200	0.99	99.62
	Adam	5700	2.88	99.94
	Adamax	5400	12.12	98.68
200	SGD	9400	0.84	99.88
	RMSprop	9800	2.87	99.64
	AdaGrad	10200	0.34	99.90
	Adadelta	12400	0.03	99.98
	Adam	11400	15.70	98.83
	Adamax	10800	0.27	99.92

Tab III.8 – PolyU (NIR) training results.

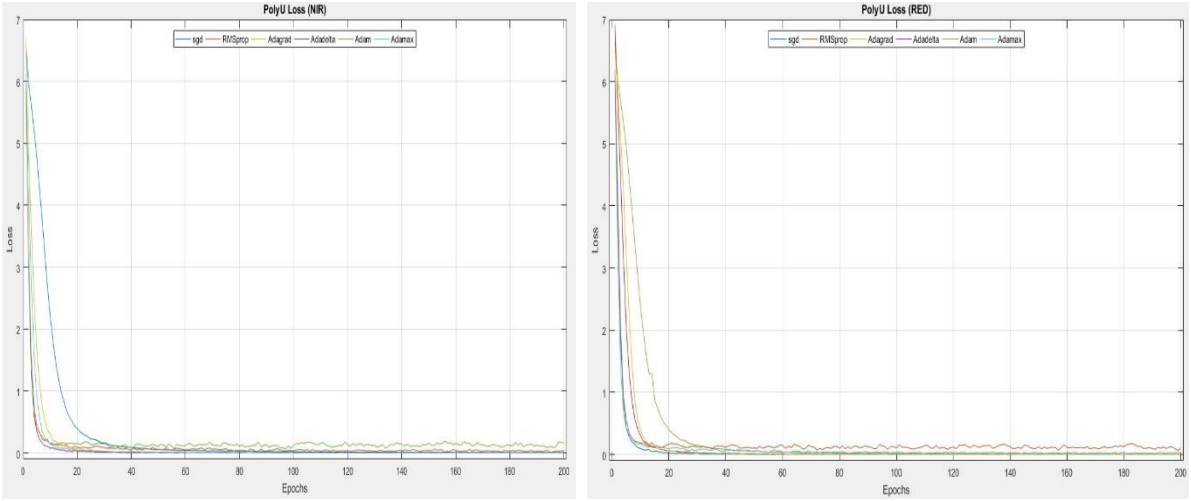
Epochs	Optimizer	Time elapsed (s)	Loss (%)	Accuracy (%)
10	SGD	437	238.22	50.51
	RMSprop	496	17.67	95.40
	AdaGrad	507	30.51	91.03
	Adadelta	650	9.78	96.90
	Adam	573	18.35	94.97
	Adamax	540	39.24	89.29
100	SGD	4700	1.57	99.64
	RMSprop	4900	3.02	99.53
	AdaGrad	5100	0.42	99.87
	Adadelta	6200	0.33	99.92
	Adam	5700	10.39	98.90
	Adamax	5400	0.42	99.90
200	SGD	9460	0.86	99.88
	RMSprop	9930	1.86	99.74
	AdaGrad	10140	0.24	99.90
	Adadelta	13000	0.11	99.94
	Adam	11460	10.72	99.14
	Adamax	10830	0.17	99.92

Tab III.9– PolyU (RED) training results.



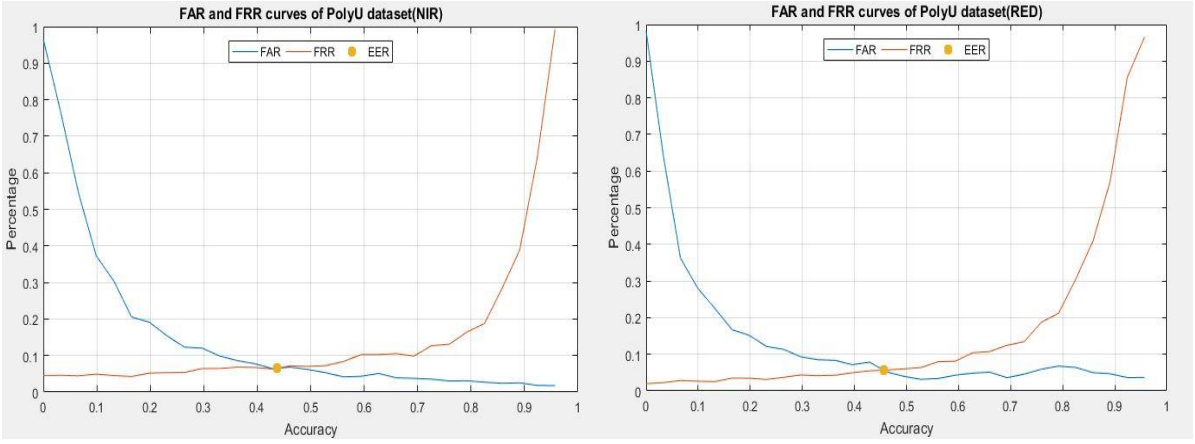
(a) (b)

Fig III.14 – (a) PolyU NIR Model accuracy, (b) PolyU RED Model accuracy.



(a) (b)

Fig III.15 – (a) PolyU NIR Model Loss, (b) PolyU RED Model Loss.



(a) (b)

Fig III.16 – (a) PolyU NIR Model FAR and FRR curves, (b) PolyU RED Model FAR and FRR curves...

The model has trained on the same data set with the same condition but the only difference is the wave length that images captured in which are RED and NIR. The results shown in previous figures shows that Adadelta was the best optimizer in the recognition problem with an accuracy of 99.94% for the RED data set and 99.98 % for the NIR data set.

Training time was not that big difference because it related to batch size, layers and neurons number but the quality of the image did change from one data to another. In the end our model has a great performance even with the few errors made because of the quality of the samples.

III.4.5. CASIA Multispectral palmprint dataset classification

The data set is rich with samples captures in a lots of wave lengths. In this experience we only use the 850 nm, 940 nm and the WHT (visible) waves, similar to the polyU dataset in the samples but a less categories because it contains only 60 categories.

III.4.5.1. Network architecture

CASIA dataset deep learning model is the same as the one used in polyU data set with all the layers and function. It witnesses a one change in the number of neurons in the output layer from 500 to only 60 neurons according to categories in the data set. This network used with the different wave length in this dataset with the same training details in PolyU dataset classification. The model is present in Tab III.10 and fig. III.17 Illustrate the network architecture.

Layer (type)	*Output Shape*	*Param #*
conv2d_1 (Conv2D)	(None, 62, 62, 32)	896
activation_1 (Activation)	(None, 62, 62, 32)	0
batch_normalization_1	(Batch (None, 62, 62, 32)	128
max_pooling2d_1	(MaxPooling2 (None, 20, 20, 32)	0
conv2d_2 (Conv2D)	(None, 18, 18, 64)	18496
activation_2 (Activation)	(None, 18, 18, 64)	0
batch_normalization_2	(Batch (None, 18, 18, 64)	256
max_pooling2d_2	(MaxPooling2 (None, 9, 9, 64)	0
conv2d_3 (Conv2D)	(None, 7, 7, 128)	73856
activation_3 (Activation)	(None, 7, 7, 128)	0
batch_normalization_3	(Batch (None, 7, 7, 128)	512
conv2d_20 (Conv2D)	(None, 5, 5, 256)	295168
activation_32 (Activation)	(None, 5, 5, 256)	0
batch_normalization_20	(Batch (None, 5, 5, 256)	1024
max_pooling2d_2	(MaxPooling2 (None, 9, 9, 64)	0
flatten_5 (Flatten)	(None, 1024)	0
dense_13 (Dense)	(None, 1024)	1049600
activation_33 (Activation)	(None, 1024)	0
dropout_9 (Dropout)	(None, 1024)	0
dense_14 (Dense)	(None, 1024)	1049600
activation_34 (Activation)	(None, 1024)	0
dropout_10 (Dropout)	(None, 1024)	0
dense_15 (Dense)	(None, 60)	61500
activation_35 (Activation)	(None, 60)	0
Total params:		2, 550,460
Trainable params:		2, 549,500
Non-trainable params:		960
Total Epochs:		200

Tab III.10 – CASIA (850nm, 940nm, WHT) network architecture.

Fig III.17 – CASIA (850nm, 940nm, WHT) network layers.

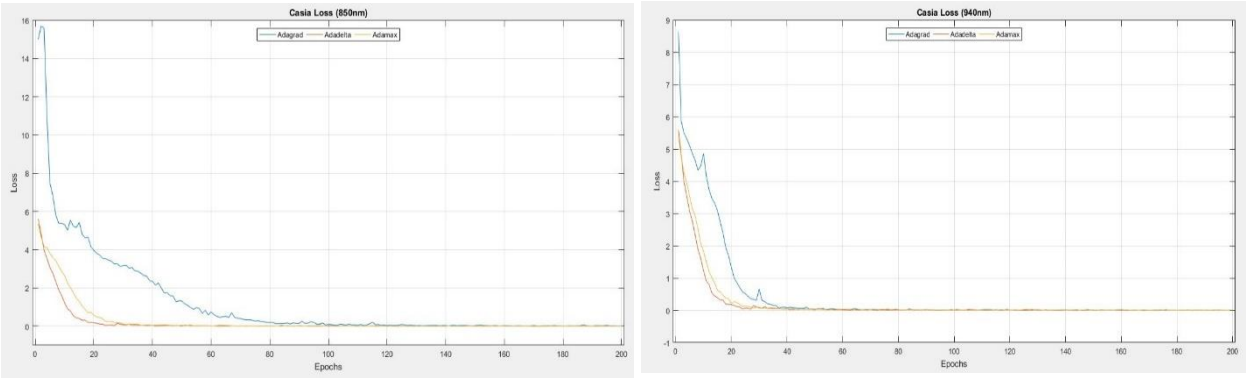
III.4.5.2. Experiment results

The obtained simulation results are given in the table III.11. Several optimization algorithms were used; we noticed that the optimizer Adadelta gives for the three bands a better accuracy in the interval between 99.85% and 99.97%. The following figures illustrate the accuracy graphs; the loss function and the FAR, FRR curve for the three bands.

The choice of those three specific optimizers is based on the last test. They did great in the classification of the polyU dataset. Adamax, adadelta and adagrad are the chosen optimizers for this test with the following accuracy and loss plots. Also, the test of the models was by choosing different samples for each dataset and trying to get a FAR and FRR graphs to see the performance of the models. The graphs are below:

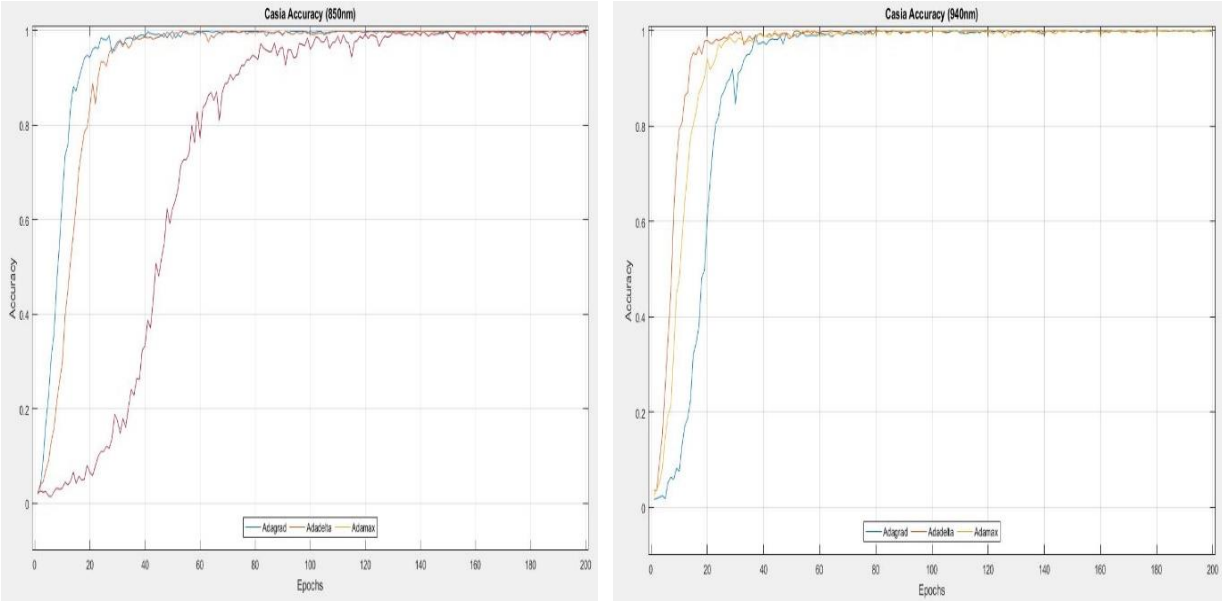
Dataset	Epochs	Optimizer	Time elapsed	Loss (%)	Accuracy (%)
850 nm	10	AdaGrad	140	530.9	3.24
		Adadelta	155	122.2	64.66
		Adamax	130	265.5	29.62
	100	AdaGrad	700	9.63	95.98
		Adadelta	715	0.05	99.85
		Adamax	680	2.81	99.22
	200	AdaGrad	1400	1.10	99.38
		Adadelta	1430	0.10	99.85
		Adamax	1360	0.12	99.73
940 nm	10	AdaGrad	140 s	485.20	7.56
		Adadelta	155 s	688.05	79.32
		Adamax	130 s	183.44	47.38
	100	AdaGrad	700 s	0.50	99.97
		Adadelta	715 s	0.55	99.84
		Adamax	680 s	0.37	99.92
	200	AdaGrad	1400 s	0.44	99.85
		Adadelta	1430 s	0.03	99.97
		Adamax	1360 s	0.17	99.92
WHT	10	AdaGrad	140 s	442.95	8.02
		Adadelta	155 s	136.51	81.17
		Adamax	130 s	630.25	59.56
	100	AdaGrad	700 s	2.80	99.53
		Adadelta	715 s	0.49	99.84
		Adamax	680 s	0.02	99.85
	200	AdaGrad	1400 s	1.82	99.53
		Adadelta	1430 s	0.09	99.85
		Adamax	1360 s	0.16	99.69

Tab iII.11– CASIA (850nm, 940nm, WHT) training results.



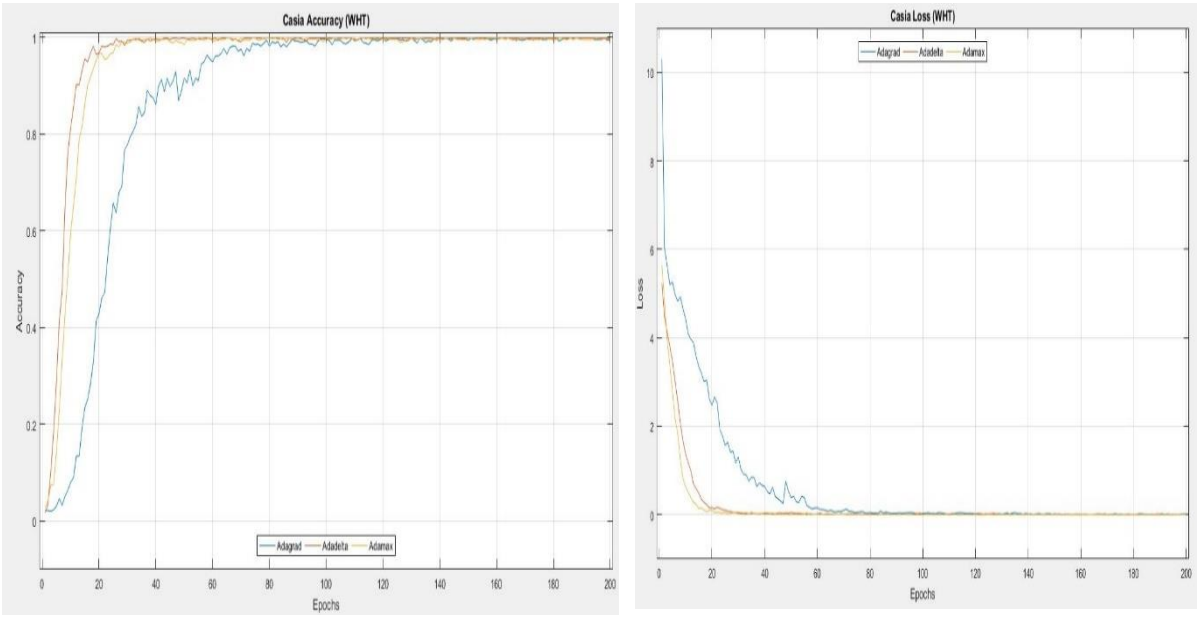
(a) (b)

Fig III.18 – (a) CASIA 850nm Model loss; (b) CASIA 940nm Model loss.



(a) (b)

Fig III.19 – (a) CASIA 850nm Model Accuracy; (b) CASIA 940nm Model Accuracy.



(a) (b)

Fig III.20 – (a) CASIA WHT Model accuracy; (b) CASIA WHT Model loss.

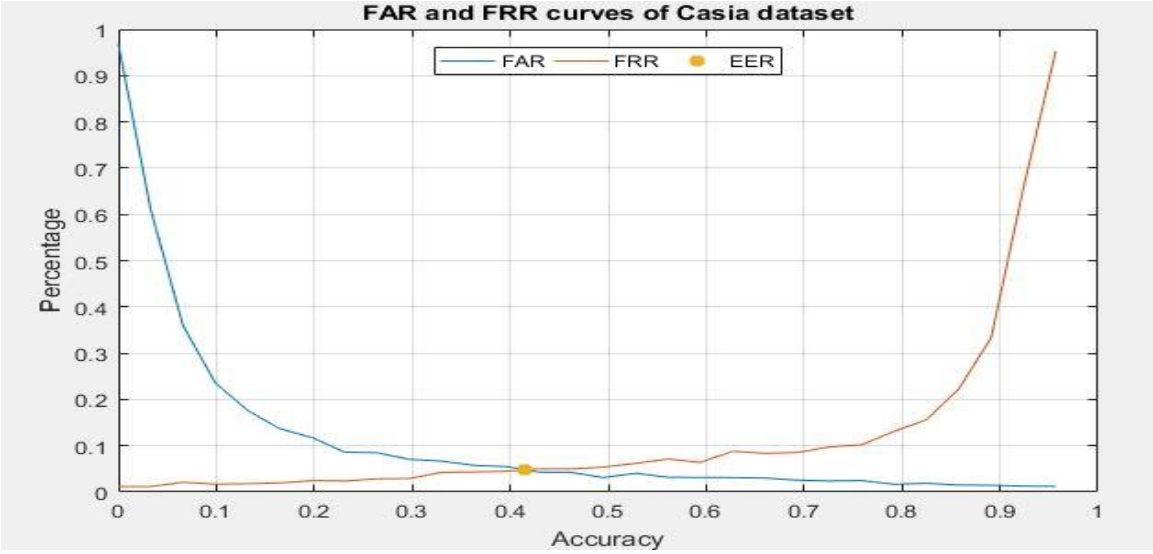


Fig III.21 – CASIA FAR and FRR plots.

The model as we tested it in polyU data set did a very awesome performance, as expected even with the change of the data to casia dataset and only with three types of optimizers. Adadelta optimizer was the optimizer who performed a very excellent recognition with high performance and it experienced three types of data. Casia 850nm images had a 99.85% of accuracy and a loss of 0.1%. As well as the WHT samples only with a loss of 0.09, our model was better in the 940nm samples because it scored a 99.97% of recognition accuracy. The loss of this data was almost perfect a percentage of 0.03% to be exact. If this means something it will be that the model is capable of handling palm print recognition problems very well even with the different data quality.

III.5. Results of multispectral image classification using LBP and K- Nearest Neighbor classifier

In this section, we will present the results obtained by applying the LBP-based approach for the extraction of image features. The K-nearest neighbor (k-NN) [25] is one of the best known algorithms in supervised classification. We will use it to measure the performance of the proposed method. To evaluate the performance of this classifier, we use a procedure called cross-validation. The latter consists in dividing the data into several partitions. K-NN is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). For the classification of a new observation x, the idea is to make vote the nearest neighbors of this observation. The class of x is determined as a function of the majority class among the k nearest neighbors of observation x .

III.5.1. LBP feature extraction results

Figure II.2 shows an example of a calculation of classical LBP operator performed on the multispectral image of the palmprint. The performance of the LBP operator depends on the choice of the parameters *P* and *R* as well as the total number of pixels comprising the image.

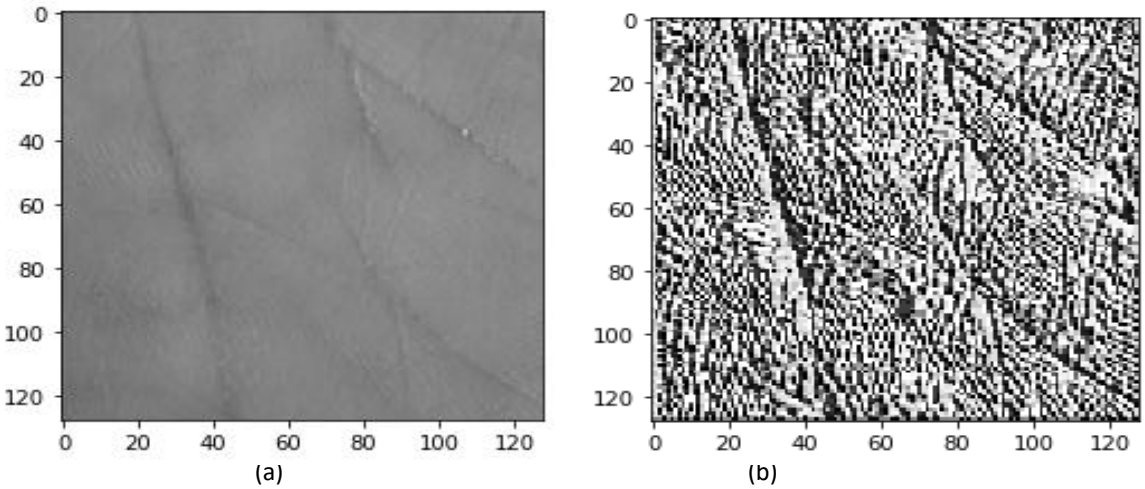


Fig III.22 – (a) Sample from polyU red dataset; (b) PolyU red dataset sample after feature extraction with LBP.

Cross-validation [26] is when the dataset is randomly split up into 'k' groups. One of the groups is used as the test set and the rest are used as the training set. The model is trained on the training set and scored on the test set. Then the process is repeated until each unique group has been used as the test set. For example, for 5-fold cross validation, the dataset would be split into 5 groups, and the model would be trained and tested 5 separate times so each group would get a chance to be the test set. This can be seen (see Fig III.23).

Fig III.23 – Cross-validation data split example.

III.5.2. Classification using k-NN

In this section, we study the influence of the k parameter of the k-NN classifier on the classification performance. We therefore varied the parameter k from 1 to 9. In the algorithm, we used the necessary libraries and we also divided the datasets into two parts represented at 80% for training and 20% for testing. In the cross-validation approach, we used five records. The table below displays the overall accuracy for several k parameters.

K values / Data set	1	3	5	7	9
	Accuracy (%)				
CASIA 850nm	68.88	48.33	29.44	17.22	8.33
CASIA940nm	55	39.44	23.88	11.11	5
CASIA WHT	92.22	75.55	60	45	31.66
PolyU NIR	90.06	76.26	60.06	45.46	30.8
PolyU RED	90.2	78.33	65.66	51.46	37.46

Tab III.12– Classification rates for test data using different parameters k and k-NN classifier.

The results of the k-NN classifier was promising, it scored a 92% of accuracy on the white CASIA band dataset and 90.2% on PolyU.

III.6. Conclusion

Classification problem is one of the best fields to apply deep learning in it. Distinguishing two categories seems a small problem but difficult to handle especially when the number of samples in each category is too large it may create a problem in time and effort. Our model did a great job in the binary classification problems because it recognizes samples in the right categories no matter how much samples in each category. In biometry or in multispectral palmprint classification, it did great in the process with the minimum errors can happen. The model layers or neurons number can be small and have a good performance if we just chose the right optimizer and parameters according to the specific problem. In our study, we have implemented the binary and multiclass classification of images using the CNN algorithm

General Conclusion

General conclusion

Deep convolutional neural networks have become the leading methods for image classification tasks. A major advance in the construction of models for image classification was the discovery that a convolutional neural network could be used to gradually extract ever higher-level representations of image content. Instead of preprocessing the data to obtain derived characteristics such as texture and shape, a convolutional neural network only uses the raw pixel data from the image.

In this project, we have discussed the fundamentals of deep learning, the most popular algorithms. Neural networks in general and convolutional neural networks in particular. We have introduced these convolutional neural networks by presenting the different types of layers used in the classification (convolutional layer, correction layer, pooling layer and fully connected layer).

We implemented two models of convolutional neural networks with different architectures; the first which has 2 convolution layers and the second has 4 convolution layers and subsequently, we applied for each of these models tests which consist in changing the optimization function each time, calculating the execution time for each of the tests and display at the end the confusion matrix to retrieve the result of the well and badly classified images. And the graphs of FAR and FRR.

The implementation was done with the python programming language and we used libraries to facilitate the task of creating our models and for the acceleration of training and finally we ended with a summary and comparative table of optimizers.

Finally, to validate the performance and robustness of the image classification approach based on the CNN algorithm; we have compared this approach with a classical classification technique using the LBP descriptors for the feature extraction of the texture associated with the K-NN classifier. The obtained results show the efficiency of the CNN algorithm.

Bibliography

[1] T. Kumar, K. Verma, "A Theory Based on Conversion of RGB image to Gray Image", International Journal of Computer Applications, Vol. 7, N° 2, pp. 0975 – 8887, 2010.

[2] M.S. Nixon, A. S. Aguado, "Feature Extraction and Image Processing », book, Newnes; Edition British Library; 2002.

[3] J. Shen, T. Zhang, "Multispectral Image Processing and Pattern Recognition", Society of Photo-optical Instrumentation Engineers World Scientific, 2001.

[4] M. Kunaver, J. F. Tasič "Image feature extraction – an overview", EUROCON 2005 Serbia & Montenegro, pp. 22-24, 2005.

[5] M. Lorentzon "Feature extraction for image selection using machine learning", Master of Science Thesis in Electrical Engineering, 2017

[6] M. Hassaballah, Abdelmgeid A. Ali, Hammam Alshazly, "Image Features Detection, Description and Matching", Studies in Computational Intelligence 630, DOI 10.1007/978-3-319-28854-3_2, 2016.

[7] M. S. Nixon and A. S. Aguado, Feature Extraction & Image Processing for Computer Vision, Book 3rd Edition, Academic Press , 2013

[8] Mokri Mohammed Zakaria. "Classification des images avec les réseaux de neurones convolutionnels" mémoire de fin d'étude Master en informatique, Université Abou Bakr Belkaid Tlemcen 2017.

[9] S S. Sawakare and D. Chaudhari, "Classification of Brain Tumor Using Discrete Wavelet Transform, Principal Component Analysis and Probabilistic Neural Network", International journal for Resea. Emer. and Techn., Vol.1 (6), pp.13-19, 2014

[10] Boughaba M. et B. Boukhris, "L`apprentissage profond (Deep Learning) pour la classification et la recherché d'images par le contenu", mémoire de fin d'étude Master Informatique et Technologie de l'Information, Université de Ouargla, Algérie, 2017.

[11] M. Mohammed, M. Badruddin Khan, E. B. M. Bashier . "Machine Learning: Algorithms and Applications ", International Standard Book , Publisher: CRC Press, 2016.

[12] B. Meddah, "Classification des images selon la sémantique". Mémoire de fin d'étude Master en Informatique, Université de Mostaganem, Algérie, 2016.

[13] A. Khan, A. Sohail, U. Zahoora and A. S. Qureshi," A Survey of the Recent Architectures of Deep Convolutional Neural Networks", Published in Artificial Intelligence Review, 21 April 2020.

[14] X. Dong, J. Wu, L. Zhou "How deep learning works — the geometry of deep learning" Faculty of Computer Science and Engineering, Southeast University, Nanjing, China arXiv: 1710.10784v1 [cs.LG] 30 Oct 2017.

[15] S. ALBAWI, T. Abed MOHAMMED. S. AL-ZAWI." Understanding of a Convolutional Neural Network", Antalya, Turkey August 2017 DOI:10.1109/ICEngTechnol.2017.8308186

[16] C. E. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning". https://arxiv.org/abs/1811.03378.

[17] P. Li "Optimization Algorithms for Deep Learning" Department of Systems Engineering and Engineering Management,The Chinese University of Hong Kong.

[18] F. LAMARE. "OCT en phase pour la reconnaissance biométrique par empreintes digitales et sa sécurisation". Doctoral School: Computer Science, Telecommunications and Electronics of Paris March 21, 2016.

[19] M. Hossin, and M.N. Sulaiman, "A Review on evaluation metrics for data classification evaluations ", International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol.5, No.2, 2015

[20] J. Brownlee, "Metrics To Evaluate Machine Learning Algorithms in Python"; in Python Machine Learning, 2016.

[21] https://www.microsoft.com/en-us/download/confirmation.aspx?id=54765

[22] https://data.mendeley.com/datasets/4drtyfjtfy/1#:~:text=Multi%2Dclass%20weather%20ataset(MWD,image%20using%20heterogeneous%20ensemble%20method%E2%80%9D.
[23] https://www4.comp.polyu.edu.hk/~biometrics/MultispectralPalmprint/MSP.htm
[24] http://biometrics.idealtest.org/dbDetailForUser.do?id=5
[25] P. Cunningham and S. Jane Delany. “K-Nearest neighbor classifiers”, Technical Report UCD-CSI-2007-4 March 27, 2007.
[26] S. Varma and R. Simon, “Bias in error estimation when using cross-validation for model selection”, BMC Bioinformatics vol. 7, N° 91, 2006.