PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
MOHAMED BOUDIAF UNIVERSITY - M'SILA

FACULTY OF MATHEMATICS
AND COMPUTER SCIENCE
COMPUTER SCIENCE
DEPARTMENT
N° :……..

Domain: Mathematics and
Computer Science
Branch: Computer Science
Specialty: AI

A DISSERTATION
SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE
OF MASTER IN COMPUTER SCIENCE

By: Dilmi Anes / Larbaoui Ayoub Fateh Allah

TOPIC

HANDWRITTEN DIGITS RECOGNITION

The jury composed of:

Dr. Heraguemi Kamel Eddine	University of M'sila	President
Dr. Bentrcia Rahima	University of M'sila	Supervisor
Dr. Kamel Mohamed	University of M'sila	Examiner

Academic Year: 2021 / 2022

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
MOHAMED BOUDIAF UNIVERSITY - M'SILA

FACULTY OF MATHEMATICS
AND COMPUTER SCIENCE
COMPUTER SCIENCE
DEPARTMENT
N° :……..

Domain: Mathematics and
Computer Science
Branch: Computer Science
Specialty: AI

A DISSERTATION
SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE
OF MASTER IN COMPUTER SCIENCE

By: Dilmi Anes / Larbaoui Ayoub Fateh Allah

TOPIC

HANDWRITTEN DIGITS RECOGNITION

The jury composed of:

Dr. Heraguemi Kamel Eddine	University of M'sila	President
Dr. Bentrcia Rahima	University of M'sila	Supervisor
Dr. Kamel Mohamed	University of M'sila	Examiner

Academic Year: 2021 / 2022

ACKNOWLEDGEMENTS

In the name of God, the most merciful,

the most merciful and his prophet Muhammad.

« صلى الله عليه و سلم »

First of all, we thank God for providing everything this project required.

We have put effort into this project. However, this would not have been possible without the kind support and help from many individuals. We would like to extend our sincere thanks to all of them.

We would like to express our deep appreciation to our Final Year Project Director **Ms. Bentrcia Rahima** for this consistent guidance and supervision as well as for providing the necessary information in relation to the Automated Handwriting Recognition project and also for this support in completing this project. We thank all members of the jury for the honor they have given us by agreeing to screen this job. Also, to our teachers who have contributed to our training and everyone who has helped us from near or far to realize this memory.

We would like to express our gratitude to our parents for their cooperation and encouragement.

Special thanks go to our friends and families who have contained the hectic moments and stress we have been through during the course of the research project.

We thank the school for giving us the grand opportunity to work as a team which has indeed promoted our team work spirit and communication skills.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND SYMBOLS

HDR	Handwritten Digits Recognition
CNN	Convolution Neural Network
FC-ANN	Fully Connected Artificial Neural Network
AI	Artificial Intelligence
MNIST	Modified National Institute of Standards and Technology
SVM	Support Vector Machines
DBN	Deep Belief Networks
HOG	Histogram of Oriented Gradients
MLP	Multi-Layer Perceptron
DNN	Deep Neural Network
KNN	K-Nearest Neighbors
ANN	Artificial Neural Network
RGB	Red Green Blue
MRF	Markov Random Fields
ReLU	Rectified Linear Unit
NN	Neural Network
CNN-RNN	Convolution Neural Network Recurrent Neural Networks
CNN-HMM	Convolution Neural Network Hidden Markov Model

LIST OF EQUATIONS

GENERAL INTRODUCTION

As science has evolved over time, man has been able to create machines that compensate for the limitedness of his thought and his physical incapacity and integrate them into his daily life to alleviate difficulties. One of the most important of these is a computer which makes it easier for man to do long and accurate calculations and do things in a few minutes or hours, compared to the fact that it takes years if man does it himself. The latter did not stand here, but came to make the computer mimic the human intelligence which is what is known as Artificial Intelligence, for example, issued decisions based on what it was trained on independently, deduced the features of things automatically, or Computer Vision that enables computers and systems to derive meaningful information from digital images, videos, and other visual inputs. In addition, there are several applications [1], namely: fingerprint recognition, identification of people, recognition of defects in parts (quality control), recognition of handwritten characters, etc. In this last application, we distinguish the recognition of handwritten digits, which will be the subject of our work.

For a long time, writing has been considered a privileged mode of communication between individuals. It is also a means of disseminating and preserving knowledge. Today, although the printing press and then the computer have made it possible to automate the writing process, handwriting remains extremely present in our world. Indeed, a mass of handwritten documents continues to grow every day, and more and more industries and services have a need to use fast processing techniques while ensuring the security of these documents.

In the next days, a handwritten recognition system might serve as a cornerstone to initiate paperless surroundings by digitizing and processing existing paper documents.

1 Problem Statement

In this thesis, we are particularly interested in the recognition of isolated handwritten digits. In general, two types of handwriting recognition systems are known: offline recognition and online recognition. The first works on images scanned using a scanner or camera. The second takes the data directly from an electronic pen or a digitizing tablet and transforms it into a digitized text while writing in real-time [2]. In this work, we are interested in the first category which is a task more complex than the other one where the number of natural classes is reduced to ten classes (numbers ranging from 0 to 9). The difficulty in recognizing this category is due to a large number of different fonts and differences from person to person. The writing style may also differ according to the situation and the pen or paper used.

Depending on the writing style and speed of the people, the digits can be diverse in terms of orientation, size, and distance from the margins, thickness, texture, background, and stress on some parts of numbers, etc. The human visual system is not affected by these differences in the letters, but in an automatic system, these pose big problems. As an example, we cite postal sorting which is one of the first applications where every day thousands of envelopes are automatically sorted [3,4]. Like this application, there is a distinction between reading the digital amount of bank checks and identifying the social security number. However, despite the impressive progress of the techniques used as well as the explosion in the computing power of computers, research on Handwriting Digits Recognition (HDR) is advancing with a recognition performance that remains far from that of the human eye. Among the similar applications, we mention the taxation process of bank checks, reading forms, and document indexing based on dates (e.g., document date, birthdate, marriage date, and death date), etc.

2 Novel Contributions

The main task in offline handwriting recognition is to transform the written text into a representation that can be understood by a machine and easily reproduced by a word processor. The following are the processing steps: initially, a dataset is given as input. This is followed by preprocessing, where an image is subjected to various operations like noise reduction, digit skew correction, slant correction, normalization, smoothing, and skeletonization [5]. The result of this phase can be given as an input to the feature generation phase.

Basically, this thesis includes the use of Convolution neural network (CNN) approach, which is part of deep learning that is mostly utilized to group images, cluster them by similarity, and perform picture acknowledgment within the scenes, it extracts more diverse features from each handwritten digital image and different features sets are prepared through filter-based feature selection as well as training an Fully Connected Artificial Neural Network FC-ANN classifier on different feature sets. The accuracy of the final model is important as more accurate models make better decisions.

The models with low accuracy are not suitable for real-world applications. For example, as we mentioned earlier, the automated bank check processing system [6,7], where the system recognizes the amount and date and bank account number on the check, if the system incorrectly recognizes a digit, it can lead to major damage which is not desirable. That's why an algorithm with high accuracy is required in these real-world applications.

3 Thesis Organization

Our thesis is structured as six parts:

In the first, we have introduced the main theme of the thesis with motivations and contributions to the field of study.

Chapter 1 describes the dataset used in this research.

Chapter 2 reports an overall review of the literature about handwritten digits recognition and the techniques used in all phases of system development.

Chapter 3 presents the architecture of the proposed system and explains in details each phase of its implementation starting from the preprocessing phase and ending by the recognition phase.

Chapter 4 reports the experimental results and analysis, and discusses the achieved accuracy rate.

Finally, we provide the conclusion and future work.

CHAPTER 1

DATASET

The dataset is a collection of related sets of information that is composed of separate elements but can be manipulated as a unit by a computer. It can be video files, images, texts, sounds, or even statistics in the form of an information matrix usually used to train models. However, when it comes to building practical AI systems, the data set on which it is trained is just as important as the accuracy of the choice of algorithm.

As part of this project, we have used a subset of the handwritten digits taken from the available datasets of the Modified National Institute of Standards and Technology (MNIST). It was extensively studied by the machine learning community for more than two decades to train and test different pattern recognition methods (models). Basically, it was created in 1998 as a combination of two National Institute of Standards and Technology (NIST) datasets written by high school students and employees of the US Census Bureau [8]. MNIST is a collection of 70000 digits images written by 250 individuals in their own handwriting. This collection contains 60000 samples (Table 1.1) for training and 10000 samples for testing [9], with each sample bearing a label identifying the class they fall under (Figure 1.1). All digits' images are size normalized to fit in a 28x28 pixel box while preserving their aspect ratio before they go through the image preprocessing phase. Finally, the output images contain gray levels as a result of the anti-aliasing technique used by the normalization algorithm. See Figure 1.2.

MNIST dataset is very useful in such fields of research because it is free [10] and the images are already preprocessed. Moreover, it is the largest dataset available on the web for handwritten digits.

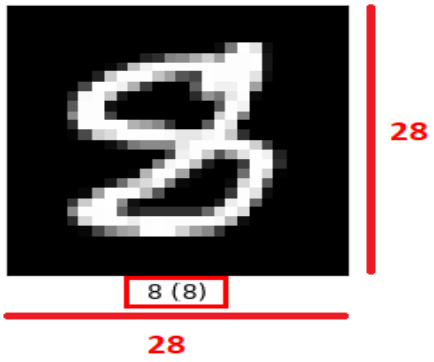


Figure 1.1 **: 8 labeled 28*28 Grayscale color digit**

Figure 1.2 **: Example of the MNIST database**

In our work, we used MNIST digits images to train and test the proposed recognition system. Table 2.0 describes how data is divided into a training sample which includes 60000 images, a testing sample of 10000 images, and 10 output classes represent the handwritten digits.

Class	Number of samples		
	Train Data	Test Data	Total
'0'	5923	980	6903
'1'	6742	1135	7877
'2'	5958	1032	6990
'3'	6131	1010	7141
'4'	5842	982	6824
'5'	5421	892	6313
'6'	5918	958	6876
'7'	6265	1028	7293
'8'	5851	974	6825
'9'	5949	1009	6958
All	60,000	10,000	70,000

Table 1.1: MNIST Handwritten Digits used in training and testing

CHAPTER 2

LITERATURE REVIEW

Handwritten digits recognition has become a challenging issue among specialists. There is a large number of studies and researches published these days about it.

In 1995, Support Vector Machines (SVM) have been used for the first time in handwritten digit recognition systems [11]. Convolutional neural networks have also been used to recognize handwritten numbers, especially on MNIST data. Most models have achieved high recognition accuracy between 98% and 99% [12]. They are attempting to expand the precision with fewer mistakes in CNN. The time factor is also considered for training the system, these works that we will mention are all applied on the MNIST dataset.

The Deep Belief Networks (DBN) with three layers along with a greedy algorithm were investigated for the MNIST dataset and reported an accuracy of 98.75% and error rate of 2.49% [13]. Also, Fatma Siddique et al. In [14] used 7 layered CNN model with 5 hidden layers along with gradient descent and back prorogation model to find and compare the accuracy on different epochs, thereby getting maximum accuracy of 99.2% with (15 epochs), and an error rate of 2.63%.

In [15], R. Ebrahimzadeh and M. Jampour proposed an appearance feature-based approach which processes data using Histogram of Oriented Gradients (HOG), and linear SVM has been employed as a classifier. The accuracy rate reaches 97.25%. Moreover, R. Dixit et al. compared the accuracy of three models on the MNIST dataset: Support Vector Machines (SVM), Multi-Layer Perceptron (MLP), and Convolution Neural Network (CNN) models [16]. They concluded that the best model is CNN in terms of training time, accuracy which is 99.53%, and the error rate of 4%.

Younis and Alkhateeb [17] provided the implementation of Deep Neural Network (DNN) models on MNIST dataset for solving handwritten OCR problem. Models are capable of extracting important features without preprocessing with an accuracy of 98.46%. On the other side, Gupta et al. in [18] proposed a novel multi-objective optimization framework for identifying the most informative local regions from a character image. The work was also evaluated on isolated handwritten English numerals MNIST dataset, along with three other popular Indic scripts, namely, handwritten Bangla numerals and handwritten Devanagari characters. The authors used features extracted from a convolutional neural network in their model and achieved 98.92% recognition accuracy.

A context-dependent Deep Neural Network and Hidden Markov Model for large vocabulary speech recognition is proposed in [19]. The system is tested for both MNIST and TIMIT database. By using MNIST, 0.83% error rate is calculated. The high recognition accuracy of 99.73% on the MNIST dataset is achieved while experimenting with the famous committee technique of combining multiple CNNs in an ensemble network from 7-net [20]. The work was further extended into 35-net where the accuracy increased to 99.77%, with an error rate of 0.39% [21]. Also, Niu and Suen integrated the CNN and SVM for MNIST digit database and reported a recognition rate of 99.81% which is the highest training accuracy for a hybrid model [22].

Ghosh and Maghari [23] conducted a comparative study on three neural network approaches demonstrating that DNN was the best algorithm with 98.08% accuracy. However, every neural network has some error rate due to similarity in digit shape. In addition, Hamid et al. [24] used three different classifiers, namely KNN, SVM, and CNN, to assess the performance on MNIST datasets. The performance of Multilayer perceptron on that platform was not up to mark as it wasn't able to accurately recognize digits 6 and 9, and stuck in the local optimal rather than global minimum. With the implementation of Keras modality, it was reported that accuracy was improved on CNN as other classifiers, performed accurately 99.26%.

A different work applied three classifiers SVM, ANN, and CNN to recognize digits with noise. It demonstrated that SVM, ANN, and CNN systems can achieve high accuracy in the recognition of handwritten digits on documented images [25].

In this dissertation, we search for the best algorithms in terms of highest recognition accuracy, lowest error rate, and fast execution, using MNIST data of handwritten digits.

CHAPTER 3

THE PROPOSED RECOGNITION SYSTEM

In this research, we developed a system for handwritten digits recognition which processes data using Convolutional Neural Networks. We seek an efficient and effective model in terms of recognition accuracy, computational time, and high efficiency for feature extraction. However, these depend directly on the accuracy of the enhanced images, and if the variations of data could be kept minimum. The next section introduces the first phase in the development of the handwritten digits' recognition model.

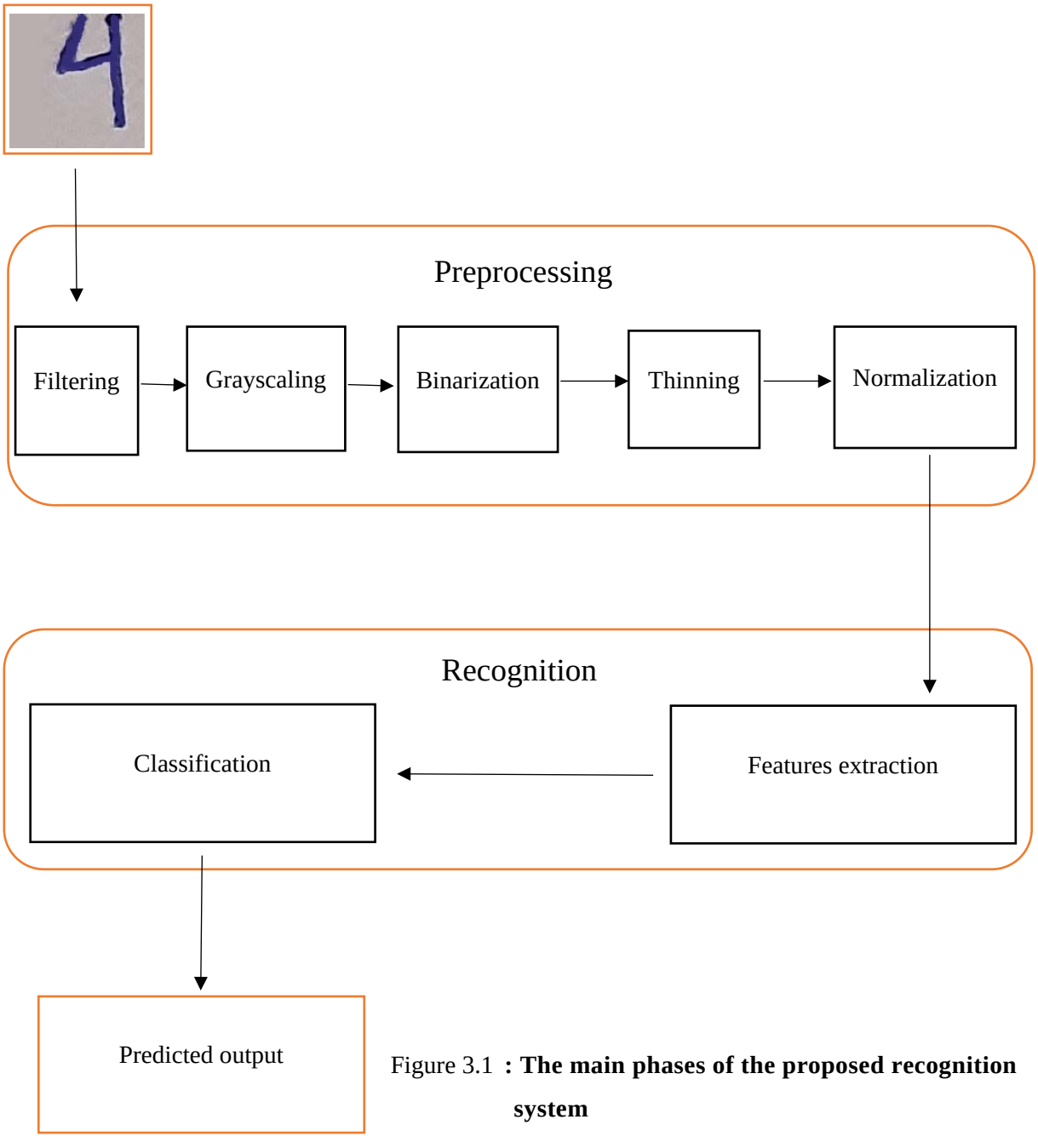


Figure 3.1 **: The main phases of the proposed recognition system**

1 Preprocessing

Generally, after we collect the data from different data resources that are considered a raw data, in usual cases it has some noise, missing values, or is represented on unsuitable format which needs to be handled correctly. The quality of the data should be checked before applying machine learning or deep learning algorithms so it firstly goes throw a crucial and mandatory task known as preprocessing. In this step, we apply many techniques of transforming, cleansing, cleaning our data from wrong values and fill the missing ones and put it in the right format in order to ensure or enhance the algorithm performance when results achieve high accuracy. As long as we are interested in deep learning problems, we often deal with images as our input data. The preprocessing phase passes through many steps accomplished by different algorithms as described next.

1.1 Gray scaling

Before we do any sort of manipulation on the collected images, we already realized that it is on RGB color channeling (Figure 3.2, Figure 3.4) which needs 32 bits of space to represent each pixel in each single image so it consumes a lot of hardware capacities in the other side we got a commonly used solution by transforming these images in gray color level.

The gray level is the value of the light intensity at a point. Grayscale images are obtained using multiple gray shades (256 gray shades available, from black to white-passing through a finite number of intermediate levels) between 0-255 to every one pixel on the image (Figure 3.3, Figure 3.5) that takes only 8 bits of space according to the formula shown below:

Grayscale value = (red value + green value+ blue value) /3. **(1)**

0=< Grayscale value <256

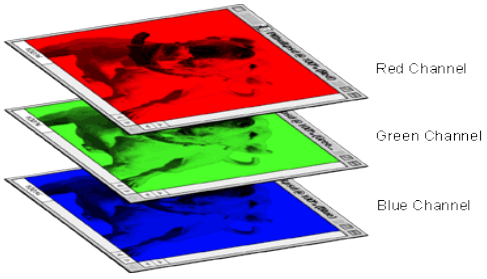


Figure 3.2 **: Image RGB color channels**

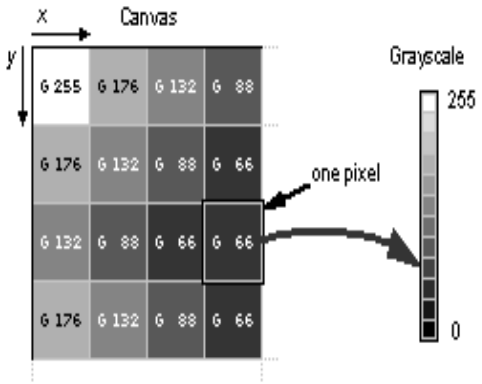


Figure 3.3 **: gray image pixels**

There are 3 methods to render the grayscale image [26]: the lightness method averages the most prominent and least prominent colors, the average method, and the luminosity method which forms a weighted average of the RGB colors. This preprocessing step is necessary so as to overcome the problems that may arise due to the use of pens of different colors and different intensities on various noisy and colored backgrounds.

The output of Gray scaling step is applied to the digit 4, as shown in Figure 3.4 and Figure 3.5.

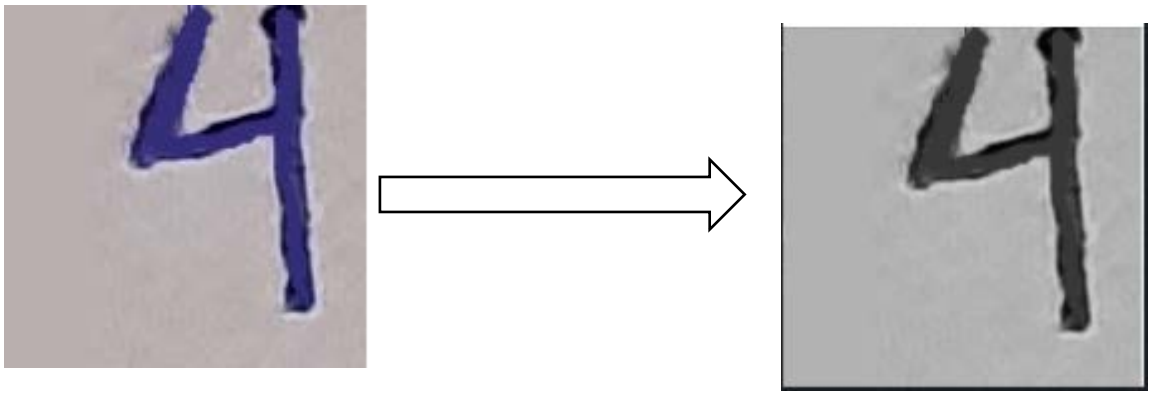


Figure 3.4 **: Image of the digit 4 on RGB**

Figure 3.5**: Grayscaling image of the digit 4**

1.2 Binarization

This process stands for the conversion of the gray image (Figure 3.5) into the bi-level of colors [27] (Figure 3.6) we mean by that only black and white colors for a purpose of classifying background pixels (0) from the foreground pixels (1) aim to increase the distinguishability also give the images more segmentation in our study case digits that we will feed it to our deep learning model to train. The binarization algorithm must preserve the maximum useful information and details present in the original image, and on the other hand, it must eliminate the background noise in an efficient way.

Many methods for document image binarization have been introduced in the literature, and these methods can be broadly categorized into three major groups: threshold-based methods, optimization-based methods, and classification-based methods.

1.2.1 Threshold-based methods

Thresholding is a process of converting a grayscale input image to a bi-level image by using an optimal threshold [28].

$$Ib(x,y)=\begin{cases} 0 \text{ if } In(x,y) < T \\ 1 \text{ if } In(x,y) >= T \end{cases} \quad \textbf{(2)}$$

In(x, y) describes the intensity at "n" gray levels at each point of the image, Ib(x, y) represents the intensity at two levels and T is the binarization threshold. If In(x,y) is greater than the threshold value then the image point corresponding to the maximum intensity value (white) is assigned. Otherwise, the point is considered black and assigned the minimum intensity value, and we calculate the threshold in the following way:

1. Select an initial estimate for T. (A suggested initial estimate is the midpoint between the minimum and maximum intensity values in the image.

2. Segment the image using T. This will produce two groups of pixels: G1, consisting of all pixels with intensity values ≥ T, and G2, consisting of pixels with value < T.

3. Compute the average intensity values μ1 and μ2 for the pixels in regions G1 and G2.

4. Compute the new threshold value: = ½ (+).

5. Repeat steps 2 through 4 until the difference in T in successive iterations is smaller than a predefined parameter To.

Also, there are two main categories of threshold-based binarization methods:

1.2.1.1 Global binarization

In which a single threshold value is applied for all pixels of the image [29,30], but it fails for the images with complex background means like uneven illumination, background noise, bleed through, and variation in illumination and contrast. Choose an optimal threshold is considered a challenge because the erroneous threshold estimation leads to pixels misclassification. We have used this technique in our model, specifically the Otsu algorithm [31].

Otsu's method is a popular global binarization technique, it divides the pixels into two classes one is foreground and the other is background (Figure 3.6). It chooses an optimal threshold that separates the image into two different classes. The threshold value is chosen such that the within-class variance is minimized and the between-class variance is maximized.

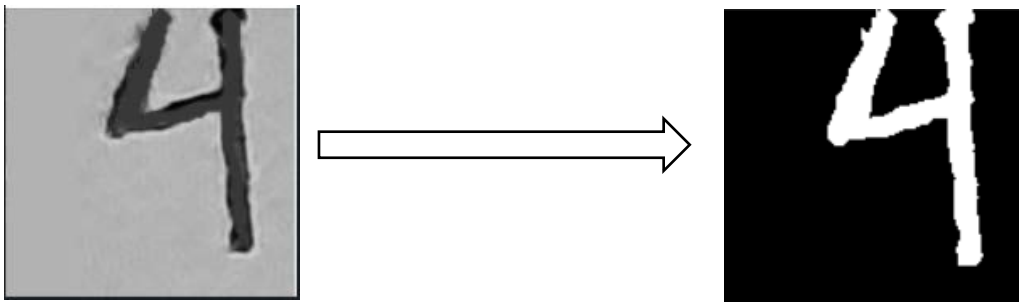


Figure 3.5: **Grayscaling image of the digit 4** Figure 3.6 **: Binarization image of the digit 4**

1.2.1.2 Local binarization

Instead of one threshold value for the whole image, this method uses different types of threshold values for each pixel depending upon the neighborhood pixels [32], the output results of these methods have shown that they give a lot of details, therefore present highly noisy backgrounds, sometimes even completely hide the writings.

It is for this reason that our choice of the method of binarization is carried on a global method of the fixed threshold. Some of the most prominent algorithms are: Niblack's Method [32], Adaptive Method [33], Sauvola's Method [34], Bernsen's Method [35], and Nick's Method [36].

1.2.2 Optimization-based methods

Researchers have recently introduced another class of methods in which binarization is formulated as an optimization problem. In most of these methods, typical examples include the use of Markov Random Fields (MRF) for binarization and intensity variation reduction using the Laplacian kernel [37].

1.2.3 classification-based methods

Methods belonging to this category are comparably new in the domain of document image binarization. supervised and unsupervised machine learning methods [38,39] have been applied to binarize document images.

1.3 Thinning

Such a process is a preprocessing operation of pattern recognition since a thinned object is easier to trace and hence is easier to recognize. It called a thinning algorithm, and the result is called a skeleton.

The basic idea of thinning according to this approach lies in the generation of a skeleton with a one-pixel width (Figure 3.7), this is done by iterative layer-by-layer erosion until only some skeleton like shape features are left. Thinning algorithms use reduction operators that transform binary pictures (Figure 3.6) (i.e., images containing only black and white points) only by

changing some black points to white ones, which is referred to as deletion of dark points along the edges of the pattern until it is thinned to a line, the points to be preserved are called terminal points or endpoints.

All thinning algorithms can be classified as either iterative or non-iterative [40].

1.3.1 Iterative method approaches

Thinning algorithms produce a skeleton by examining and removing the unwanted pixels through an iterative process in either a sequential or parallel way.

1.3.1.1 Sequential algorithms

Contour points are examined for deletion in a predetermined order. These algorithms have an advantage over raster scans because they require the examination of only the contour pixels instead of all the pixels in every iteration [41].

1.3.1.2 Parallel algorithms

This algorithm uses the directional strategy according to the North, South, East, West sequence: all the simple points having their North neighbor in the complementary are deleted in parallel, then all the simple points having their South neighbor in the complementary are deleted in parallel, and so on.

The usual practice is to use 3 x 3 neighborhoods but to divide each iteration into sub iterations or sub-cycles in which only a subset of contour pixels is considered for removal. At the end of each sub-iteration, the remaining image is updated for the next sub-iteration [42].

Parallel algorithms are considered superior for various reasons. Firstly, the deletion retention of a pixel in a sequential algorithm is more unpredictable, because the result depends partly on the order in which the pixels are processed. Since parallel algorithms examine all pixels simultaneously using the same set of conditions for pixel deletion, the results could be more isotropic.

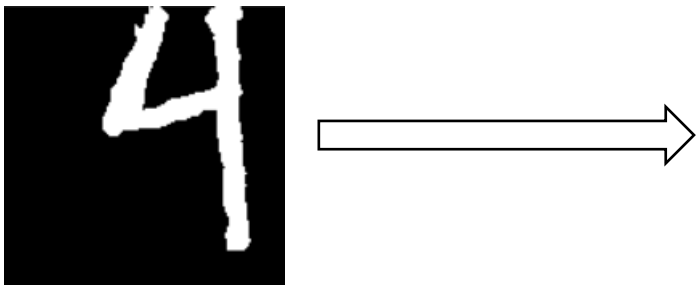


Figure 3.6: **Binarization image of the digit 4**

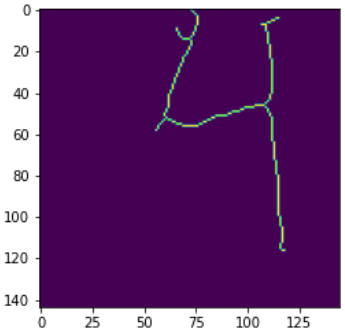


Figure 3.7 **: Thinning image of the digit 4**

1.3.2 Non-iterative method approaches (non-pixel based)

These methods produce a certain median or centerline of the pattern to be thinned directly in one pass, without examining all the individual pixels [43]. Unfortunately, these techniques are difficult to implement and slow as well.

Generally, one of the disadvantages of the skeletonization algorithms, is the alteration of the shape of digits such as the reduction of the double and unique diacritical points in the shape.

1.4 Smoothing and Noise Removal

Noise reduction consists of detecting and eliminating the pixels that represent it. Smoothing and suppression of noise (parasites) can be achieved by subjecting it to a process called filtering [44,45].

Filtering is a neighborhood operation, the value of a pixel is replaced by the value of a function applied to this pixel and its 8 neighbors (north, northwest, east, etc.). In the field of handwriting recognition, several types of filtering can be considered depending on the type of noise encountered or the desired processing.

There are two major groups of filters available:

1.4.1 Linear Filters

Like the smoothing filter that reduces the amount of spatial intensity variation between one pixel and its surroundings and the sharpening filter that emphasizes the fine details of an image, or the Gaussian filter [46] (Figure 3.8).

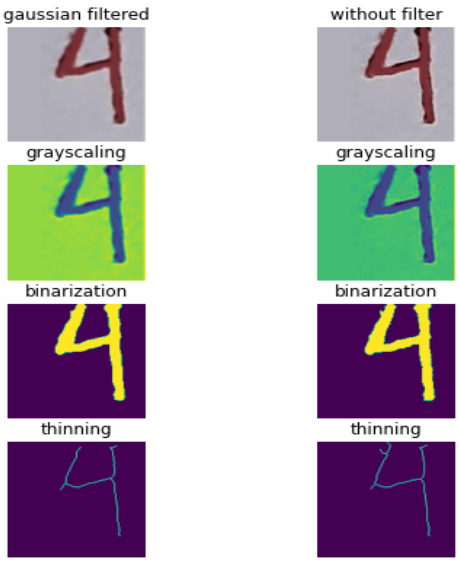


Figure 3.8 **: Image of the digit 4 with Gaussian filter 5x5 and without it.**

There is a noticeable change when applying the Gaussian filter in the thinning step, where we notice the deletion of some small fonts, and this helps in a good recognition of the digit.

1.4.2 Non-Linear Filters

These are used to remove impulse noise and negative outlier noise, and enhance dark values in the image, such as:

The median filter is a non-linear digital filtering technique, used to remove noise from an image or signal. Median filtering is very widely used in digital image processing because, under certain conditions, it preserves edges while removing noise in addition to the maximum, minimum, and range filters.

Generally, filters are distinguished according to the action they have on the spectrum (i.e., by the shape of their transfer function) [47]:

1.4.3 Low pass filters (smoothing)

This type of filtering consists of attenuating the components of the image having a high frequency (dark pixels), it is generally used to attenuate the noise of the image. This is why we usually talk about smoothing [48], for example:

The moving average filter which is a simple Low Pass FIR (Finite Impulse Response) filter commonly used for smoothing an array of sampled data. It takes L samples of input at a time (First Figure 3.9), and takes the average of those L samples and produces a single output point (Second Figure 3.9).

As the filter length increases (the parameter L) the smoothness of the output increases. The MA filter performs three important functions: It takes L input points (Figure 3.10), computes the average of those L-points, and produces a single output point (Figure 3.11).

Figure 3.9 **: The moving average filter with L=9x9**

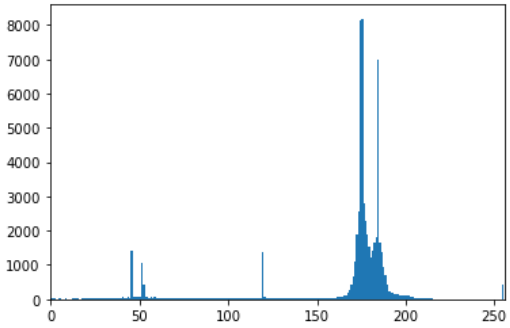


Figure 3.10 **: Histogram of digit 4**

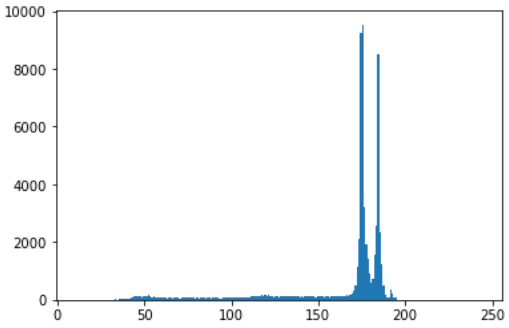


Figure 3.11 : **Histogram of digit 4 with moving average filter**

1.4.4 High-pass filters (emphasis)

Unlike low-pass, they attenuate the low-frequency components of the image and allow, in particular, to accentuate contours and their extraction. This is why we usually speak of accentuation.

1.5 Normalization

It is specifically applied to the foreground of an image only. Input numeral images may have lots of variation in size as writing style varies from one individual to another, this leads to the need of scaling the digit size within the image to a standard size (Figure 3.12), as this may lead to better recognition accuracy, in this dataset MNIST handwritten digits images are already size normalized and centered.

This operation generally introduces slight deformations on the images, certain characteristic features such as the ascender in the digit image can be eliminated following the normalization, which can lead to confusion between certain digits classes. There are three types of normalization methods:

1.5.1 Skew normalization

It is used because of the different types of handwriting fonts, the skew can damage the effectiveness of line recognition, so it is easy to detect and correct the baseline [49].

1.5.2 Slant normalization

The slope of the number in the parchment is called Slant. Formally, the number slant is defined as the angle between the longest term of the number and the vertical direction. One of the used methods is based on the center of gravity, another method uses the projection profiles and some used a variant of the Hough transform [51].

1.5.3 Size normalization

Character size may vary from one font to another and even within the same font after enlargement or reduction. This technique calculates the center of gravity of the entered shape, then aligns the centroid (center of gravity), position and shape (dimension) of a number's image [52].

The result of this step appears in Figure 3.12.

Figure 3.12 **: Normalization of handwritten digits**

2 Features Extraction

2.1 Introduction

The most important characteristic of large datasets when it is related to AI or machine learning problems is that they have a large number of variables or features, these variables require a lot of computing resources to process. The most important machine learning technology comes to perform features engineering or more specifically features extraction that defined as a part of the dimensionality reduction process. It is an initial set where the raw data is divided and reduced to more manageable groups which are easier in the processing phase [53]. So, feature extraction helps to get the best feature from those big data sets by selecting and combining features thus effectively reducing the amount of data, also enhancing model accuracy while at the same time describing the actual data set with accuracy and originality. These feature extraction tasks are done manually by machine learning specialists till an effective deep learning architecture comes to automate this task. This is exactly what we are discussing in the next sections where we employed CNN to extract features from handwritten digits images.

2.2 Features extraction with CNN

In the first place, CNN contains a convolutional layer which is the main key behind features extraction, it takes an image as input and also specific pre-configured matrix known as filter or kernel and applies dot multiplication across every possible patch on the image depending on

fixed filter sliding amount (n*n stride) to detect simple patterns. This process called features filtering which produces a convolved feature map from the original image. In addition, we can use multiple filters on the same image and we apply a nonlinearity activation function (relu) on each single instance in the features map. Figure (3.13, 3.14), describes an illustrative example to the hole previous steps that we explain by the stride 1*1 a 3*3 kernel:

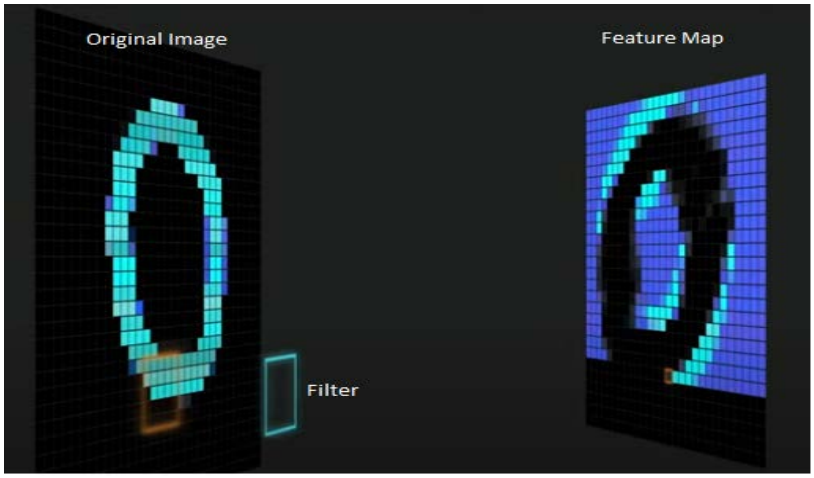


Figure 3.13 **: Image filtering preview**

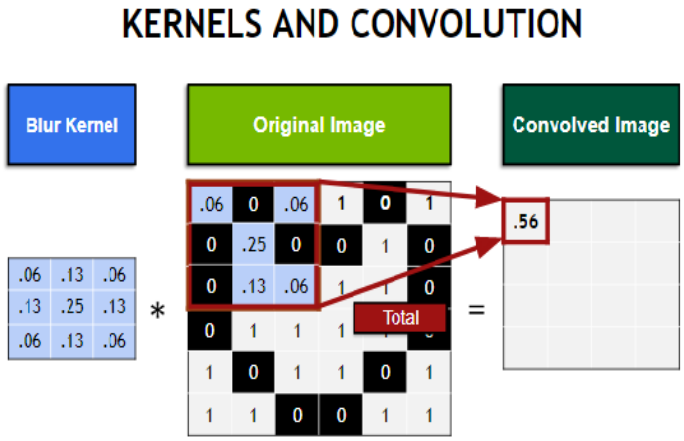


Figure 3.14 **: Ilustrative image about kernals and convolution 2**

The next layer that comes above is a pooling layer and it is as important as the convolutional layer. It is used to reduce the dimensions of the feature maps by keeping only the most important parts and discarding the others. As a result, it reduces the number of parameters to learn and the number of computations performed in the network. We can divide the most used pooling layers into two types:

2.2.1 Max Pooling

Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output of the max-pooling layer would be a feature map containing the most prominent features of the previous feature map figure 3.15.

Figure 3.15 **: Max-pooling**

2.2.2 Average Pooling

Average pooling computes the average of the elements present in the region of the feature map covered by the filter. Thus, while max-pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the average of features present in the patch figure 3.16.

Figure 3.16 **: Illustrative image of Average pooling**

3 Recognition

3.1 System Design and Training

In this chapter, we discuss the Convolutional Neural Network that was applied in this work to achieve the highest possible accuracy in handwritten digit recognition. Our architecture can be decomposed into two main units: the feature extraction unit and the decision unit.

3.1.1 Features Extraction Unit

We have set up a feature extraction unit from three convolutional layers with 128 (3x3) filters for the first layer, 256 (3x3) for the second convolutional layer, and 512 (3x3) filters for the last convolutional layer. Each one followed by a pooling layer with a dimension of 2x2 and an amount of displacement equals (3) [54] and [55]. In the beginning, we inserted the images in the form of matrices (the input layer) in the first convolutional layer where we scan the image from left to right by the filter by multiplying each value in the matrix that represents the raw image with the corresponding value of the filter matrix (filter weights matrix). Then, we recorded the result in a new matrix. After that, we shift the filter matrix by a predetermined amount (we mean a stride with size 1x1) and we repeat the process until we reach the end of the image matrix so that a new feature map with dimensions is produced for us to be calculated according to the following equation:

$$K=[(N-M)/S]+1 \qquad (3)$$

Where N is the size of image matrix, M is the size of kernel matrix, S is the size of stride matrix, and K is the size of the feature map. After that, the convolved output (features map) goes inside the batch normalization layer that increases the learning speed and stability by normalizing the previous layer outputs for the next layer, by subtracting the batch mean and dividing by the batch standard deviation. As a result, it helps to avoid weights and biases to become imbalanced with extremely high and low values which will over-influence the training process.

Following, we apply the activation function Relu (rectified linear unit) to this matrix that performs an element-wise operation and sets all the negative pixels to 0. It introduces non-linearity to the network, and the generated output is a rectified feature map, we represent it as a distinct layer in our work. Below is the graph of the ReLu function:

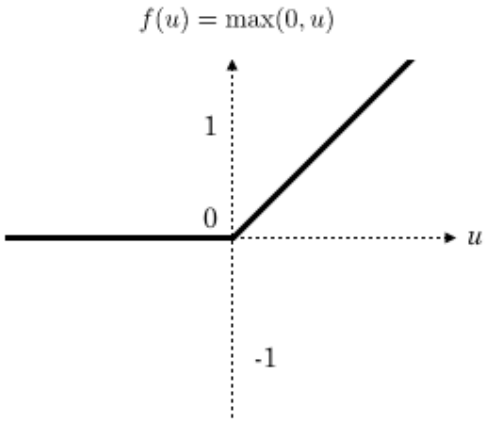


Figure 3.17 **: Relu activation function Graphic curve**

Since the network structure with a deeper number of layers is usually used in image classification tasks, this function improves the nonlinear ability of the network and avoid the gradient disappearance or the gradient explosion after multiple iterations of the network [43]. To demonstrate how the above calculation was done, we take an example of a 4x4 image, 2x2 filter, and 1x1 stride where:

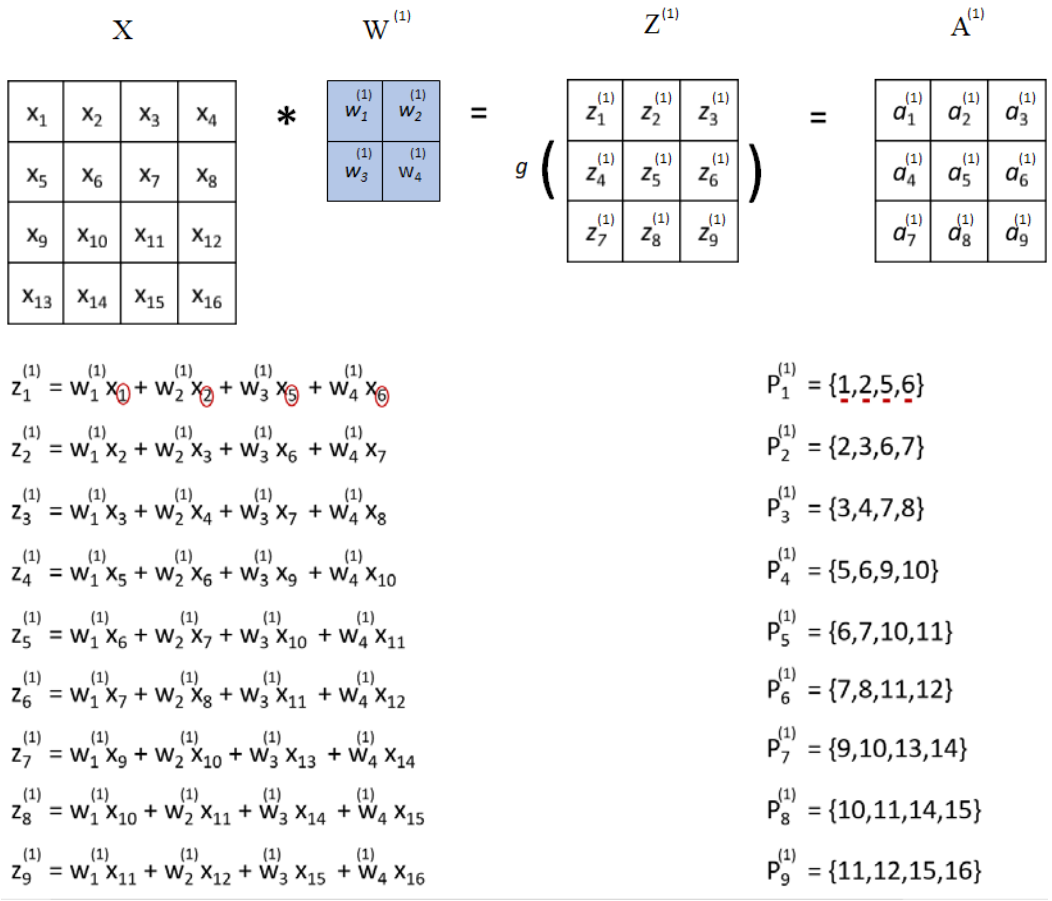


Figure 3.18 **: Illustration of how we calculate convolved image**

X represents image pixels.

W represents kernel weights(filter).

Z convolved image.

g activation function.

A rectified feature map.

Then we the convolved image into the pooling layer that we explained previously where the important features are selected and the rest are disposed according to the type of pooling layer

we used. Therefore, if we use the Max pooling, we will choose the largest value in the selected zone while if we use the average scaling, we will choose the median of the selected region. In our study case, we work with the Max pooling layer [54].

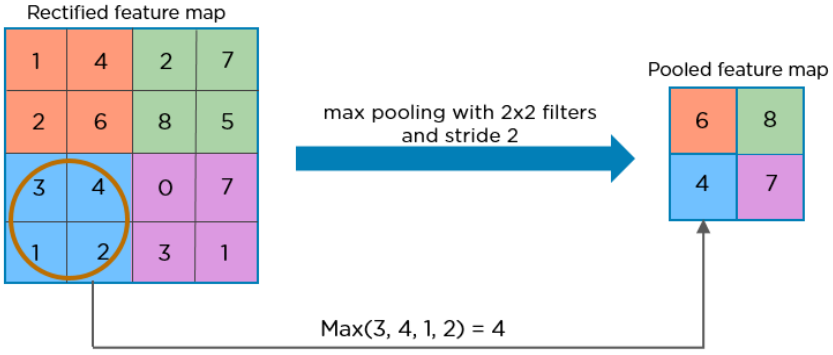


Figure 3.19 **: Illustration of pooled feature map**

Hereafter, we put a dropout layer for the purpose of preventing overfitting. This layer simply drops out neurons at a given probability in the training phase. Dropped-out neurons are counted as zero on the forward pass and their weights are not updated on the backward pass. By dropping out, randomly different "thinned" networks are trained that are sharing the weights. As a result, the network becomes less sensitive to the specific weights of neurons. This makes the network capable of better generalization and less likely to overfit the training data [57].

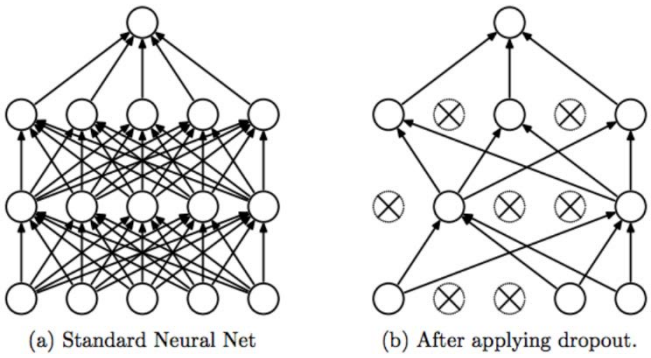


Figure 3.20 **: Illustration of after and before dropout neurons**

The output of this transformation is considered as the input of our next second convolutional layer then we do the same process and we repeat it again for the third conv layer until we get pooled feature map from the last pooling layer. Before we go to the decision subnet, we should make a crucial step which is the flattening process. In other resources, it is considered as a separated layer (flatten layer).

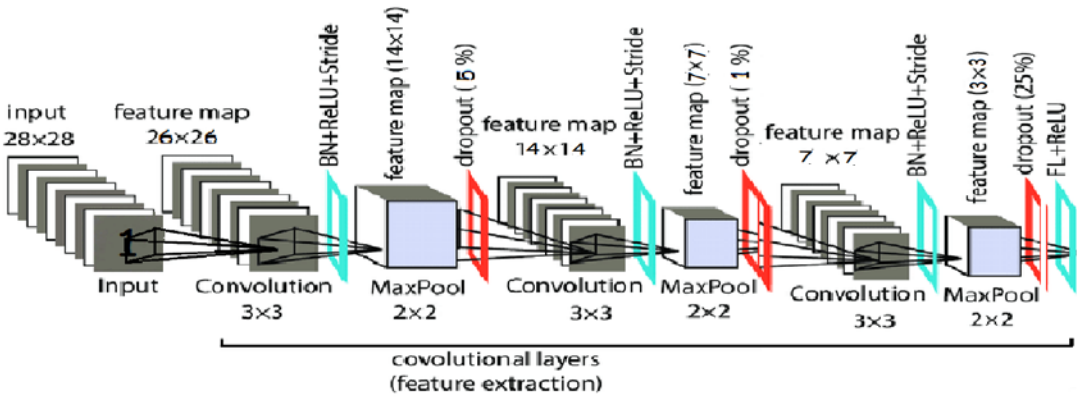


Figure 3.21 **: The architecture of the proposed feature extraction unit**

- **Flattening Process**

Flattening is used to convert all the resultant 2-dimensional arrays from pooled feature maps into a single long continuous linear vector. The flattened matrix is fed as input to the decision subnet.

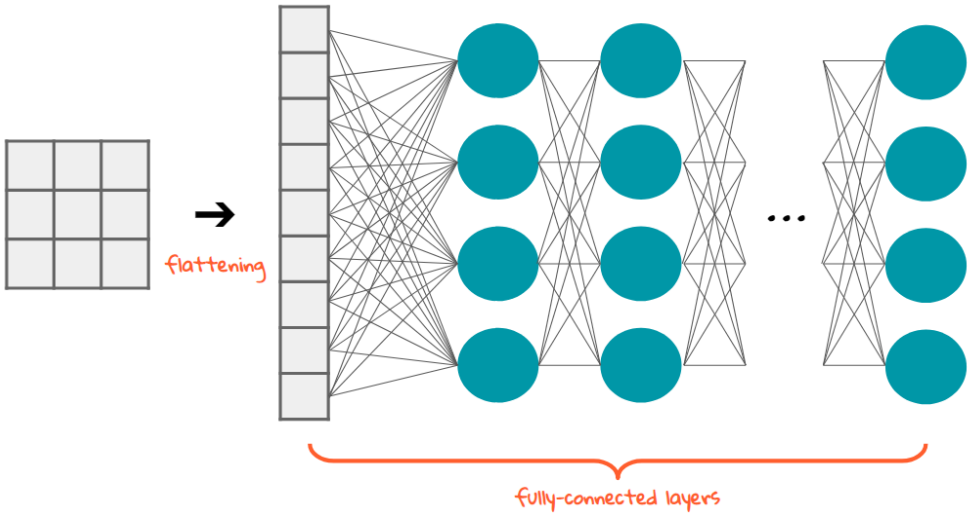


Figure 3.22 **: The flattening process in features extraction phase**

3.1.2 Decision Unit

We have reached the main part, which is the fully connected neural network, through which our model is trained to make the right decision in classifying the input images. This is done by a series of learning from making mistakes and adjusting them. We divide our ANN into an input layer (vector from the flattening process), a first hidden layer with 256 nodes and ReLu activation function, a second hidden layer with 128 nodes and ReLu activation, a third hidden

layer with 16 nodes and ReLu activation function, and an output layer with 10 nodes with a SoftMax activation function.

In the next subsection, we will describe the proposed neural network, and how it is trained.

- **Artificial Neural Network**

An Artificial Neural Network is an artificial simulation of the human brain structure and its components neurons, synapses, and so on. ANN consists of 3 major layers input, hidden, and output layers where every single layer is defined as a group of neurons (nodes) interconnected with weights, collaborating to do our classification task [58, 59] as described in Figure 3.23.

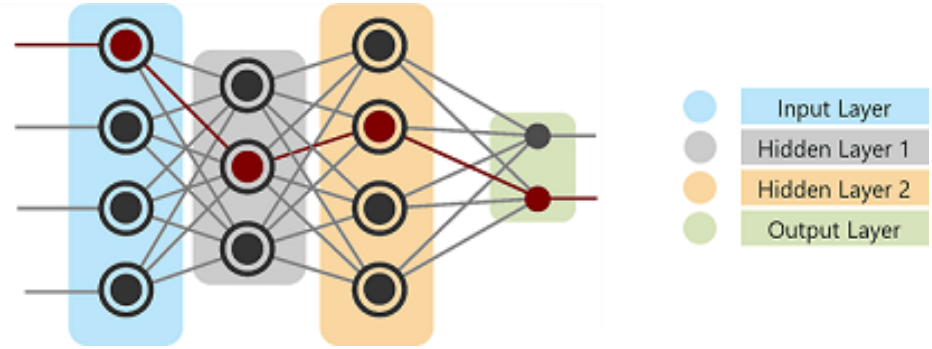


Figure 3.23 **: ANN general architecture**

- **Input layer**

The input layer of a neural network consists of input artificial neurons that bring the data into the system from which our neural network has to learn from, for further processing by subsequent layers attached with a set of random weights at the beginning. In the case of our study, the vector generated by the flattening process is fed to our NN as input data.

- **First Hidden layer**

It is a set of 512 artificial neural units where each node works as follows:

The inputs (x) received from the input layer are multiplied by their assigned weights w then adding all these products gives us the weighted sum. The weighted sum of the inputs and their respective weights are then applied to Relu activation function which becomes the inputs of the next hidden layer after we drop random neurons that match drop out amount. Figure 3.24 explains this task below:

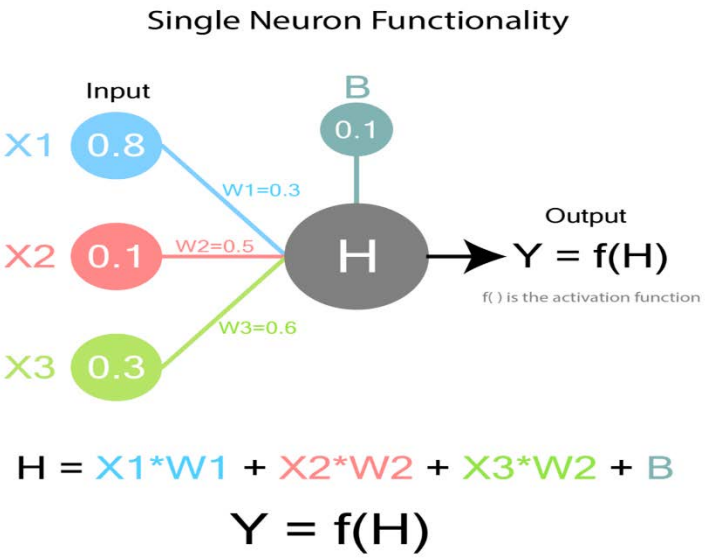


Figure 3.24 **: Explanation how single neuron works**

❖ W represents the weights between the nodes in the previous layer and the output node.

❖ X represents the nodes' values of the previous layer.

❖ B represents the bias, an additional value that is present for each neuron. Bias is basically a weight without an entry term. It is useful to have an additional amount of adjustability that does not depend on the previous layer.

❖ H is the intermediate node value (weighted sum). This is not the final value of the node. F(H) is an activation function and is something we can choose.

❖ Y is the final value of the node.

- **Second Hidden layer**

This layer consists of 128 neurons that takes the output of the previous layer as input, and we do the same process that we went throw in the first hidden layer. We will drop out all the neurons that hold a value of 0.2 or less after we run the activation function on the intermediate value.

- **Third Hidden layer**

It is also a set of 16 artificial neural nodes and it works as the same as the first and the second hidden layers. We repeat the same workflow but here we don't use the drop out technique because of the small number of neurons. The AI scientists advise to use the drop out layer when

there is a large number of artificial units stacked together and it has a proportional relationship with the drop out layer value.

- **Output layer**

The output layer is where we are supposed to get the targeted value. This represents what exactly our neural network is trying to predict or learn. Usually, it takes the output of the previous layer multiplied by weights and the softmax activation function is applied to generate the final values. They are known as the classification result appears in the form of a vector which contains 10 probability values from min 0 to max 1, and represents the percentage that our input image is under this class, figure 3.25 illustrates clearly the output layer.

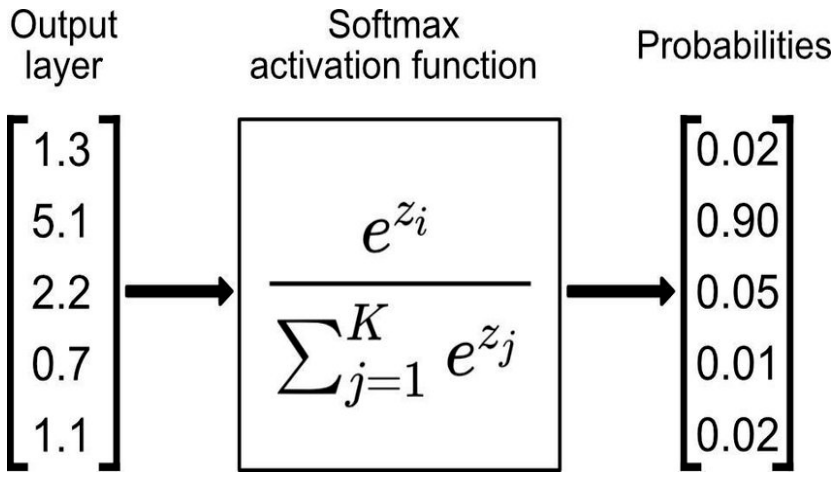


Figure 3.25 **: The output layer in CNN**

This process that we have gone through is called forward propagation. All the above-described layers are presented in Figure 3.26. We'll explain in the next lines how our model corrects the learning mistakes if the class prediction is wrong.

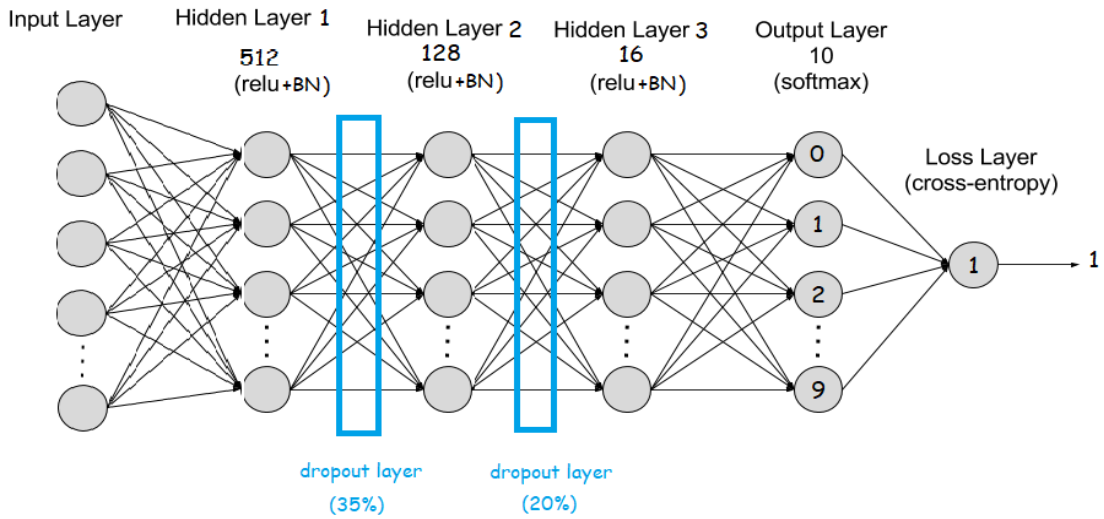


Figure 3.26 **: The CNN classifier**

Back Propagation

After we get the predicted classification, we calculate the prediction error using the error function (cost function). We use Cross-entropy loss, or log loss function in our model to measure the performance of the classification phase by calculating a separate loss for each class label per observation and summing the result [60]. See formula 4.

$$\text{Loss} = -\sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i \qquad \textbf{(4)}$$

Figure 3.27 describes in details the working method of Cross-entropy loss function where:

L represents the desired output.

S represents the predicted output.

D represents the cost function(cross-entropy).

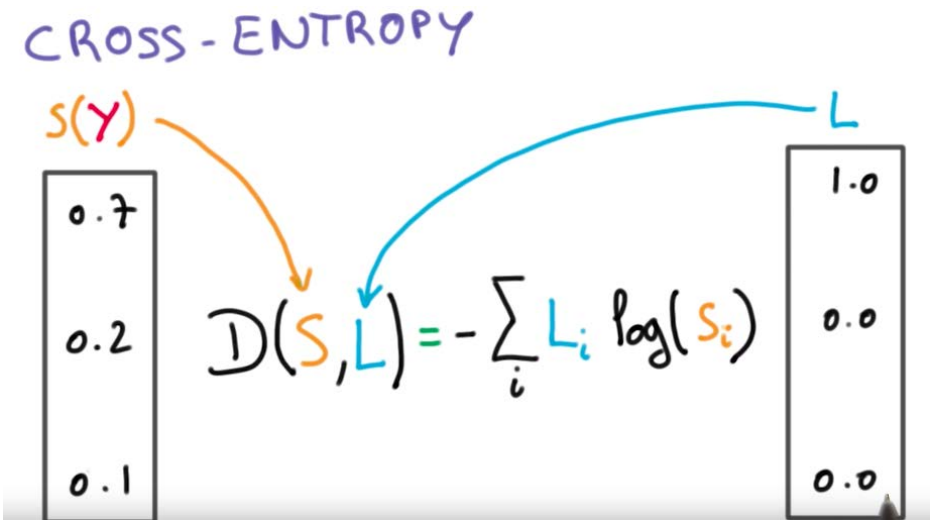


Figure 3.27 **: How cross-entropy loss function works**

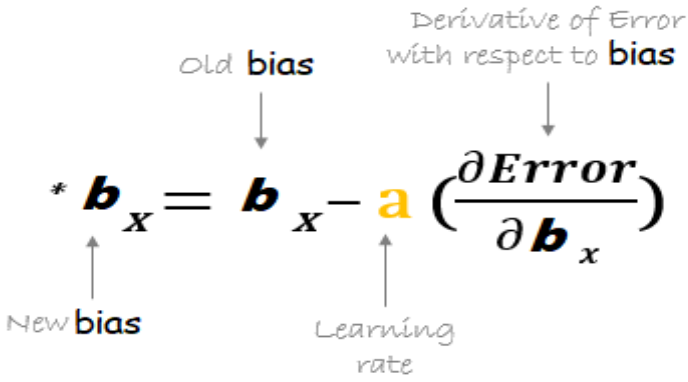
Afterward, we backpropagate the loss through every single hidden layer of our model to update the trainable params (weights, biases, and kernels weights) by making the multiple partial derivatives accomplish our target to find the derivative loss/weights using the chain rule as shown in formula 5.

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial z} * \frac{\partial z}{\partial w} \qquad (5)$$

New weight values are deduced according to the following formula:

$$^{*}W_x = W_x - a\left(\frac{\partial Error}{\partial W_x}\right) \qquad (6)$$

Old weight — Derivative of Error with respect to weight — New weight — Learning rate



(7)

Last but not least, we mention that the training process was performed using 60000 input images that reside in MNIST dataset. Thus, we make forward pass and backpropagation 60000 times. Eventually, once the training process is accomplished, we evaluate the model in Figure 3.28 with the validation dataset using several evaluation techniques (classification accuracy, recall, etc.) with the goal of fine-tuning our model hyperparameters.

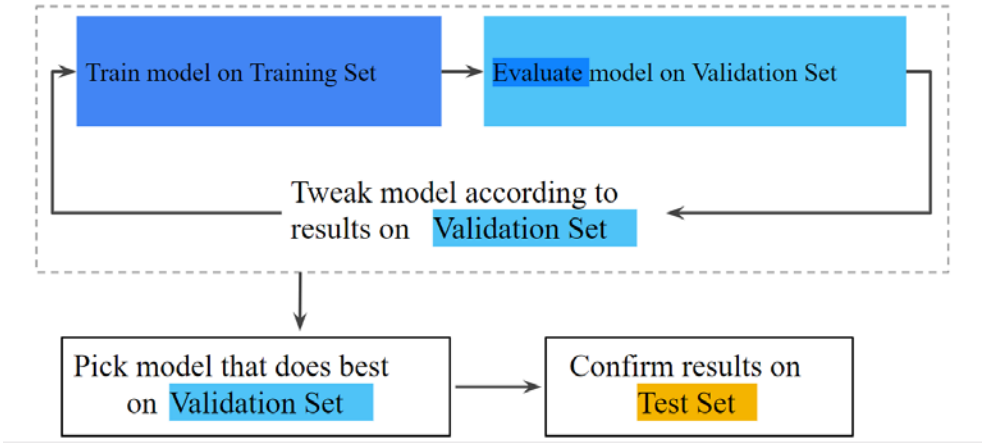


Figure 3.28 **: Schema of steps to train & test a model**

$$\text{Accuracy} = \frac{Total\ Positive\ Prediction}{Total\ Number\ of\ Prediction}$$

(OR)

$$\text{Accuracy} = \frac{(TP+TN)}{(TP+FP+FN+TN)} \quad \textbf{(8)}$$

$$\text{Recall} = \frac{True\ Positives}{True\ Positives + False\ Negative} \quad \textbf{(9)}$$

3.2 Testing the model

Once the training process is done, the proposed model that we feel confident about based on the least error rate and high approximate prediction will be evaluated and tested with a test dataset that contains 10000 images. This phase is compulsory to ensure that the model still perform well and match with the validation dataset results. Therefore, if we find the model accuracy is high, then we must ensure that the test/validation sets are not leaked into our training dataset.

CHAPTER 4

EXPERIMENTAL RESULTS

This chapter introduce the performance evaluation of the proposed CNN model which is used in the recognition of handwritten digits. We have used MNIST dataset from Kaggle [61] that is a subsidiary of Google LLC [62].

We divide this chapter into two parts as follows:

1 Discussion of the Obtained Simulated Results

The determination of the proposed network architecture is not a trivial task. Indeed, the structure which gives the best performance is only obtained after a series of experiments.

This part discusses the parameter settings, as well as the classification results by using TensorFlow [63] and keras [64] as our backend.

We will conduct the experiments by modifying the values of the input parameters in order to get results for different scenarios and then analyzing these results to draw knowledge and conclusions about the system under study. Among the input parameters that can be used, we try to investigate variants of CNN architecture with three layers (CNN_3L) and variants of CNN architecture with four layers (CNN_4L). A convolutional layer is parameterized by the size and the number of the maps, number of fully connected layer nodes, dropout, and epochs.

1.1 The settings architectures

A convolution layer is introduced by the Conv2D function of the sequential model of a convolutional network in our environment:

Conv2D (filters, kernel size, activation, padding, stride).

1.1.1 The number of convolution filters (15 epochs and size (3,3)

Model / filters		CNN_3L					CNN_4L				
		Acc train	Loss train	Acc test	Loss test	time (S)	Acc train	Loss train	Acc test	Loss test	time (S)
Case 1	Layer 1=16 Layer 2=128 Layer 3=32	99.70	0.94	99.43	2.17	121	99.62	1.27	98.61	4.72	136
	Layer 4=16										

Case 2	Layer 1=512 Layer 2=64 Layer 3=128 Layer 4=512	99.74	0.77	99.36	2.38	211	99.77	0.76	99.19	3.26	226
Case 3	Layer 1=128 Layer 2=256 Layer 3=512 Layer4=1024	**99.80**	**0.64**	**99.38**	**2.28**	**196**	**99.75**	**0.84**	**99.33**	**2.33**	**301**
Case 4	Layer 1=512 Layer 2=256 Layer 3=64 Layer 4=32	99.72	0.81	99.31	2.27	271	99.67	1.01	99.18	2.70	271

Table 4.1 A Comparison between different architectures and number of filters maps (MNIST)

In this table, we changed the number of filters for convolutional layers each time and we see the results of the accuracy and error ratio for each model, where the first column represents the training accuracy, meaning a set of 60,000 training images, while the second column represents the training error rate, the third column represents the testing accuracy, meaning a set of 10000 testing images, and the last column in each model represents the testing error percentage.

In case 1, we set the number of convolutional layer filters in a convex manner (ascending after that descending) as follows:

For model CNN_3L: Layer 1 16 filters, Layer 2 128 filters, Layer 3 32 filters

For model CNN_4L: Layer 1 16 filters, Layer 2 128 filters, Layer 3 32 filters, Layer 4 16 filters

In case 2, we set the number of convolutional layer filters in a concave manner (descending after that ascending) as follows:

For model CNN_3L: Layer 1 512 filters, Layer 2 64 filters, Layer 3 128 filters

For model CNN_4L: Layer 1 512 filters, Layer 2 64 filters, Layer 3 128 filters, Layer 4 512 filters

In case 3, we set the number of convolutional layer filters in an ascending manner as follows:

For model CNN_3L: Layer 1 128 filters, Layer 2 256 filters, Layer 3 512 filters

For model CNN_4L: Layer 1 128 filters, Layer 2 256 filters, Layer 3 512 filters

Layer 4 1024 filters

In case 4, we set the number of convolutional layer filters in a descending manner as follows:

	Layer 1 512 filters		Layer 1 512 filters
For model CNN_3L	Layer 2 256 filters	For model CNN_4L	Layer 2 256 filters
	Layer 3 64 filters		Layer 3 64 filters
			Layer 4 32 filters

After analyzing the results of (Table 4.1), we note that as we increase the number of layer filters in upward manner, from the first layer to the last layer, the training accuracy increased with a positive relationship, so we choose the case 3.

Note: Actually, we did not rely on the data mentioned in the above table only to make our decisions, but we also made other tests not mention here.

1.1.2 Size of filters (15 epochs)

Model / Filters size		CNN_3L					CNN_4L				
		Acc train	Loss train	Acc test	Loss test	Time (S)	Acc train	Loss train	Acc test	Loss test	Time (S)
Case 1	Layer 1=(3,3) Layer 2=(3,3) Layer 3=(5,5) Layer 4=(5,5)	99.77	0.73	99.37	2.49	286	99.77	0.77	99.39	2.47	705
Case 2	Layer 1=(3,3) Layer 2=(5,5) Layer 3=(3,3) Layer 4=(3,3)	99.78	0.67	99.32	2.88	215	99.75	0.88	99.29	3.01	335
Case 3	Layer 1=(5,5) Layer 2=(3,3) Layer 3=(3,3) Layer 4=(5,5)	99.79	0.68	99.25	2.87	195	99.78	0.73	99.37	2.79	615
Case 4	Layer 1=(3,3) Layer 2=(3,3) Layer 3=(3,3) Layer 4=(3,3)	**99.80**	**0.64**	**99.38**	**2.28**	**196**	**99.75**	**0.84**	**99.33**	**2.33**	**301**
Case 5	Layer 1=(5,5) Layer 2=(5,5) Layer 3=(5,5)	99.78	0.69	99.38	2.24	332	99.79	0.73	98.93	3.81	750

	Layer 4=(5,5)										

Table 4.2 A Comparison between different architectures and size of filters (MNIST)

In this table we have changed the size of the convolutional layer filters each time and then see the results associated with each instance, we used only popular filter sizes 3x3 and 5x5 and then tested as much as possible for possible cases.

From (Table 4.2) we conclude that the best choice for filters size is 3x3 for both models because it is medium-sized and very common in use in the convolution layers, we also note that the larger the size, the longer the training period.

1.1.3 The number of fully connected layer nodes (15 epochs)

Model / Nodes		CNN_3L					CNN_4L				
		Acc train	Loss train	Acc test	Loss test	Time (S)	Acc train	Loss train	Acc test	Loss test	Time (S)
Case 1	Layer 1=64 Layer 2=32 Layer 3=128	99.75	0.75	99.29	2.53	180	99.71	0.87	99.30	3.15	302
Case 2	Layer 1=512 Layer 2=128 Layer 3=16	**99.79**	**0.63**	**99.40**	**2.15**	**195**	**99.79**	**0.65**	**99.24**	**2.64**	300
Case 3	Layer 1=128 Layer 2=512 Layer 3=1024	99.77	0.81	99.32	2.65	198	99.73	0.89	99.31	2.69	316
Case 4	Layer 1=256 Layer 2=512 Layer 3=128	99.77	0.68	99.28	2.53	**196**	99.78	0.67	99.04	3.72	301

Table 4.3 A Comparison between different architectures and number of fully connected layer nodes (MNIST)

In this table we have done almost the same methodology as Table 4.1 but applied it to the fully connected layers as shown next:

In case 1, We set the number of fully connected layer filters in a concave manner (descending after that ascending) as follows:

For model CNN_3L Layer 1 64 nodes
Layer 2 32 nodes

For model CNN_4L Layer 1 64 nodes
Layer 2 32 nodes

	Layer 3 128 nodes		Layer 3 128 nodes

In case2, we set the number of fully connected layer filters in a descending manner as follows:

	Layer 1 512 nodes		Layer 1 512 nodes
For model CNN_3L	Layer 2 128 nodes	For model CNN_4L	Layer 2 128 nodes
	Layer 3 16 nodes		Layer 3 16 nodes

In case3, we set the number of fully connected layer filters in an ascending manner as follows:

	Layer 1 128 nodes		Layer 1 128 nodes
For model CNN_3L	Layer 2 512 nodes	For model CNN_4L	Layer 2 512 nodes
	Layer 3 1024 nodes		Layer 3 1024 nodes

In case 4, we set the number of fully connected layer filters in a convex manner (ascending after that descending) as follows:

	Layer 1 256 nodes		Layer 1 256 nodes
For model CNN_3L	Layer 2 512 nodes	For model CNN_4L	Layer 2 512 nodes
	Layer 3 128 nodes		Layer 3 128 nodes

From (Table 5.3), we find that all results are very close, so our selection is based on the speed of training shown in Case 2.

1.1.4 Dropout

The basic work of the dropout enables us to stop some randomly selected neurons (both hidden and visible) from participating in the training process. It is better to use it on a large number of neurons or feature maps.

Table 4.4 is divided into two parts, part 1 is for convolutional layers and part 2 is for fully connected layers.

Dropout \ Model		CNN_3L					CNN_4L				
		Acc train	Loss train	Acc test	Loss test	Time (S)	Acc train	Loss train	Acc test	Loss test	Time (S)
Case 1	Layer 1=0 Layer 2=0.1 Layer 3=0.25	**99.77**	**0.75**	**99.46**	**2.03**	**195**	**99.70**	**1.03**	**99.34**	**2.40**	**315**

	Layer 4=0.35										
Case 2	Layer 1=0.2 Layer 2=0.2 Layer 3=0.2	99.65	1.09	99.28	2.49	197	99.64	1.12	99.42	2.01	316
	Layer 4=0.2										
Case 3	Layer 1=0.35 Layer 2=0.25 Layer 3=0.1	99.66	1.10	99.39	1.98	193	99.64	1.16	99.33	2.37	315
	Layer 4=0										
Fully Connected Layers											
Case 1	Layer 1=0 Layer 2=0.2 Layer 3=0.35	97.44	7.53	99.20	3.98	181	97.86	6.73	99.18	5.19	302
Case 2	Layer 1=0.25 Layer 2=0.25 Layer 3=0.25	98.98	3.65	99.51	2.32	195	98.97	3.54	99.47	3.09	305
Case 3	Layer 1=0.35 Layer 2=0.2 Layer 3=0	**99.73**	**0.82**	**99.36**	**2.35**	192	**99.79**	**0.75**	**99.04**	**3.86**	**301**

Table 4.4 A Comparison between different architectures and dropout (MNIST)

In this table, we have tested these cases based on what we have found from the previous tables, as the dropout is applied to the number of convolutional layer filters and the number of nodes in fully connected layers, so that part 1 of the last table we chose Case 1 because the more filters there are, the greater the amount of dropout. We can see this in the results of the table.

As for Part 2, the greater the number of nodes, the greater the amount of dropout, so we chose case 3.

1.1.5 Epochs

Model / Epochs	CNN_3L					CNN_4L				
	Acc train	Loss train	Acc test	Loss test	Time (S)	Acc train	Loss train	Acc test	Loss test	Time (S)
1	95.29	22.48	97.82	7.21	15	97.90	24.72	94.40	7.47	23
5	98.91	3.63	99.13	2.98	68	98.92	3.91	98.75	4.00	107

15	99.60	1.18	99.54	1.84	198	99.66	1.15	99.49	1.89	317
30	99.78	0.66	99.56	1.82	393	99.60	1.18	99.54	1.84	632
40	99.90	0.3	99.39	2.89	523	99.91	0.31	99.41	2.42	842

Table 4.5 A Comparison between different architectures and epochs (MNIST)

After analyzing the obtained results, we noted the following remarks:

- From (Table 4.5), the accuracy of training and testing increases with the number of epochs, this reflects that with each epoch the model learns more information.
- If the precision is decreased like in Case 1 (epochs=1), then we will need more information to make our model learn, and therefore we must increase the number of epochs. This is called underfitting. Look at the training accuracy rate of both models (CNN_3L = 94.93 and CNN_4L = 93.92) and the high error rate.
- Similarly, the increased number of epochs causes overfitting and consequently can influence the recognition accuracy.
- According to the table, we conclude that the error rate is poor when we increase the number of epochs. So, to obtain a minimum error rate or a maximum recognition rate our choice must respect an exemplary number of epochs which is 30. Although the number of epochs 40 gave a high training accuracy of 99.90, the learning error rate is large compared to the number of epochs 30.

Generally, the model CNN_3L is better than CNN_4L because increasing the number of layers leads to overfitting. The concept of dropout may be used to solve the problem of overfitting.

1.2 Model CNN_3L results

After selecting the appropriate parameters for our model, which is applied to the MNIST data, we now come to see the results of this model.

```
Epoch 21/30
1875/1875 [==============================] - 22s 12ms/step - loss: 0.0093 - accuracy: 0.9973
Epoch 22/30
1875/1875 [==============================] - 22s 12ms/step - loss: 0.0088 - accuracy: 0.9973
Epoch 23/30
1875/1875 [==============================] - 22s 12ms/step - loss: 0.0086 - accuracy: 0.9972
Epoch 24/30
 696/1875 [==========>...................] - ETA: 12s - loss: 0.0092 - accuracy: 0.9970
```

Figure 4.1 **: The training of CNN3_L model**

Figure 4.1 describes a section of the results of the model training using 30 epochs, where we notice a gradual decrease in the error rate and a relative stability in the training accuracy.

```
Epoch 23/30
1875/1875 [==============================] - 22s 12ms/step - loss: 0.0086 - accuracy: 0.9972
Epoch 24/30
1875/1875 [==============================] - 23s 12ms/step - loss: 0.0081 - accuracy: 0.9973
Epoch 25/30
1875/1875 [==============================] - 22s 11ms/step - loss: 0.0072 - accuracy: 0.9977
Epoch 26/30
1875/1875 [==============================] - 22s 12ms/step - loss: 0.0071 - accuracy: 0.9978
Epoch 27/30
1875/1875 [==============================] - 22s 12ms/step - loss: 0.0064 - accuracy: 0.9981
Epoch 28/30
1875/1875 [==============================] - 22s 12ms/step - loss: 0.0053 - accuracy: 0.9984
Epoch 29/30
1875/1875 [==============================] - 22s 12ms/step - loss: 0.0067 - accuracy: 0.9980
Epoch 30/30
1875/1875 [==============================] - 22s 12ms/step - loss: 0.0066 - accuracy: 0.9978
313/313 [==============================] - 2s 6ms/step - loss: 0.0182 - accuracy: 0.9956
```

Training

Testing

Figure 4.2 **: The training and testing results on MNIST data**

Figure 4.2 shows what we found earlier in (Table 4.5). Where we reached the latest version of our model by changing the parameters, where we notice a relative decline in the error rate until it reaches the last epoch 0.66% and a gradual rise in the accuracy of training until it reaches 99.78%, As for the testing results, they are indicated by the red arrow.

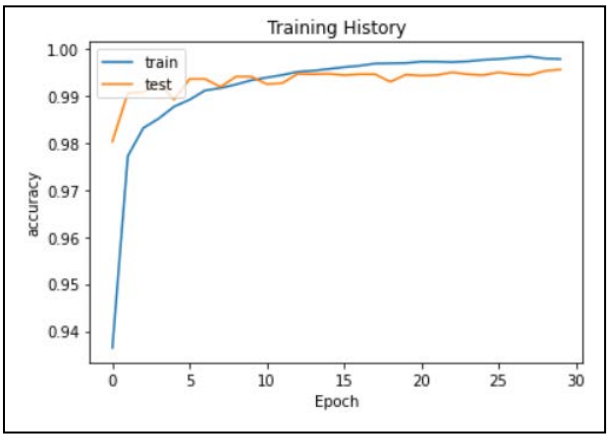


Figure 4.3 **: Graph illustrating the transition of training accuracy of CNN_3L with increasing number of epochs (Accuracy v/s Number of epochs)**

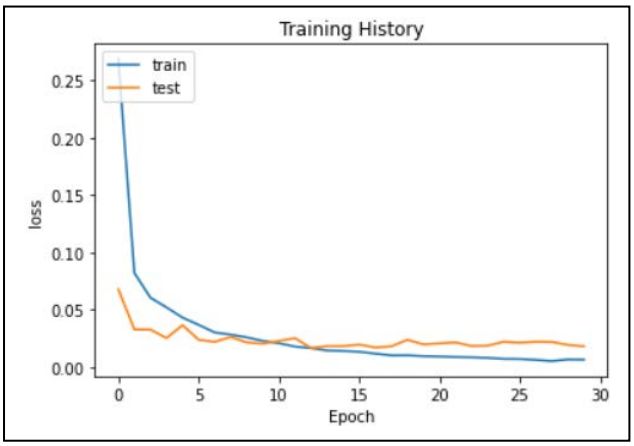


Figure 4.4 **: Graph illustrating the transition of training loss of CNN_3L with increasing number of epochs (Loss rate v/s Number of epochs)**

Moreover, Figure 4.3 shows the evolution of training and testing accuracy in terms of the number of epochs where:

- In the first graphic curve we notice a graphic jump in the accuracy of training in a very small number of epochs, then relative stability with a slight rise, as the accuracy rate approaches 100% as the number of epochs increases.
- The second graphic curve begins with an initial accuracy of 98%, where we notice a slight fluctuation in the accuracy of the test, then it begins to gradually rise with the increase of the epochs.

On the other hand, Figure 4.4 represents the evolution of the training and test error rate in terms of periods, where:

- The first graphic curve represents a sharp descent in the training error rate until reaching 8% and the number of epochs is 2 and then begins with a slight descent along the ordinate axis.
- With regard to the second graphic curve, it starts with an initial ratio of 7%, then slight fluctuations, then relative stability in the graphic curve until it reaches a minimum value equal to 1.82%.

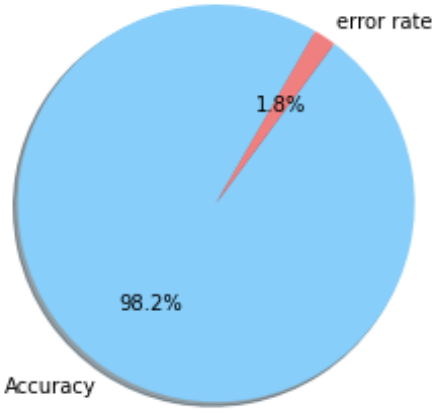


Figure 4.5 **: The testing error rate in our CNN_3L model**

Figure 4.5 represents the number of images which our model misrecognized them, where the ratio of 1.82% means 182 out of 10,000 testing images.

	precision	recall	f1-score	support
class 0	0.994914	0.997959	0.996434	980
class 1	0.995614	1.000000	0.997802	1135
class 2	0.995155	0.995155	0.995155	1032
class 3	0.996040	0.996040	0.996040	1010
class 4	0.993915	0.997963	0.995935	982
class 5	0.993274	0.993274	0.993274	892
class 6	0.995807	0.991649	0.993724	958
class 7	0.997067	0.992218	0.994637	1028
class 8	0.997945	0.996920	0.997432	974
class 9	0.996028	0.994054	0.995040	1009
accuracy			0.995600	10000
macro avg	0.995576	0.995523	0.995547	10000
weighted avg	0.995602	0.995600	0.995599	10000

Figure 4.6 **: The testing results (precision and recall) of the proposed model**

Here is a brief definition of each accuracy criteria described in the previous chapter:

• **Precision** is the number of correct images predicted in a particular class divided by the total number of images predicted in the same class.

• **Recall** is the number of correct images predicted in a particular class divided by the total number of images for the same digit in all classes (support).

• **F1 Score** is a harmonic average between precision and recall and it is the complete result of the model's efficiency and accuracy.

Support is the number of testing digits in each class.

Accuracy is the number of total testing images divided by the number of correct, predicted images in each category.

Macro avg: The macro-averaged F1 score (or macro F1 score) is computed by taking the arithmetic mean (aka unweighted mean) of all the per-class F1 scores.

Weighted avg: The weighted-average F1 score is calculated by taking the mean of all per-class F1 scores while considering each class's support.

Figure 4.6 puts our attention to the fact that digit (1) has been 100% recognized which means that all the digit (1) images. For the rest of the digits, we see a fluctuation in the classification ratios for each class, and that's often due to the similarity between the digits.

1.3 A practical test on a number

```
Predicting the digit 4 in all classes
[[1.09556226e-04 9.26551002e-05 7.98042456e-05 7.71156629e-05
  9.95409310e-01 1.93761356e-04 1.15487986e-04 3.19135113e-04
  1.24893602e-04 3.47810728e-03]]
The digit class is [4]
```

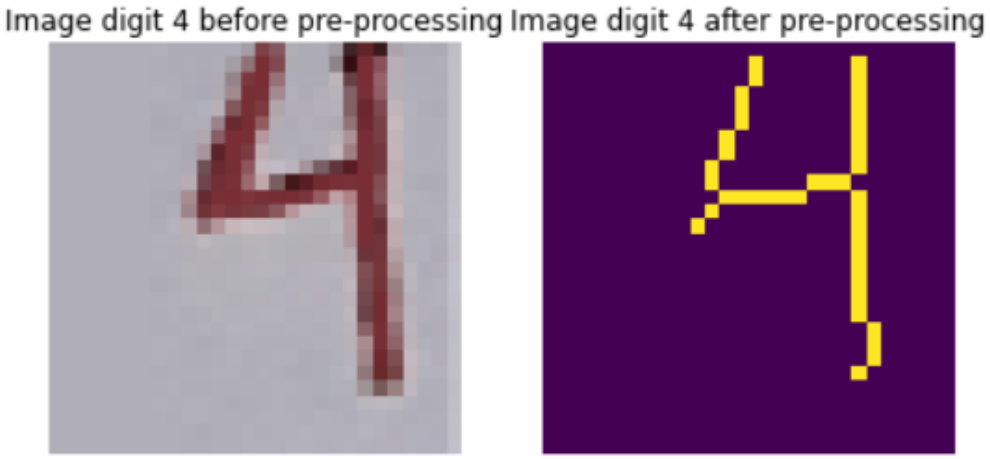


Figure 4.7 **: The result of predicting the digit 4**

From Figure 4.7, we find that our model has recognized the digit 4 with a ratio of 99.54%, which is an impressive percentage for the rest of the remaining classes, as we see that the closest class to it is 9 with a ratio of 0.34%, this is due, as we said earlier, to the similarity between the two numbers, but the ratio is very different between them, so our model works fine.

2 Comparison with Existing Research Work

Based on the literature review, we believe that the current obtained results are satisfactory when digit recognition performances in the literature are considered. We decided to review and tabulate the performances of some of the works mentioned in the "Related Works" section along with our results in the Table 4.6. Although there is no exact match between the tests, it may give an idea about how successful our results are.

References	Algorithms	Training Accuracy %	Testing Error Rate %
[15]	HOG–SVM	97.25	-
[23]	3-DNN	98.08	2
[13]	DBN	98.75	2.49
[14]	CNN	99.2	2.63
[16]	CNN	99.53	4
[19]	DNN–hidden markov	-	0.83
[21]	CNN–35-net	99.77	0.39
[22]	CNN –SVM	99.81	0.19
Our model	CNN	99.78	1.82

Table 4.6: Comparing training and testing results with previous works on MNIST.

From Table 4.6, we notice convergence in the results of training accuracy and varying differences in the testing error rate, where the best model 16 achieved the lowest percentage present, and this was due to the failure to identify 19 testing images, including image No. 5938, which is considered the most difficult image in the testing group, there were only 9 subjects identified. It is correctly recognized as a digit "5". All the other people said it as a digit "3" in the survey. Upon examining this image closely, we see that the digit is so cursively written and there exist no gaps between the top and middle strokes in the upper part of the image. It is so hard for humans to identify what the numeral is without the ground truth, as for our model, it recognized this picture and classified it as the digit 5.

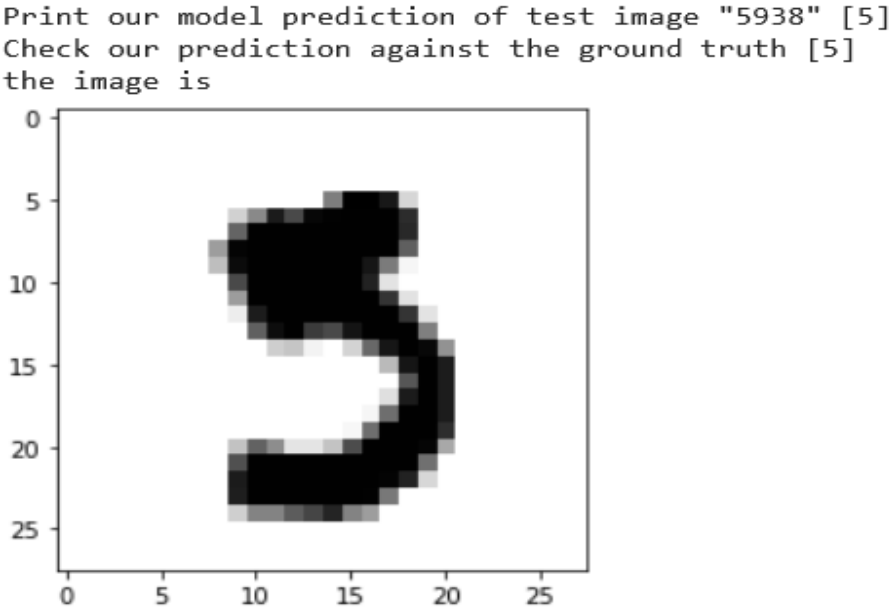


Figure 4.8 **: The result of predicting the digit 5**

3 Conclusion

For the MNIST dataset, the best recognition result in the literature was 99.81% using a hybrid model CNN (Convolution Neural Network) as a feature extractor and an SVM as a classifier [22]. Therefore, our results are not too far from this hybrid model although we only have used 3 convolutional layers and 3 layers of FC-ANN.

GENERAL CONCLUSION AND FUTURE WORK

In the end, we say that the development controlled by machine education is a great jump in the life of all human beings and in the field of technology after a winter era that lasted more than 15 years of stagnation. Machine learning was considered a term written ink on paper and what the various algorithms achieved by machine learning were effective in most tasks. They were used according to the purpose to be accomplished from learning under the supervision of programmers to correct its mistakes manually or without supervision, where the machine corrects its mistakes automatically without the need for external intervention.

Based on the fact that one of the most important problems that programmers and developers have faced is how a machine can recognize and accurately identify things. Here comes the problem that we have worked on, which is how to make the machine able to recognize handwritten digits because of the similarity of digits in many properties. This issue becomes an important topic due to its use in several fields such as banking, education, and so on.

Handwritten digits recognition is an active area of research that always needs improving the inaccuracy. In the model we have proposed for classifying handwritten digits in MNIST database using primarily convolutional networks, in addition to a classifier which is a neural network, we get very good results. Surprising in comparison with previous works in the same field which used various machine learning techniques such as KNN, Random Forest, clustering, we found an accuracy rate of 99.52 and an error rate of 0.2 in the training stage, also an accuracy of 99.10 and an error amount equal to 0.4 in the testing stage, which is the largest result reached.

In the first, we are introduced by mentioning human endeavors to make his life easier by making most of the tasks automated by making machines, such as the beginning of the invention of the computer until he reached the ability to teach it. Then we referred to a critical topic, which is optical character recognition and how a person can allow the computer to recognize characters in general, after that, we mentioned the main problem on which this thesis revolves mainly, which is how can we recognize digits written by hand, how this problem can be solved by various machine learning techniques and clarify our thesis subdivisions. In addition, chapter 1 talks about the dataset that we use during all the phase of training and testing which is the Mnist dataset that holds 70000 samples divided to 60000 for training and 10000 for testing, and we describe the different features and benefits behind why we choose exactly Mnist dataset. Going to chapter 2, we touched the previous work in the same field of research and mentioned the most important models that were reached using various machine learning algorithms and mentioned the accuracy of each model. Chapter 3 represents the important work that we did in

details. Basically, we show how digits images are preprocessed before they are fed into the model using various image processing functions like normalization binarization and so on. Also, we explain very specifically how we extracted each digit image features that distinguish one number from another using CNN. Next, we define the units that make up the proposed system, which are features extraction unit and classifier artificial network unit. Last but not least, in chapter 4, we discussed the different experimental results and we compared each try results to improve the model accuracy by finetuning the model and we mentioned the various problems that we faced in this phase like overfitting and misclassification of digits…etc. Lastly, we end up with a conclusion of all the work.

The work we have done throughout this research constitutes an important and contributing step in the recognition of handwritten digits. However, we believe that it can be improved and expanded through the following points:

- Expanding the research field to classify digits of all kinds, Arabic, Roman, etc.
- Improving the handwritten digits recognition performance using other enhanced deep learning techniques.
- Various CNN structures, namely CNN-RNN and CNN-HMM hybrid models and domain-specific recognition systems, can also be investigated.
- Modify the trainable parameters in order to achieve greater accuracy, namely the number of layers, learning rate, kernel sizes convolutional filters, and other checking algorithms plus learning steps.

In the future, we are looking to develop other related new architectures or optimization algorithms in order to increase the efficiency of the model by reducing computation time, hardware exploitation, and complexity.

BIBLIOGRAPHY

[1] D.Poisson, S. Mahjoub. Mémoire doctorat. Comment améliorer les techniques de reconnaissance de formes 3D?. Ecole supérieur de génie informatique, Paris. 2010.

[2] Dr. Firoj Parwej.2013. The State of the Art Recognize in Arabic Script through Combination of Online and Offline. International Journal of Computer Science and Telecommunications [Volume 4, Issue 3, March 2013] pp 60-66.

[3] Y. Lu, CL.Tan, ―Combination of multiple classifiers using probabilistic dictionary and its application to postcode recognition‖. Pattern Recognition, vol: 35. Pages: 2823–2832. 2002.

[4] A. Brakensiek, J. Rottland, G. Rigoll,‖Confidence measures for an address reading system‖. Seventh international conference on document analysis and recognition, ICDAR2003, pp 294–298. 2003.

[5] K. Marukawa, M. Koga, Y. Shima, and H. Fujisawa, "A High Speed Word Matching Algorithm for Handwritten Chinese Character Recognition.," in MVA, 1990, pp. 445–450.

[6] N. Gorski, V. Anisimov, E. Augustin, O. Baret, D. Price, JC. Simon, ―A2iA check reader: families of bank check recognition systems‖. Proc. fifth int'l conf. document analysis and recognition,pp 523–526.1999.

[7] Q. Xu, L.Lam, CY. Suen, ―A knowledge-based segmentation system for handwritten dates on bank cheques‖. Sixth international conference on document analysis and recognition, ICDAR2001, pp 384–388. 2001.

[8] B. Zhang and S. N. Srihari, "Fast k-nearest neighbor classification using cluster-based trees," IEEE Transactions on Pattern analysis and machine intelligence, vol. 26, no. 4, pp. 525-528, 2004.

[9] E. Kussul and T. Baidyk, "Improved method of handwritten digit recognition tested on MNIST database," Image and Vision Computing, vol. 22, no. 12, pp. 971-981, 2004.

[10] Mnist database, https://yann.lecun.com/exdb/mnist/

[11] Cortes, C. and Vapnik V. (1995) "Support vector networks", Machine Learning, 20.

[12] Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009) "What is the best multistage architecture for object recognition?" In: Proceedings of IEEE 12th International Conference on Computer Vision (ICCV) 2146-2153.

[13] Hinton, G. E., Osindero, S. and Teh, Y. W. (2006) "A fast learning algorithm for deep belief nets." Neural Computation 18(7):1527-1554.

[14] Fathma Siddique, Shadman Sakib, Md. Abu Bakr Siddique, "Recognition of Handwritten Digit using Convolutional Neural Network in Python with Tensorflow and Comparison of Performance for Various Hidden Layers".

[15] R.Ebrahimzadeh, M.Jampour, 'Efficient Handwritten Digit Recognition based on Histogram of Oriented Gradients and SVM', International Journal of Computer Applications (0975 – 8887) Volume 104 – No.9, October 2014.

[16] R.Dixit, R.Kushwah, S.Pashine, 'Handwritten Digit Recognition using Machine and Deep Learning Algorithms', International Journal of Computer Applications (0975 – 8887)

Volume 176 – No. 42, July 2020.

[17]. Younis KS, Alkhateeb AA (2017) A new implementation of deep neural networks for optical character recognition and face recognition. In: Proceedings of the new trends in information technology, Jordan, Apr 2017, pp 157–162

[18]. Gupta, A.; Sarkhel, R.; Das, N.; Kundu, M. Multiobjective optimization for recognition of isolated handwritten Indic scripts. Pattern Recognit. Lett. 2019, 128, 318–325.

[19] L. Deng, and D. Yu, "Deep convex network: a scalable architecture for speech pattern classification", in Proccedings of International Speech Communication Association, pp 2285-2288, 2011.

[20] Ciresan, D. C., Meier, U., Masci, J., Gambardella, M. L., Schmidhuber, J., Flexible, High-Performance Convolutional Neural Networks for Image Classification, In proceedings of Twenty-Second International Joint Conference on Artificial Intelligence, 1237-1242, 2011.

[21] Ciresan, D., Meier, U. and Schmidhuber, J. (2012) "Multi-column Deep Neural Networks for Image Classification." IEEE Conference on Computer Vision and Pattern Recognition 3642-3649.

[22] Niu, X. X. and Suen, C. Y. (2012) "A novel hybrid CNN–SVM classifier for recognizing handwritten digits", Pattern Recognition 45(4):1318–1325.

[23]. Ghosh MMA, Maghari AY (2017) A comparative study on handwriting digit recognition using neural networks. In: International conference on promising electronic technologies, pp 7781

[24]. Hamid, N.A.; Sjarif, N.N.A. Handwritten recognition using SVM, KNN and neural network. arXiv 2017, arXiv:1702.00723.

[25] K. Bansal and R. G. Kumar, "Cleaning and Recognition of Numerals in a Handwritten Devnagari Document Consisting of Roll Numbers and Marks," PhD Thesis, 2012.

[26] M. TSABET, B. BOUMAAD, “Arabic handwriting recognition using Curvelet transform and SVM”, Master Thesis, Faculty Electrical and Electronic Engineering, University M’Hamed BOUGARA – Boumerdes, 2018.

[27] M. Cheriet, R. F. Moghaddam, and R. Hedjam, “A learning framework for the optimization and automation of document binarization methods,” Comput. Vis. Image Underst., vol. 117, no. 3, pp. 269–280, 2013.

[28] Maged Mohamed Mahmoud Fahmy, S.Al Ali : « Automatic recognition of handwritten Arabic characters using their geometrical features ». Studies in informatics and control journal (SIC journal), vol. 10, No 2, 2001.

[29] N. Otsu, “A threshold selection method from gray-level histograms,” IEEE Trans. Syst., Man, Cybern., vol. SMC-9, no. 1, pp. 62–66, Jan. 1979.

[30] J. Kittler, J. Illingworth, and J. Föglein, “Threshold selection based on a simple image statistic,” Comput. Vis., Graph., Image Process., vol. 30,no. 2, pp. 125–147, 1985.

[31] Otsu, “A thresholding selection method from gray-scale histogram,”IEEE Transactions on System, Man, and Cybernetics 9 (1979) 62–66.

[32] W. Niblack, An Introduction to Digital Image Processing, vol. 34. Upper Saddle River, NJ, USA: Prentice-Hall, 1986.

[33] L. Jiang, K. Chen, S. Yan, Y. Zhou, and H. Guan, “Adaptive Binarization for Degraded Document Images,”IEEE, pp. 5–8, 2009.

[34] J. Sauvola and M. Pietikäinen, “Adaptive document image binarization,” Pattern Recognit., vol. 33, no. 2, pp. 225–236, 2000.

[35] J. Bernsen, “Dynamic thresholding of gray level images,”Proceedings of International Conference on Pattern Recognition (ICPR), pp. 1251–1255, 1986.

[36] K. Khurshid, I. Siddiqi, C. Faure, N. Vincent, “Comparison of Niblack inspired Binarization methods for ancient documents,”16th International conference on Document Recognition and Retrieval, USA, 2009.

[37] X. Peng, S. Setlur, V. Govindaraju, and R. Sitaram, “Markov random field based binarization for hand-held devices captured document images,” in Proc. 7th Indian Conf. Comput. Vis., Graph. Image Process.,2010, pp. 71–76.

[38] N. Papamarkos, “A neuro-fuzzy technique for document binarisation,” Neural Comput. Appl., vol. 12, nos. 3–4, pp. 190–199, 2003.

[39] C. Tensmeyer and T. Martinez. (2017). “Document image binarization with fully convolutional neural networks.” [Online]. Available: https://arxiv.org/abs/1708.03276.

[40] “Skeletonization Algorithm for Binary Images - ScienceDirect.” [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212017313004027. [Accessed: 15-Jan2019].

[41] P.Kardos, G.Nemeth, K.Palagyi , ‘An Order–Independent Sequential Thinning Algorithm’ , P.Wiederhold and R.P. Barneva (Eds.): IWCIA 2009, LNCS 5852, pp. 162–175, 2009.

[42] L.Lam, Ching Y. Suen, ‘An Evaluation of Parallel Thinning Algorithms for Character Recognition’, 0162-8828/95$04.00 0 , VOL 17, NO. 9. SEPTEMBER 1995 IEEE.

[43] M. TABEDZKI, M. RYBNIK , K.SAEED, M .ADAMSKI, " A UNIVERSAL ALGORITHM FOR IMAGE SKELETONIZATION AND A REVIEW OF THINNING TECHNIQUES ", DOI:10.2478/v10006-010-0024-4, Int. J. Appl. Math. Comput. Sci., 2010, Vol. 20, No. 2, 317–335.

[44] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” Phys. Nonlinear Phenom., vol. 60, no. 1–4, pp. 259–268, 1992.

[45] H. Kusetogullari, A. Yavariabdi, Evolutionary multiobjective multiple description wavelet based image coding in the presence of mixed noise in images, Applied Soft Computing Volume 73, Pages 1039-1052, 2018.

[46] B. Scholkopf et al., “Comparing support vector machines with Gaussian kernels to radial basis function classifiers,” IEEE Trans. Signal Process., vol. 45, no. 11, pp. 2758–2765, 1997.

[47] M.Bergounioux, “Quelques méthodes de filtrage en Traitement d’Image”, HAL Id: hal-00512280

https://hal.archives-ouvertes.fr/hal-00512280v1

[48] M .T.Chikh, « AMÉLIORATION DES IMAGES PAR UN MODÈLE DE RÉSEAU DE NEURONES (COMPARAISON AVEC LES FILTRES DE BASE) », université Aboubakr belkaid Tlemcen 2011.

[49] A. Bagdanov and J. Kanai, “Projection profile based skew estimation algorithm for JBIG compressed images,” in Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on, 1997, vol. 1, pp. 401–405.

[50] T. Greenhalgh, G. Robert, F. Macfarlane, P. Bate, O. Kyriakidou, and R. Peacock, “Storylines of research in diffusion of innovation: a meta-narrative approach to systematic review,” Soc. Sci. Med., vol. 61, no. 2, pp. 417–430, 2005.

[51] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa, “Handwritten digit recognition: investigation of normalization and feature extraction techniques,” Pattern Recognit., vol. 37, no. 2, pp. 265–279, 2004.

[52] S. Kahan, T. Pavlidis, and H. S. Baird, “On the recognition of printed characters of any font and size,” IEEE Trans. Pattern Anal. Mach. Intell., no. 2, pp. 274–288, 1987.

[53] Vazquez, F. A., & Marin, R. (1992)." Pattern Recognition Applied to Formatted Input of Handwritten Digits". IFAC Proceedings Volumes, 25(6), 361–364.

[54]https://medium.com/analytics-vidhya/deep-dive-into-convolutional-neural-networks800a7fdf9fd9

[56] Saha, Roshni, (2019). "Classification of Parkinson’s Disease Using MRI Data and Deep Learning Convolution Neural Networks" Creative Components. 241.

[57] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15, 1929-1958.

[58] Jonghyun Yu, Taeil Hur, Jaeho Jung, and In Gwun Jang, (2018), Deep learning for determining a near-optimal topological design without any iteration, Article in Structural and Multidisciplinary Optimization .

[59] Md. Lizur Rahman, Efrat Jahan, Akash Saha, Md. Nawab Yousuf Ali, (2018), Efficient Recognition of Bangla Handwritten Digits Based on Deep Neural Network, International Journal of Recent Technology and Engineering (IJRTE) , Volume-7 Issue-4.

[60] https://pavisj.medium.com/convolutions-and-backpropagations-46026a8f5d2c

[61] https://www.kaggle.com/

[62] Wikipedia, https://en.wikipedia.org/wiki/Kaggle , consulted on: 11/05/2022.

[63] Martin Abadi et el. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[64] F.Chollet et al .”Keras,” https://github.com/fachollet/keras,2015

APPENDIX

Start up (The realization of an automatic teller machine that recognizes handwritten checks)

1 Introduction

Recently, what is known as an automated teller machine (ATM) has appeared in Algeria using electronic cards, which covered most of the problems of financial institutions, but a significant group of customers remains that uses traditional techniques such as postal checks in financial transactions, and this is what we see now in most post offices of overcrowding of customers and long queues.

Starting from this problem, we proposed a system for reading checks using the system for recognizing handwritten numbers, which is the subject of our study in this note.

2 Business Model Canvas

2.1 Customer Segments

2.1.1 Main slide

- Algeria post.

2.1.2 Secondary slide

- Banks, financial institutions that use chekes in their financial transactions.
- Institutions from the private sector that use electronic payment.

2.2 Customer Relationships

- Direct communication with the client.
- Special Personal Assistance.

2.3 Distribution Channels

Direct

- Communicate personally with all the customer needs from device installation and maintenance.

Indirect

- Marketing Agents.

2.4 Value Propositions

- Through this device, the checks reader enables us to deposit, withdraw, transfer, pay gas, electricity, water, mobile phone sim bills, and purchase goods and services for financial transactions.
- It contains a fingerprint scanner.
- Night camera for facial recognition.
- Scanner for checks.
- An automated system that recognizes handwriting and printed font.
- National ID card reader.
- Voice interactions with the device.

2.5 Key Activities

Gathering the necessary data.

System programming.

Device equipment installation.

The combination of system and hardware, and finally marketing.

2.6 Key Resources

Human resources: workers and engineers

Material Resources:

- Device user data (fingerprints, face photos).
- Thermal camera.
- Scanner.
- Interactive screen with side buttons.
- fingerprint reader.
- CPU.
- National ID Card Reader.

2.7 Key Partners

- Emerging startup Fund.
- The municipality's digital biometric data preservation services to obtain a data set of fingerprints and facial images.

2.8 Revenue Streams

- Subscription fee for using the device.
- sell the device.

2.9 Cost Structure

Direct: central processing units, electronic suffixes, distribution, marketing, workers.

Indirect: Electricity, water, phone bill, startup company center rent.

الملخص

الغرض الرئيسي من هذه الأطروحة هو بناء طريقة التعرف التلقائي على الأرقام المكتوبة بخط اليد. لإنجاز مهمة التعرف ، أولاً ، يتم تقديم مجموعة البيانات كمدخلات, يتبع ذلك معالجة مسبقة, حيث تخضع الصورة لعمليات مختلفة مثل تقليل الضوضاء, وتصحيح انحراف الأرقام, وتصحيح الميل, والتطبيع, والتنعيم, والهيكل العظمي. يمكن إعطاء نتيجة هذه المرحلة كمدخل لمرحلة إستخراج خريطة الميزات تلقائيًا من الصور بإستخدام 3 طبقات من الشبكات العصبية التلافيفية (CNN), ثم يتم تغذيتها ب4 طبقات من مصنف عصبي إصطناعي FC ANN حيث يصنف هاته الصور إلى عشرة فئات (من 0 الى 9), تعد دقة النموذج النهائي مهمة لأن النماذج الأكثر دقة تتخذ قرارات أفضل, أظهرت نتائجنا دقة تصنيف 99.78% وقيمة خسارة 1.82% إعتمادا على مجموعة بيانات MNIST .

الكلمات المفتاحية: الشبكات العصبية التلافيفية؛ MNIST؛ الميزات المستخرجة؛ المعالجة المسبقة؛ الشبكات العصبية الاصطناعية.

ABSTRACT

The main purpose of this thesis is to build a method for automatic recognition of handwritten digits. To accomplish the recognition task, first, the data set is presented as input, this is followed by a preprocessing phase, where the image undergoes various operations such as noise reduction, normalization, smoothing, and skeletonization. The result of this stage can be given as input to the stage of automatically extracting feature maps from images using 3 layers of Convolutional Neural Networks (CNN), and then fed to 4 layers of FC ANN, where these images are classified into ten classes (0 to 9). The accuracy of the final model is important because more accurate models make better decisions. Our results showed a classification accuracy of 99.78% and a loss value of 1.82% using the MNIST data set.

Key words: Convolutional Neural Networks; MNIST; features extraction; preprocessing; Artificial Neural Networks.

Résumé

L'objectif principal de cette thèse est de construire une méthode de reconnaissance automatique des chiffres manuscrits. Pour accomplir la tâche de reconnaissance, tout d'abord, l'ensemble de données est présenté en entrée, suivi d'un prétraitement, où l'image subit diverses opérations telles que la réduction du bruit, la correction de l'aberration numérique, la correction de l'inclinaison, la normalisation, le lissage et la squelettisation. Le résultat de cette étape peut être donné en entrée à l'étape d'extraction automatique des cartes de caractéristiques à partir d'images à l'aide de 3 couches de réseaux de neurones convolutifs (CNN), puis transmis à 4 couches de FC ANN, où ces images sont classées en dix classes (0 à 9), dont la précision est Le modèle final est important car des modèles plus précis prennent de meilleures décisions. Nos résultats ont montré une précision de classification de 99,78 % et une valeur de perte de 1,82 % sur la base de l'ensemble de données MNIST.

Mots clés : réseaux de neurones convolutifs ; MNIST ; caractéristiques extraites ; prétraitement ; Réseaux de neurones artificiels.