PEOPLE DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

YAHIA FARES UNIVERSITY OF MEDEA



Faculty of **T**echnology

Department of **E**lectrical **E**ngineering

Master Thesis

Sector: Electronics

Specialty: Electronics of Embedded Systems

Realization of an intelligent check reader written in Arabic

Presented by:

Mr. BENHADJER Mosaab

Proposed and Directed by:

Dr. KARA Reda

Dr. BENYEZZA Hamza

Promotion June 2022

Work prepared within the Laboratory of Advanced Electronic Systems (LSEA)
of the University of Medea, ALGERIA

Dedication

بسم الله الرحمن الرحيم والحمد لله

I dedicate this project thesis to my family and many friends. A special feeling of gratitude to my loving parents, who tirelessly encouraged me and counseled me, and to all my siblings who have never left my side and are very special.

I also dedicate this project thesis to my friends who have supported me throughout this journey. I will always appreciate their support and all the things they have done.

I dedicate this work and give special thanks to my best friends for being there for me throughout the Master's program. All of you have been my best cheerleaders.

Above all, I dedicate this project thesis to the Almighty God above. Finishing this manuscript is a great opportunity to express my whole-hearted thanks and many praises unto him.

Mosaab BENHADJER

Acknowledgement

بسم الله الرحمن الرحيم والحمد لله

I would like to thank my dear teacher and supervisor Dr. Kara as well as my co-supervisor Dr. Benyezza for supporting me throughout this tough year and kept motivating me to give the best I could.

Thank you to the jury members for giving me the honor of judging my work. I also sincerely thank all the Electronics engineering department teachers.

Finally, I thank my family, friends, colleagues and all the class of 2022

ملخص:

تتعامل المؤسسات المالية اليوم مع التكاليف المتزايدة لعبء التنظيم والإمتثال؛ لذلك فهم بحاجة إلى موظفين مدربين للحفاظ على الإمتثال. لكن هؤلاء الموظفون كبشر مسؤولون عن التحقق من الشيك يدويا قبل تسليم الأموال إلى العميل، مما سيستغرق وقتًا طويلاً نسبيا وقد يتسبب في ارتكابهم لأخطاء، وخاصةً عندما يكونون مشغولين بالزحام فتجدهم يسارعون إلى مواكبة ذلك.

كحل لمشكلة الفعالية، تم اقتراح نظام ذكي أوتوماتيكي للتحقق من الصكوك والذي يمكنه القيام بواجبات موظفي المصرف في وقت أقصر كما سيساعد في تقليل الخطأ البشري وبالتالي زيادة أرباح المؤسسات المالية. يتكون هذا النظام من جزأين: جزء برمجي وجزء المعدات. يتكون الجزء البرمجي من نظام للتحقق من الصكوك يعتمد على أربعة شبكات عصبية إلتفافية، بينما يتكون جزء المعدات من كاميرا متصلة عن بعد بوحدة معالجة.

عندما يريد شخص يحمل صك مملوئا سحب المال من البنك، يجب عليه أولاً وضع الصك فوق الشباك والضغط على زر البدأ، بذلك سيتم أخذ صورة للصك إلى نظام التحقق لمعاينة الرقم المعرف للصك، التوقيع، المبلغ المراد سحبه مكتوبا بالصيغة الرقمية والعددية والتي سيتم التحقق منها لتأكيد قبول التحقق من الصك.

الكلمات المفتاحية: الشبكات العصبية الإتفافية، الصك، Raspberry، التوقيع، الأرقام المكتوبة بخط اليد، الكلمات المكتوبة بخط اليد باللغة العربية.

Résumé:

Les institutions financières d'aujourd'hui sont confrontées à des coûts accrus liés au fardeau réglementaire et à la conformité ; par conséquent, ils ont besoin d'un employé formé pour maintenir la conformité. En tant qu'êtres humains, ces employés sont chargés de compter l'argent manuellement plusieurs fois avant de le remettre à un client, ce qui prend du temps et peut leur faire commettre des erreurs, en particulier lorsqu'ils sont occupés et se précipitent pour suivre le rythme.

En tant que solution aux problèmes d'efficacité, un système intelligent qui automatisée la vérification des chèques, aussi peut refaire les fonctions de guichet dans un délai plus court et aider à réduire l'erreur humaine, augmentant ainsi la rentabilité des institutions financières. Ce système est composé de deux parties : logiciel et matériel. La partie logicielle consiste un système de vérification des chèques basé sur quatre CNN différents, tandis que le matériel consiste en une caméra connectée à distance à une unité de traitement appelée Raspberry PI 3 Model B.

Lorsqu'une personne avec un chèque rempli veut retirer de l'argent, elle doit d'abord le mettre sur le comptoir de guichet et appuyer sur un bouton poussoir, ainsi une photo de son chèque sera prise à notre système de vérification pour traiter : le numéro d'identification du chèque, la signature , le montant

d'argent en chiffres et en format texte, ce sera vérifié pour confirmer l'acceptation de la vérification du chèque.

Mots-clés: CNN, Chèque, Raspberry, Signature, Numéro manuscrit, Mots manuscrits en langue arabe.

Abstract:

Today's financial institutions are dealing with increased costs for regulatory burden and compliance; as a result, they require a trained staff to maintain compliance. As humans, these employees are responsible for verifying a check manually before handing cash to a client. It is time consuming and may cause them to make mistakes, especially when they are busy and rushing to keep up.

As a solution to the efficiency issues, a smart automated check verification system that can replicate the teller line duties in a shorter time and reduce human error, thus increasing the profitability of the financial institutions, has been proposed. This system is comprised of two parts: software and hardware. The software consists of a check verification system based on four different CNNs, while the hardware consists of a camera liked remotely to a processing unit called Raspberry PI 3 Model B.

When a person with a filled check wants to withdraw money, he must first put it on the cash counter and press on a push button, so a picture of his check will be taken to our verification system to process: the check ID number, signature, amount of money in digits and text formats, that will be verified to confirm the check verification acceptance.

Keywords: CNN, Check, Raspberry, Signature, Handwritten numbers, Handwritten words in Arabic language.

Table of Contents

List of Acronyms I

List of Figures III

List of Tables V

General Introduction VI

Chapter I : Document Verification 1

I.1 Introduction 2

I.2 Evolution of Document Verification 2

I.2.1 Definition of Document Verification 3

I.2.2 Comparing Digital and Traditional Document Verification 4

I.2.3 Digital Document Verification Methods 5

I.2.4 Machine Learning Influence on Document Verification 6

I.3 Check Verification 6

I.3.1 History 6

I.3.2 Check Definition 7

I.3.3 Check Features 7

I.3.4 Check Types 8

I.3.5 Problematic 10

I.3.6 Check Advantages 10

I.4 Conclusion 11

Chapter II : Artificial Intelligence 12

II.1 Introduction 13

II.2 Basic Concepts of Machine Learning 13

II.2.1 Machine Learning Definition 13

II.2.2 Machine Learning Types 13

II.2.3 Supervised ML Learning Algorithms 17
II.2.4 Evaluation of a Binary Classifier 18
II.3 Artificial Neural Networks 21
II.3.1 ANNs Definition 22
II.3.2 Layers in Artificial Neural Network 22
II.3.3 Structure of Neural Networks 23
II.3.4 Feedforward Neural Networks 25
II.3.5 ANN Learning 26
II.4 Convolutional Neural Networks 28
II.4.1 Convolution operation 28
II.4.2 Layers in CNN 30
II.4.3 Performance optimization of CNN 36
II.4.4 CIFAR CNN Example 39
II.5 Conclusion 44
Chapter III : Materials Used 45
III.1 Introduction 46
III.2 Hardware Used 46
III.2.1 Raspberry pi 3 Model B 46
III.3 Software Used 47
III.3.1 Python 47
III.3.2 TensorFlow 48
III.3.3 Keras 48
III.3.4 OpenCV 49
III.3.5 SQLite 49
III.3.6 Tkinter 50
III.3.7 Jupyter and Google Collaboratory 50

III.3.8 IP Webcam 51
III.4 Conclusion 51
Chapter IV : Implementation, Tests and Results 52
IV.1 Introduction 53
IV.2 Specification Book 53
IV.2.1 Purpose 53
IV.2.2 Product Features 53
IV.2.3 Interfaces Requirements 53
IV.3 Global System Flow Diagram 54
IV.4 Check Verification Model 55
IV.4.1 ID Number Verification CNN 56
IV.4.2 Signature Verification CNN 59
IV.4.3 Money Number Verification CNN 64
IV.4.4 Money Text Verification CNN 66
IV.4.5 Discussion 70
IV.5 Implementation 71
IV.5.1 Administration Graphical User Interface 72
IV.5.2 Check Verification 73
IV.5.3 Hardware Implementation 80
IV.5.4 Final Product Assembly 81
IV.5.5 Results 81
IV.6 Conclusion 83
General Conclusion 85
Bibliography 86

List of Acronyms

ACC: Accuracy

AI: Artificial Intelligence

ANN: Artificial Neural Network

API: Application Programming Interface

BLE: Bluetooth Low Energy

CIFAR: Canadian Institute for Advanced Research

CNN: Convolutional Neural Network

CPU: Central Processing Unit

CSI: Camera Serial Interface

DSI: Display Serial Interface

FN: False Negative

FP: False Positive

FPR: False Positive Rate

GPIO: General-purpose Input/Output

GUI: Graphical User Interface

HDMI: High-Definition Multimedia Interface

ID: Identification

IP: Internet Protocol

KNN: K-nearest Neighbor

LAN: Local Area Network

LED: Light-Emitting Diode

MAE: Mean Absolute Error

ML: Machine Learning

MSE: Mean Squared Error

NFC: Near Field Communication

NN: Neural Network

NSF: Non-sufficient Funds

PCA: Postal Check Account

PPV: Positive Predictive Value

RAM: Random Access Memory

ReLU: Rectified Linear Unit

RGB: Red, Green, Blue

RL: Reinforcement Learning

ROC: Receiver Operating Characteristic

SD: Secure Digital

SGD: Stochastic Gradient Descent

SQL: Structured Query Language

SVM: Support Vector Machine

TN: True Negative

TP: True Positive

TPR: True Positive Rate

USB: Universal Serial Bus

VGG: Visual Geometry Group

List of Figures

Figure I-1 Algeria Poste Check 8
Figure I-2 Algerian Post Money Order 9
Figure II-1 Supervised learning algorithm in features extraction [19] 14
Figure II-2 Unsupervised learning algorithm in features extraction [19] 15
Figure II-3 Machine learning Algorithms [22] 16
Figure II-4 Confusion Matrix 19
Figure II-5 A basic ROC Graph [26] 20
Figure II-6 Some theoretical precision-recall curves 21
Figure II-7 Structure of Artificial Neural Network [30] 23
Figure II-8 A simple CNN architecture [38] 30
Figure II-9 Feature maps after convolution filtering [39] 31
Figure II-10 CIFAR CNN Architecture #1 39
Figure II-11 CIFAR CNN Architecture #2 40
Figure II-12 CIFAR CNN Training Loss and Accuracy Results 42
Figure II-13 CIFAR Model Confusion Matrix 43
Figure III-1 Raspberry Pi 3 Model B [53] 46
Figure III-2 Python Programming Language Logo 47
Figure III-3 TensorFlow Logo 48
Figure III-4 Keras Logo 49
Figure III-5 OpenCv Logo 49
Figure III-6 SQLite Logo 50
Figure III-7 IP Webcam Android App 51
Figure IV-1 Global System Flow Diagram 55
Figure IV-2 ID Number CNN Training Flow Diagram 56
Figure IV-3 ID Number CNN Dataset Sample Image After Pre-Processing 57
Figure IV-4 ID Number CNN Validation Loss and Accuracy Results 58
Figure IV-5 ID Number Model Confusion Matrix 59
Figure IV-6 Signature CNN Training Flow Diagram 59
Figure IV-7 SQLite based Users Database 60
Figure IV-8 Signature Sample Images after Data Augmentation 61
Figure IV-9 Signature Training Loss and Accuracy Results 63

Figure IV-10 Money Number CNN Training Flow Diagram 64
Figure IV-11 Money Number Dataset Sample Images after Pre-Processing 64
Figure IV-12 Money Number Confusion Matrix for architecture #4 66
Figure IV-13 Money Text CNN Training Flow Diagram 67
Figure IV-14 Money Text Dataset Sample Images after Pre-Processing 68
Figure IV-15 Money Text Training Loss and Accuracy Results 70
Figure IV-16 Administration Graphical User Interface 73
Figure IV-17 Check Segmentation Part 1 73
Figure IV-18 Check Segmentation Part 2 74
Figure IV-19 Check Segmentation Part 3 74
Figure IV-20 Check Segmentation Part 4 75
Figure IV-21 Check Segmentation Part 5 75
Figure IV-22 Check Segmentation Part 6 75
Figure IV-23 Check Segmentation Part 7 76
Figure IV-24 Check Segmentation Part 8 76
Figure IV-25 Detailed Flowchart of the Check Verification System Part 1 77
Figure IV-26 Detailed Flowchart of the Check Verification System Part 2 78
Figure IV-27 Detailed Flowchart of the Check Verification System Part 3 79
Figure IV-28Electrical wiring diagram of our system 80
Figure IV-29 Assembled Check Verification System 81
Figure IV-30 Check Segmentation Progress 82
Figure IV-31 ID Number Verification Progress 82
Figure IV-32 Signature Verification Progress 82
Figure IV-33 Money Number Identification Progress 82
Figure IV-34 Money Text Identification Progress 83

List of Tables

Table IV-1 ID Number CNN Configuration 58
Table IV-2 Signature CNN Configurations 62
Table IV-3 Money Number CNN Configuration 65
Table IV-4 Money Text CNN Configurations 69
Table IV-5 Comparison between the obtained CNN results 71
Table IV-6 Check Verification System Execution Time 83

General Introduction

Nowadays, the financial institutions and their client still suffer from the classic way of handling regulatory burdens and compliance by human employees (Teller line) due to the human mistakes and work speed which affect the financial institution's profitability and the client user experience negatively. For this reason, several approaches were implemented using Debit and Credit cards for withdrawing money, but since the paper check still has its advantages over the Debit or Credit cards, our approach come to solve that problem is by keeping the paper check part but replacing the human employees with an intelligent automated check verification system.

Document verification systems are already implemented in some sectors like the Art Museum sector in which the old manuscripts are verified; companies like google are investing heavily on this document verification systems especially for digital document verification and they have a bunch of publications and patents about that particular subject [1] [2] [3].

In this Master thesis, we will propose an intelligent check reader written in Arabic that will automate the teller line, thus increasing the profitability of the financial institutions by decreasing the costs associated with the additional staff necessary to maintain compliance, improve the customer experience by speeding up the transaction process while doing it more accurately since the human factor will be eliminated.

This thesis is made of four chapters. The first one is a presentation of the state of the art of document verification and checks verification. In the next chapter, we will dive into the world of artificial intelligence; from ML concepts to ANN to CNN architectures, with a practical example in the end. The third chapter will be a general description of the different components used in this project. The fourth chapter is dedicated to the presentation of our system from implementations to tests to results.

Chapter I : Document Verification

I.1 Introduction

Documents are an integral part of our lives. Without a passport or an ID card we would not be able to go on with our lives. Our functioning in today's world would be impossible. Due to their importance, documents are verified by humans or machines to prove document ownership. For example: verifying that financial statements, passports, licenses, utility bills, and certifications are real and belong to the applicant who submitted them.

The goal is to determine if the document presented (e.g., a bank check) is authentic or not by comparing it to official records, which is an easy task for a trained human but may take quite a long time.

Is it the same for a machine? In this chapter, we will present the main lines of our work. Firstly, we take a general idea of document verification by introducing its evolution, set an up-to-date definition of it, compare the traditional and modern document verification, and highlight the influence of machine learning on modern document verification. Secondly, we are going to narrow our focus on a paper check by exploring its history, then explaining it plus mentioning its features and types. Finally, we are going to highlight paper check advantages nowadays by mentioning what should be enhanced for it to adapt with our digital age.

I.2 Evolution of Document Verification

There is a revolution underway in the secured document field, as society migrates from using physical secured documents, such as banknotes and identity cards, to using smartphones and electronic payment cards for financial transactions and as carriers of our identity credentials.

There is a perception that this revolution is well underway and that the transition from physical to digital is inevitable, unstoppable and irrevocable. The drivers are user convenience and provider cost savings, allied to the technical community's belief that it can deliver a new way of doing things.

Physical documents are tangible, familiar, and have built-in security and authentication features. Moreover, there are a thousand years of history and experience in the world of banknotes, passports and other secured documents, and a key driver for specifiers and designers is security and document protection.

Digital financial transaction systems and identity credentials, for comparison, have barely a 20-year history. This field is technology driven, starting with the development of near field communication (NFC) chips on phones and the 2004 launch of Alipay in China, followed by the launch of the iPhone in 2007.

Identity becomes a critical factor in this move to digital systems, requiring users to establish their identity and then repeatedly prove it. This involves biometrics, encrypted digital signatures, hash codes, two-factor signatures, self-sovereign ID, encrypted apps, encrypted phones and much more. But all of this depends on the algorithms in the systems.

A Bank teller can look at and verify a check and look at the person handing it – but this takes time. When digital solutions are created, the primary goal is a more convenient product; Security is required too but is currently a secondary consideration in the development of these digital systems.

Digital identity is being introduced across health systems, driver's licenses and national ID cards, facilitating more efficient health care and easing the passenger experience, enabling kerb-to-seat progression without showing any documents at an airport. Significantly, though, no immigration agency allows the same convenience for arriving international passengers, who all have to show a passport or ID card, sometimes to a machine, sometimes to a person, but always with the person there as backup to the machine [4].

I.2.1 Definition of Document Verification

The move to digital and online documents has, in many ways, distanced us from more document verification forms like picture IDs and checks. While that isn't necessarily a wrong step, there are also incredible benefits of using authentic, official documents as part of the verification process.

Providers have begun focusing on verification as a critical part of the authentication and identification process to combine the benefits of physical document verification with digital platforms.

Document verification is the process of confirming the authenticity of a document through specific characteristics like a signature [5].

A document verification procedure comprises three fundamental steps, which are listed below [6]:

- **Submitting User Information:** The document verification solution prompts the user to submit a photo of their physical document by scanning it through their webcam or smartphone camera. The system at the back-end receives input from the user in the form of an image of the document with their photo or signature on it.
- **Evaluation of Credentials:** The user submissions are put through a series of checks to determine if the information is genuine, and the document is original and not fake, compromised or altered.
- **Verification and Results:** Upon successful evaluation of user credentials, the results regarding the validation of the submitted document are shared with the client.

I.2.2 Comparing Digital and Traditional Document Verification

Traditional forms of document verification were time-consuming and tedious. They slowed down processes and frustrated customers.

Before digitization, customers visited local businesses or branches to submit applications and documents. They were verified in person with the individual present, making matching ID photos with the face easier. However, verifying documents wasn't as fast, in some cases taking weeks to process.

Identifying fraudulent documents was challenging since fraudsters had printers and skills to create counterfeit documents (e.g., bank statements and other financial paperwork). So, ensuring the documents used were 100% valid was next to impossible.

Modern document verification technology makes this possible. These tools use artificial intelligence and machine learning to identify documents, authenticate them using a database of original documents, and parse personal details (name, address, etc.).

Using AI tools to verify documents speeds up processes and ensures regulatory compliance (not to mention enhances fraud detection, which can save institutions millions in losses) [6].

I.2.3 Digital Document Verification Methods

Digital document verification works through a series of potential validation checks that scan a document and determine its authenticity while comparing it against information within a database. Some of the validation checks can include the following [5]:

I.2.3.1 Facial Detection

Documents with a face photo provide critical information for identity verification. These identity verification systems will often be able to pull image data from the document and compare it with internal photos.

I.2.3.2 Data Correlation

Many official forms of documents have linked information on both front and back sides. An authentication solution must correlate that data to guarantee that the document is legitimate and that it references a given user.

I.2.3.3 Document Scanning and Fraud Detection

Critical to some documents is to have mechanisms in place to scan documents for critical information, including signatures, ID numbers, barcodes, etc., and link them with database information. Additionally, these scanners will notice fraudulent documents through watermarks or other forms of pattern recognition.

I.2.3.4 Database Checks

Depending on the provider, the scanned document will be compared against information related to thousands of pre-taken images in a database to authenticate the document.

That’s why many verification solutions will boast how many thousands of document types they can verify—more documents often mean wider coverage for international users.

I.2.3.5 Live Selfie Checks

As part of an advanced biometrics and verification, these solutions will often deploy image checks against user selfies from mobile devices. Not only will this support immediate image comparisons against IDs to connect biometrics and document verification, but will also include liveness testing to authenticate that the right user is present and prevent spoofing.

I.2.4 Machine Learning Influence on Document Verification

Most of the problems that stem from online document verification come from inaccuracies related to scanning or document authentication. Firsts digital verification methods relied on an algorithmic understanding of documents and patterns in images, which could not compete with a human inspection. Machine learning has shifted those issues and made online verification more accurate and dependable.

Machine learning can support more advanced online document scanning and optical character recognition to notice fraudulent credentials. More importantly, machine learning tools can learn through practice. Data scientists can feed real and fake documents worldwide into an intelligent verification system to train that system to catch more advanced forgeries without sacrificing performance [5].

I.3 Check Verification

Paper checks use have declined since the early 2000s, but a large portion of the population still relies on these physical check transactions.

Nowadays, consumers want to withdraw money using checks without waiting for a long time — which is why some of them shift to using digital transactions because they are more convenient and fast-paced.

Since checks have some unique advantages compared to debit and credit cards, why not add to their advantages by automating the way checks are verified to withdraw money? [7]

I.3.1 History

Checks have been in existence in one form or another since ancient times. Many people believe a check was used among the ancient Romans [8]. While each culture that adopted a form of check had its system, they all shared the basic idea of substituting the check for physical currency.

In 1717 the Bank of England was the first organization to issue preprinted checks [9]. The oldest American check dates to the 1790s [10].

Modern checks, as we know them today, became popular in the 20th century. Check usage surged in the 1950s as the check process became automated and machines were able to sort and

clear checks. Check cards, first created in the 1960s, were the precursors to today's debit cards [11]. Credit and debit cards—and other forms of electronic payment—have since overshadowed checks as the dominant means of paying for most goods and services. Checks are now somewhat uncommon but still used among the general population.

I.3.2 Check Definition

A check is a written, dated, and signed instrument that directs a bank to pay a specific sum of money to the bearer. The person or entity writing the check is known as the payor or drawer, while the person to whom the check is written is the payee. On the other hand, the drawee is the bank on which the check is drawn.

Checks may be cashed or deposited. When the payee presents a check to a bank or other financial institution to negotiate, the funds are drawn from the payor's bank account. It is another way to instruct the bank to transfer funds from the payor's account to the payee or the payee's account. Checks are generally written against a checking account, but they can also be used to negotiate funds from a savings or other type of account [12].

I.3.3 Check Features

While not all checks look alike, they generally share the same key parts. The name and contact information of the person writing the check is located at the bottom center of the check. The name of the bank that holds the drawer's account appears on the check as well.

There are a number of lines that need to be filled in by the payor:

- The amount of the check in a DA figure is filled out in the box on the top right-hand corner of the check.
- The amount written out in words goes on the first line in the center of the check. This is indicated by the phrase "Pay against this check."
- The payee's name goes on the line underneath the amount written out in words.
- The date and the Current postal center location is written on the line underneath the payee's name.
- The payor signs the check on the line on the bottom right-hand corner of the check. The check must be signed to be considered valid.



Figure I-1 Algeria Poste Check

Figure I-1 shows a series of coded numbers found along the bottom edge of the check, underneath the payor's signature. These numbers represent the bank's identifying code and the payor's account number. The check number is on the top left-hand corner of the check [12].

I.3.4 Check Types

There are many kinds of checks that come from different financial providers and offer many different kinds of useful features, the common ones are [13]:

I.3.4.1 Personal Check

A personal check has the user routing and account number printed on it and it is issued by the bank. The payor needs to fill out the check, since it is otherwise blank. The payee will then deposit the check in their account, which transfers funds to the recipient.

I.3.4.2 Certified Check

A certified check verifies that there are enough funds in the drawer's account to honor the amount of the check. This means that the check is guaranteed not to bounce due to insufficient funds. A check can be certified at the bank from which it is drawn.

I.3.4.3 Cashier's Check

A bank guarantees these types of checks. A cashier signs the check, which makes the bank responsible for the funds. A cashier's check is generally required in large transactions, such as buying a car or house. Most banks and credit unions will charge a check fee for cashier's checks.

I.3.4.4 Payroll Check

A payroll check is another type of check; however, it is issued by an employer to compensate for the employee's work. Direct deposit and other forms of electronic transfers have overtaken the traditional, physical paycheck.

I.3.4.5 Bounced Check

As mentioned, a check can to bounce if a bank has not guaranteed it. This means the payor wrote a check for an amount larger than what is in their account. The payor has non-sufficient funds (NSF), and will likely incur a fee for attempting to overdraw on the account.

I.3.4.6 Voided Check

A voided check is a paper check that has the word "VOID" written across the front of it.

I.3.4.7 Money Order

A money order is a secure form of payment, representing an amount of money. The user will pay the full amount of the check, plus a small fee when he purchases it. Unlike checks, a money order is not directly pulled from the user's account. They are useful if he may not want to use a personal check which includes personal information.



Figure I-2 Algerian Post Money Order

I.3.5 Problematic

With most businesses, people in charge of operations are typically tasked with finding ways to become more efficient and handle manual processes better and more streamlined way. One of the most intensive things that happen at a financial institution is the process of handling cash. At a financial institution, that is typically what the teller line handles every day. They are handing people cash out of their accounts, either when a customer cashes a check or even with a simple withdrawal from their account. They could also be counting and authenticating large amounts of mixed cash from retail and commercial customers. Also, they may have to make change orders for yet other sets of customers.

The amount of time and labour associated with those tasks and the number of mistakes that can happen by doing these processes manually is significant. When you have additional labour and the need to involve other departments of the bank to help track down the mistakes of doing things manually, it becomes a goal of the institution to look at these areas and ask "How can we automate the teller line?"

I.3.6 Check Advantages

Many people still use check in the digital era since they are [14]:

I.3.6.1 Safer to Mail

Sending a check in the mail keeps the user's money a lot safer than sending cash through the mail. When the client has a check inside an envelope it can be very difficult for others to tell there is even a check inside.

But if the user put cash in an envelope, it can be much easier for others to tell there is money inside since bills have their own distinct shape, material, and color.

I.3.6.2 Cannot be Used by Thieves

Checks are also less attractive to thieves than cash or cards. Checks cannot be cashed by just anyone, and that is too much of a hassle for a thief.

I.3.6.3 Traceable

When you use a check, the bank makes a copy so you can easily prove the payments you’ve made.

I.3.6.4 Not Deposited Immediately

When a client uses a check, the bank makes a copy so he can easily prove the payments he has made.

It usually takes businesses a day or two to cash the check that a user has written for them.

This means that he has a day or two to make sure he has the money in his account to cover the check.

I.3.6.5 People who have no bank account can still cash checks

If a user gets paid with checks then he can just cash them, he don’t has to deposit them into a bank account or deal with electronic funds.

I.3.6.6 Make Budgeting Easier

Every time you write a check you are simultaneously writing your own personal receipt to go along with that purchase.

These slips of paper can make budgeting easier since a user can keep those in his checkbook, and at the end of each week or month he can see an overview of all his spending in one convenient place.

I.4 Conclusion

Unlike traditional document verification, modern document verification technology can be relied on to ensure a submitted document is 100% valid. This technology uses artificial intelligence and machine learning to identify documents, authenticate them using a database of original documents, and parse personal details (name, signature, etc.).

Relying on AI to verify bank checks should speed up processes, ensure regulatory compliance, enhance fraud detection, and save banks millions in losses.

In this chapter we have presented an overview of document verification and check verification. We have highlighted the critical points that should be taken care of and identified the parts to be addressed during the next steps of our project. In the next chapter, we will introduce the basic concepts of artificial intelligence and convolutional neural networks' building blocks.

Chapter II : Artificial Intelligence

II.1 Introduction

It is hard to find someone that didn't hear in some way or another about Artificial Intelligence and some of its hot topics like Deep learning, since with the development of the machines and abundance of data, it was developed systems that simulate the ability of the human brain. Convolutional Neural Networks (CNNs) are one of the techniques of Deep learning derived from machine learning that created outstanding solutions for the image processing problems.

In this chapter, we will be discussing the basic notions and building blocks of Machine Learning, Artificial Neural Networks and Convolutional Neural Networks, for we will use CNNs in our implementation.

II.2 Basic Concepts of Machine Learning

This section introduces the main concepts of Machine Learning - a field of computer science that employs statistical methods for making computer systems able to learn from data [15]. It covers topics that are essential to understanding complex learning systems like neural networks. Starting from the brief overview of the most important machine learning concepts, this section concludes with evaluation basics.

II.2.1 Machine Learning Definition

"Machine Learning (ML) is a field of study that gives computers the ability to learn without being explicitly programmed" Arthur Samuel [16].

ML is a vast field that studies the technology of constructing a model (algorithms) that allow computers to possess the learning feature without programming the rules for each problem, where these models rely on a set of data to extract the rules that make them able to learn and predict in the future.

II.2.2 Machine Learning Types

The learning methods are classified into supervised learning, unsupervised learning, and semi-supervised learning, another scientific name for semi-supervised learning is reinforcement learning. These learning approaches are further categorized into different classes and subclasses such as classification, clustering, regression, and dimension reduction.

II.2.2.1 Supervised Learning

Supervised learning is one of the important ML techniques to predict and classify things based on the labeled data during the training of the data model. It consists of input as well as outputs in the pair form (I, O) during the training of a data model where the correct output is already known by the systems. When you're learning a task under supervision, someone is watching to see either you're getting it right or not. Consequently, in supervised learning, a complete set of labeled data is required for the training of an algorithm. Each input data set example is tagged with the answer that the algorithm should come up with the sense to recognize it. The model can distinguish the class of labeled datasets of various flowers from images. For example, when a new labeled image is given to the model, the model compares this image to the training image to classify and predicts the object from the correct label [17].

Supervised learning is performed in the context of regression and classification. In regression and classification, the aim is to find the association between input and output data to get an accurate output result. The main complexity of the regression model is a bias-variance trade-off during the model's training. You may face a low complexity problem if you have a small number of data points. This kind of problem can cause overfitting. Commonly known algorithms that have been used in supervised learning are: logistic regression, linear regression, naive Bayes, support vector machines, artificial neural networks, and random forests [18].

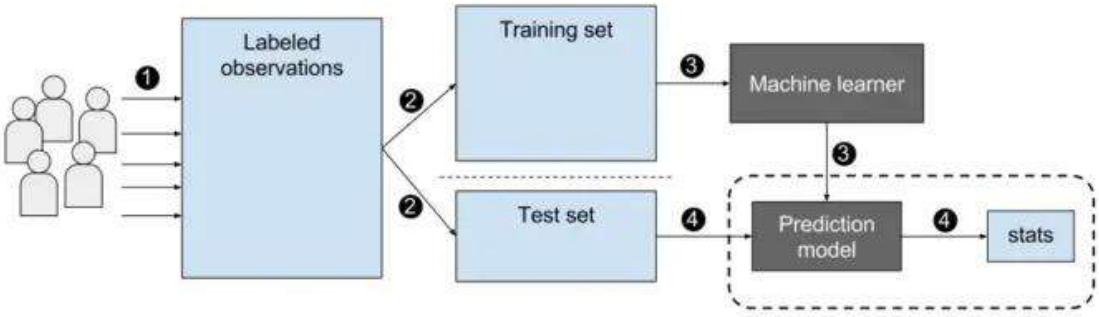


Figure II-1 Supervised learning algorithm in features extraction [19]

Figure II.1 shows the training process of a supervised machine learning algorithm that gets input data with labeled observation for training, based on prior knowledge from training the machine which can make a good prediction to produce a correct result.

II.2.2.2 Unsupervised learning

Typical, unsupervised learning refers to the learning methodology that has a huge amount of unlabeled data. In other words, it can be defined as a deep learning technique that has been implemented on the dataset without explicit instruction on how to do it and what to do. The training dataset of this learning model is only based on the input data set without any desired output dataset or correct result. A neural network is a solution to find the pattern and structure of the input dataset to make a separate cluster for extracting useful information and features. Sometimes we don't have any output values, we only have access to the set of input values In that case, the system can easily learn and identify the input data pattern [20].

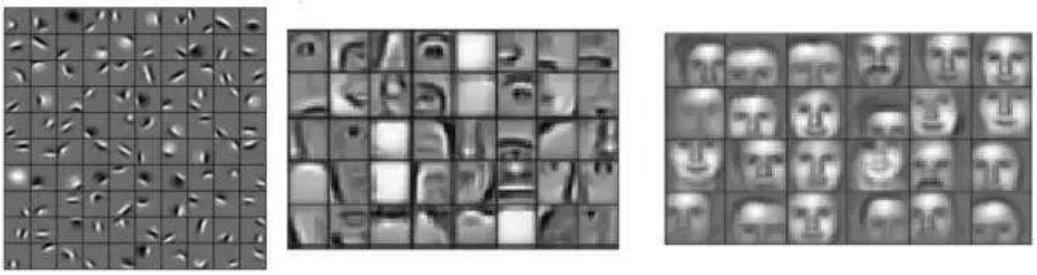


Figure II-2 Unsupervised learning algorithm in features extraction [19]

Figure II.2 is explaining the learning behavior of unsupervised learning algorithms, as the models can automatically extract the features of the object and find the suitable patterns in data for clustering purposes.

Unsupervised learning is one of the deep learning methodologies in which the algorithm does not need supervision. Instead, leave it to the model to find out what it wants to do. It is mainly concerned with data that has not been classified. The aims of using unsupervised learning include the following: It can detect all kinds of unknown patterns in data. It helps to find features from the ground which can be useful for categorization. It takes place in real-time and analyzes the input data under the supervision of the learner. It is easier to get unlabeled data instead of labelled data which requires manual supervision. For example, if you want to train a machine that can predict the time to travel from your workplace to home. It needs just some input data for prediction, including time of the day, weather condition, is it a holiday, is it raining, etc. Based on these input values the machine can predict the output value that could be the time required to travel from workplace to home [19]. The unsupervised learning algorithm is based on three main fields: clustering, Anomaly detection, and association.

II.2.2.3 Reinforcement learning

Deep learning techniques existing in between supervised and unsupervised learning are known as reinforcement learning. We may not always have direct access to the "right" output, but we can see its consistency by using the input values. RL is a mapping process from conditions to actions to maximize a scalar reward or reinforcement signal. As it's not told to the learner which actions to perform, the learner can judge automatically to perform the higher level of action which can yield the best rewards. This form of machine learning is used by AI agents and tries to find out the best way to accomplish a particular objective or improve task performance. As the agent moves closer to the goal, it receives a reward. The aim is to anticipate the best next step to maximize the final reward from the actions [21].

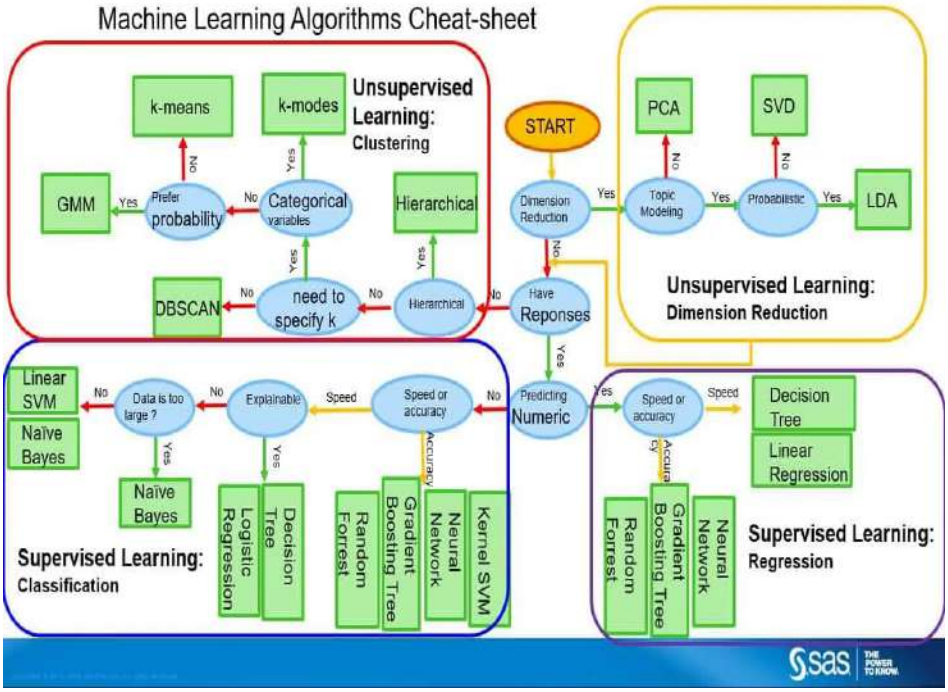


Figure II-3 Machine learning Algorithms [22]

In the figure III.3, there is a list of ML algorithms that are categorized into four different classes such as supervised learning (regression), supervised learning (classification), unsupervised learning (dimension reduction), and unsupervised learning (clustering). This figure also explains the Chee sheet of ML algorithms that have been categorized and classified into their parent groups.

II.2.3 Supervised ML Learning Algorithms

Various algorithms and computation techniques are used in supervised machine learning processes. Below are brief explanations of some of the most commonly used learning methods [23]:

II.2.3.1 Neural networks

Primarily leveraged for deep learning algorithms, neural networks process training data by mimicking the interconnectivity of the human brain through layers of nodes. Each node is made up of inputs, weights, a bias (or threshold), and an output. If that output value exceeds a given threshold, it "fires" or activates the node, passing data to the next layer in the network. Neural networks learn this mapping function through supervised learning, adjusting based on the loss function through the process of gradient descent. When the cost function is at or near zero, we can be confident in the model's accuracy to yield the correct answer.

II.2.3.2 Naive Bayes

Naive Bayes is a classification approach that adopts the principle of class conditional independence from the Bayes Theorem. This means that the presence of one feature does not impact the presence of another in the probability of a given outcome, and each predictor has an equal effect on that result. There are three types of Naïve Bayes classifiers: Multinomial Naïve Bayes, Bernoulli Naïve Bayes, and Gaussian Naïve Bayes. This technique is primarily used in text classification, spam identification, and recommendation systems.

II.2.3.3 Linear regression

Linear regression is used to identify the relationship between a dependent variable and one or more independent variables and is typically leveraged to make predictions about future outcomes. When there is only one independent variable and one dependent variable, it is known as simple linear regression. As the number of independent variables increases, it is referred to as multiple linear regression. For each type of linear regression, it seeks to plot a line of best fit, which is calculated through the method of least squares. However, unlike other regression models, this line is straight when plotted on a graph.

II.2.3.4 Logistic regression

While linear regression is leveraged when dependent variables are continuous, logistic regression is selected when the dependent variable is categorical, meaning they have binary

outputs, such as "true" and "false" or "yes" and "no." While both regression models seek to understand relationships between data inputs, logistic regression is mainly used to solve binary classification problems, such as spam identification.

II.2.3.5 Support vector machine (SVM)

A support vector machine is a popular supervised learning model developed by Vladimir Vapnik, used for both data classification and regression. That said, it is typically leveraged for classification problems, constructing a hyperplane where the distance between two classes of data points is at its maximum. This hyperplane is known as the decision boundary, separating the classes of data points (e.g., oranges vs. apples) on either side of the plane.

II.2.3.6 K-nearest neighbor

K-nearest neighbor, also known as the KNN algorithm, is a non-parametric algorithm that classifies data points based on their proximity and association to other available data. This algorithm assumes that similar data points can be found near each other. As a result, it seeks to calculate the distance between data points, usually through Euclidean distance, and then it assigns a category based on the most frequent category or average.

Its ease of use and low calculation time make it a preferred algorithm by data scientists, but as the test dataset grows, the processing time lengthens, making it less appealing for classification tasks. KNN is typically used for recommendation engines and image recognition.

II.2.3.7 Random forest

Random forest is another flexible supervised machine learning algorithm used for both classification and regression purposes. The "forest" references a collection of uncorrelated decision trees, which are then merged together to reduce variance and create more accurate data predictions.

II.2.4 Evaluation of a Binary Classifier

Sometimes it is unintuitive and hard to comprehend the accuracy of a model. However, there are specialized interpretable metrics in the classification setting, designed to show the classifier's performance from different perspectives.

II.2.4.1 Confusion Matrix

A confusion matrix is a table that is used to define the performance of a classification algorithm. A confusion matrix visualizes and summarizes the performance of a classification algorithm [24].

Figure II-4 Confusion Matrix

Predicted Value		
Actual Value	Malignant	Benign
Malignant	TP	FN
Benign	FP	TN

The confusion matrix consists of four basic characteristics (numbers) that are used to define the measurement metrics of the classifier. These four numbers are:

- **TP (True Positive):** TP represents the number of patients who have been properly classified to have malignant nodes, meaning they have the disease.
- **TN (True Negative):** TN represents the number of correctly classified patients who are healthy.
- **FP (False Positive):** FP represents the number of misclassified patients with the disease but actually they are healthy. FP is also known as a Type I error.
- **FN (False Negative):** FN represents the number of patients misclassified as healthy but actually they are suffering from the disease. FN is also known as a Type II error.

Performance metrics of an algorithm are accuracy, precision, sensitivity, and false alarm, which are calculated on the basis of the above-stated TP, TN, FP, and FN.

- Accuracy is he proportion of the total number of predictions that were correct:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

- Precision is the proportion of positive values that were correctly identified:

$$PPV = \frac{TP}{TP + FP} \quad (2)$$

- Sensitivity is the proportion of actual positive cases that are correctly identified:

$$TPR = \frac{TP}{TP + FN} \quad (3)$$

- False alarm is defined as the probability of falsely rejecting the null hypothesis:

$$FPR = \frac{FP}{TN + FN} \quad (4)$$

II.2.4.2 Receiver Operating Characteristic Curve

An ROC curve demonstrates several things:

- It shows the trade-off between sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity).
- The closer the curve approaches the left-hand border and then the top border of the ROC space, the more accurate the test.
- The closer the curve comes to the 45° diagonal of the ROC space, the less accurate the test.
- The area under the curve is a measure of comparing classifier performance. An ideal classifier has an area of 1 [25].

ROC curve can be represented with a set of pairs:

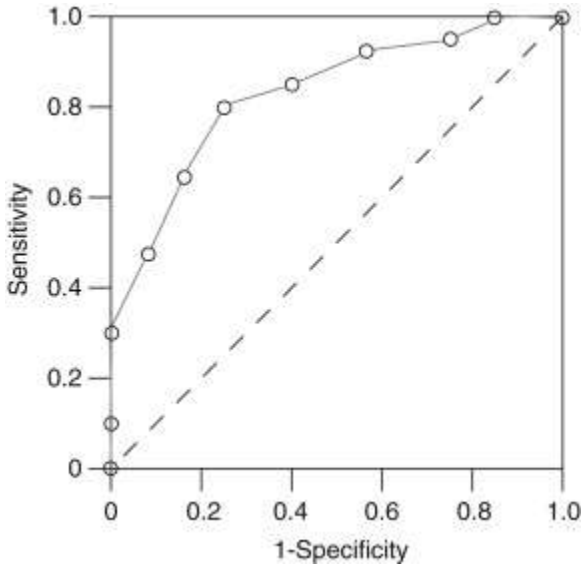


Figure II-5 A basic ROC Graph [26]

II.2.4.3 Precision Recall Curve

Much like the ROC curve, the precision-recall curve is used for evaluating the performance of binary classification algorithms. It is often used in situations where classes are heavily

imbalanced. Also, like ROC curves, precision-recall curves provide a graphical representation of a classifier's performance across many thresholds, rather than a single value. [27]

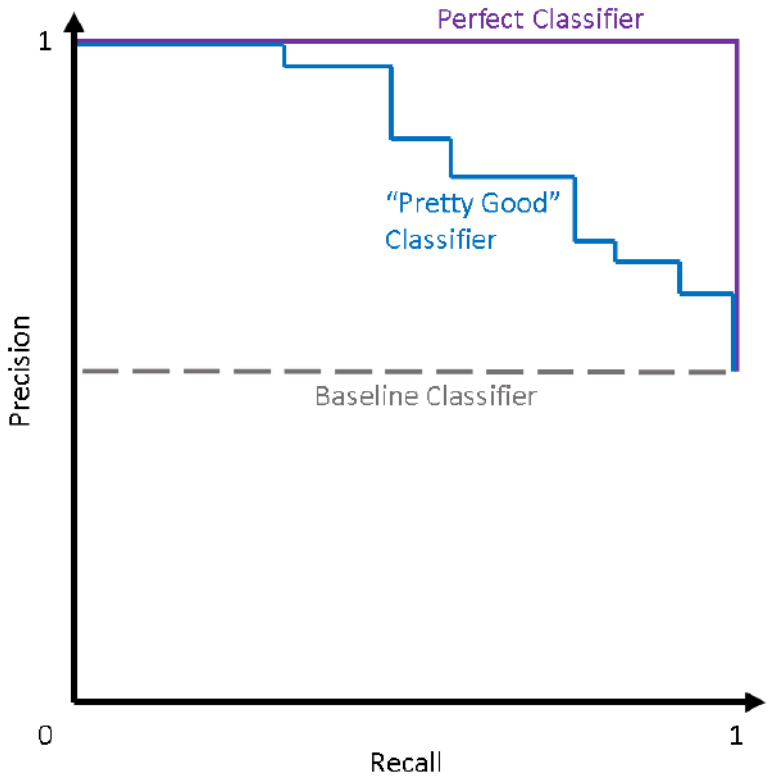


Figure II-6 Some theoretical precision-recall curves

II.3 Artificial Neural Networks

The human brain is the most precious and active organ/part of the body. It assists us in learning, understanding, behaving, acting, and deciding. Because the human brain has control over many organs of the body such as thoughts, speech, and movement, etc. The secret behind this logic is due to neurons. The neuron is considered the body's messenger because it transfers the information through impulses within different parts of the human brain and to other organs of the whole nervous system. The best way to understand the working process of a neural network is to study and understand how the natural neural network interior of the human brain works. The neurons inside the human brain are the basic component of the brain and are responsible for learning and holding the knowledge and information as we know them. Same as a human brain, an artificial neural network model was designed after the discovery of artificial neurons. It works to take input in raw forms and use different layers to process those inputs to get the desired output

from the model with the help of an artificial neuron. The artificial neural network is not efficient like the human brain, but both perform similarly. The main difference between the brain and neural network is simple, as the brain learns from experience, but the neural network learns by comparing different samples. [28]

II.3.1 ANNs Definition

Artificial Neural Networks (ANNs) are computational processing systems which are heavily inspired by the way biological nervous systems (such as the human brain) operate. ANNs are mainly composed of a high number of interconnected computational nodes (referred to as neurons), of which work entwine in a distributed fashion to collectively learn from the input in order to optimize its final output. The basic structure of an ANN can be modeled as shown in next Figure. We would load the input, usually in the form of a multidimensional vector to the input layer of which will distribute it to the hidden layers. The hidden layers will then make decisions from the previous layer and weigh up how a stochastic change within itself detriments or improves the final output, and this is referred to as the process of learning. Having multiple hidden layers stacked upon each-other is commonly called deep learning [29].

There is no universally accepted definition of neural network. It is generally considered that a neural network consists of a large set of units (or neurons), each having a small local memory. These units are connected by communication channels (connections, also called synapses in the corresponding biological term), which carry digital data. Units can only act on their local data and the inputs they receive through their connections. [29]

II.3.2 Layers in Artificial Neural Network

There are three main types of layers in artificial neural networks which can be further extended into their subtypes, they are explained below.

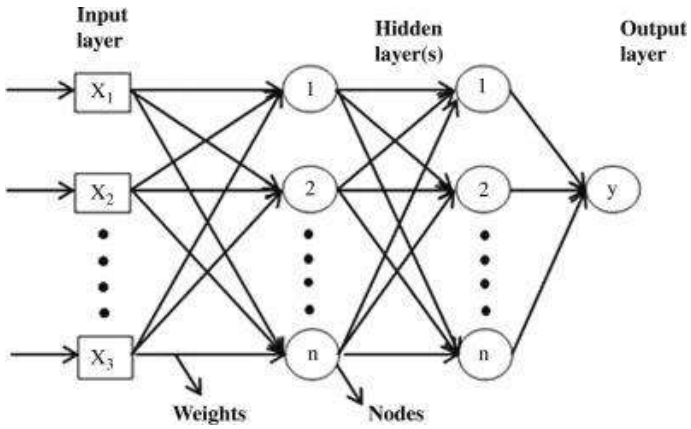


Figure II-7 Structure of Artificial Neural Network [30]

Figure II-7 describe the flow of information through different layers including input layers, hidden layers, and output layers for the artificial neural network.

II.3.2.1 Input Layers

Input layers are also known as the input node that takes information from the source in the form of signals or neurons and provides that information to the hidden layers for further processing, sometimes it's also known as visible layers in the neural network.

II.3.2.2 Hidden layers

The mapping layers between the input layers and output layers accept the data from the input layers and provide the information to the output layers. In this layer, the weight factors of the input unit are multiplied by the activation function to combine the net result. The hidden layers may be one or more depending upon the situation and need of the data model.

II.3.2.3 Output Layers

The layer of nodes that produce the output variable, as a result, is known as the output layer in neural networks. Output layers received the processed information from hidden layers.

II.3.3 Structure of Neural Networks

II.3.3.1 Inputs

For each neuron has multiple inputs x that are combined to execute some operation. Each input has a selected weight assigned to it. For example, if we have an input image of the size 50x50x1 (Height x width x Channel) pixels. Then if we input this image to a CNN model, we will have about 2500 weights in the first hidden layer itself.

II.3.3.2 Weights

Inputs of a neuron are weighted by parameters w that are changed during the learning process. Each weight gives strength to each individual input into the nerve cell. The basic awareness is that when the weight is small the input doesn't affect the output of the neuron very much. Its effect is large in the opposite case.

II.3.3.3 Bias

Another changeable parameter is bias b that controls the impact of the neuron.

II.3.3.4 Activation Function

For NN to estimate nonlinear function, each neuron must perform the nonlinear transformation of its input. This is completed with activation function φ that performs the nonlinear transformation. There are numerous different normally used activation functions. Its usage depends on the type of network and on the type of layer in which they activate.

II.3.3.4.1 Rectified Linear Unit (ReLU)

Rectified Linear Unit (ReLU) is one of the most commonly used activation function in deep learning, which is computed as:

$$f(x) = \max(0, x) \tag{5}$$

ReLU activation functions are quite popular for the creation of a non-linear network as it is easier to differentiate for back-propagation. Even though the ReLU function is not differentiable at zero unlike sigmoidal activation function, which has smooth derivatives, it still converges faster than sigmoid and hyperbolic tangent function.

II.3.3.4.2 Leaky Rectified Linear Unit (Leaky ReLU)

Leaky ReLU activation function is similar to the ReLU function with one key difference: the leaky ReLU allows a small positive gradient when the unit is inactive. A leaky ReLU is computed as:

$$f(x) = \max(x, ax) \tag{6}$$

II.3.3.4.3 Sigmoid Function

The Sigmoid function was the most widely-used activation function before ReLU, which takes real values as input and outputs the values in the range [0, 1]. The sigmoid function is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{7}$$

The larger negative input values tend to be closer to 0 while the larger positive inputs are closer to 1. Sigmoid function is rarely used now as it suffers from the gradient saturation problems and slower computation. Also, it is not zero centered.

II.3.3.4.4 Hyperbolic Tangent Function

Hyperbolic Tangent function or tanh is similar to sigmoid function with a difference of output, ranging in the interval [−1, 1] instead of [0, 1]. It is defined as:

$$\tanh(x) = \frac{2}{1 + e^{-2x}} \tag{8}$$

II.3.3.4.5 Heaviside Step Function

Heaviside step function is a discontinuous activation function which gives the output of 1 for $x > 0$ and 0 otherwise. This activation function, however, cannot be used in deep neural networks. Since deep neural networks use gradient descent with back-propagation for the training and the back-propagation requires a differentiable activation function, Heaviside Step function, being non-differentiable at x = 0, cannot be used.

II.3.3.5 Cost Function

Cost functions of NNs are a complex subject that exceeds the scope of this thesis. One of the most common cost functions used in NNs for classification in multiple classes is categorical cross entropy. In SoftMax activation function a cost function may be defined as:

$$C = -\frac{1}{n} \sum_{i=1}^{n} y^{(i)} \ln g(z^{i}) + (1 - y^{(i)}) \ln (1 - g(z^{(i)})) \tag{9}$$

Where y is the correct class of the instance and n is the total number of instances.

II.3.4 Feedforward Neural Networks

Feed-forward is a kind of neural network in which signals travel only in the forward direction and the input has been directly linked to the output. In feedforward neural networks there are no loops for backward direction, which means that the data can only flow from input to output. Networks in feedforward neural networks have fixed input and output data. Feedforward neural networks are widely used in various applications such as pattern generation and pattern recognition, document segmentation, prediction, function approximation, and classification. The mathematical representation of Feed-forward neural network models is explained below. Where the artificial

neuron takes a vector of input values such as x1, x2...xn, and every input vector is multiplied by a series of weight factors such as w1, w2.....wn, the weighted input values are combined, and a bias value (b) is added to the weighted input to produce a net output value.

$$Z= \sum_{i=1}^{n} x_i w_i + b \quad (10)$$

The final input is then passed through the activation function (g) to produce the final output a=g(z), which can be transmitted to the next neuron.

$$a= g(z)=g(\sum_{i=1}^{n} x_i w_1 + b) \quad (11)$$

Where the activation function can be chosen according to the requirement, but the weight factors and bias values (b) are selected based on the learning rules during the training phase of the neural network models.

Feedforward neural network is a classical neural network method which was used earlier for the different problems that arise from various disciplines. The network consists of multilayers and each layer is fully connected to other layers to process the input data directly into the output result. It has multiple layers; due to this reason this network is also known as the multi-layer perceptron [31].

II.3.5 ANN Learning

Neural networks learn through Gradient-based learning, which builds on the fact that the loss function can be minimized by estimating the impact of small perturbations of the parameter values on the loss function. In a Neural Network the set of parameters W is a vector of real numbers that are initialized randomly, with respect to which the loss function E(W) is continuous and differentiable almost everywhere. In such a scenario, the training aims to minimize the loss in an iterative procedure where W is updated at every step in the following way:

$$W_k = W_{k-1} - \in \frac{\partial E(W)}{\partial W} \quad (12)$$

In the simplest version, the Ô parameter is a scalar constant known as learning rate, but there are more elaborated optimizers that use variable learning rates. In general, with a high learning rate longer steps are taken in the weight updates, and thus, it may take less time for the model to converge. Nonetheless, a learning rate that is too high could result in jumps that are too large and not precise enough to reach the optimal minimum.

We can apply the above equation either on the whole dataset (Batch Gradient Descent) or on a portion of it (Stochastic Gradient Descent). A good minibatch size is small enough to avoid some of the poor local minima but large enough that it does not avoid better minima. One benefit of SGD is that it is computationally faster and it is possible to use large datasets that often do not fit in RAM.

II.3.5.1 Back-propagation

"The basic idea of back-propagation is that gradients can be computed efficiently by propagation from the output to the input" [32]

Deep Neural Networks are built as a stack of layers, each implementing a generic function $X_n = F_n(W_n, X_{n-1})$. X_n is a vector representing the output of the layer, W_n is the vector of trainable parameters of the layer, and X_{n-1} is the input of the layer, as well as the output of the preceding layer. We refer to E^p as the loss associated with the input pattern Z^p.

The main statement of back-propagation is that if the partial derivative of E^p respect to X_n is known, then the partial derivatives of E^p respect to W_n and X_{n-1} can be computed using the backward recurrence:

$$\frac{\partial E^p}{\partial W_n} = \frac{\partial F}{\partial W}(W_n, X_{n-1})\frac{\partial E^p}{\partial X_n}$$
$$\frac{\partial E^p}{\partial X_{n-1}} = \frac{\partial F}{\partial X}(W_n, X_{n-1})\frac{\partial E^p}{\partial X_n} \quad (13)$$

where $\frac{\partial F}{\partial W}(W_n, X_{n-1})$ is the Jacobian of F with respect to W evaluated at the point (W_n, X_{n-1}), and $\frac{\partial F}{\partial X}(W_n, X_{n-1})$ is the Jacobian of F with respect to X. The first equation is responsible for calculating the gradient respect to the model parameters, while the second equation is responsible for the back-propagation.

II.3.5.2 Transfer-Learning

"Transfer learning and domain adaptation refer to the situation where what has been learned in one setting is exploited to improve generalization in another setting" [33]

The main reason for the success of transfer learning is that models trained on huge datasets learn general features that facilitate the training process with small datasets, often obtaining better performances with respect to models trained from random weights. A first application of transfer

learning uses Deep Neural Networks as a fixed feature extractor. The last fully-connected layer is removed, and the network is used as a feature extractor. Once the features are extracted, it is possible to train a classifier for the new dataset. The second strategy is to replace the top classifier and fine-tune the weights of the whole network by continuing the back-propagation. It is possible to fine-tune all the layers, or it's possible to keep the first layers frozen and only tune the higher-level layers. The observation motivates this is that the first layers contain more generic features, whereas the higher-level layers are more specific to the details of the classes of the original dataset [34].

II.4 Convolutional Neural Networks

CNNs, like neural networks, consists of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, passes it through an activation function and responds with an output. The whole network has a loss function and all the tips and tricks we developed for neural networks still apply on CNNs [35].

The only notable difference between CNNs and traditional ANNs is that CNNs are primarily used in pattern recognition within images. This allows us to encode image-specific features into the architecture, making the network more suited for image-focused tasks [36].

This choice was motivated mainly by implicitly incorporating a feature extraction phase and has been used successfully in many applications [37].

One of the largest limitations of traditional forms of ANN is that they tend to struggle with the computational complexity required to compute image data [36]. That is the reason behind implementing CNNs, the properties that CNNs have such as feature extraction make them more efficient when handling images.

II.4.1 Convolution operation

Convolution is a two-operand operation applied on an input and a system function to figure out the output. To better understand the concept of convolution, the following section will be based on an example of tracking the location of a spaceship with a laser sensor to deduct two functions that they may be of use. This sensor returns only one output x(t), the spaceship's position at a given time t. x and t represent real values in case the readings from the laser are accessible in any instant [33].

In a realistic environment the laser sensor would be noisy. For more accurate readings, several measurements should be averaged. Naturally, that the more recent the measurements are, the more accurate the position. That's why the average to be calculated will give more weight to recent measurements. This can be done with a weighting function w(a), where a is the age of a easurement. If such a weighted average operation is applied at any possible instant, a new function s providing a smoothed estimate of the position of the spaceship can be obtained [33].

$$S(t) = \int x(a)w(t-a)da \quad (14)$$

This operation is called convolution. The convolution operation is typically denoted with an asterisk:

$$S(t) = (x * w)(t) \quad (15)$$

Convolution can also be an approach to different problems other than calculating a weighted average. The different actors in this example have specific terms in the machine learning field. For instance, the convolution is referred to as the input (function x). The function w is the kernel (weight) and finally the feature map, which represents the output [26].

Assuming in this example that x and w are defined only on integer t, the discrete convolution can be defined:

$$(x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)\, w(t-a) \quad (16)$$

Goodfellow et al. introduce the concept of tensors and explain it as follows ―In machine learning applications, the input is usually a multidimensional array of data, and the kernel is usually a multidimensional array of parameters that are adapted by the learning algorithm. these multidimensional arrays will be referred to as tensors.”

Because each element of the input and kernel must be explicitly stored separately, it is usually assumed that these functions are zero everywhere but, in the finite, set of points for which the values are stored. Goodfellow et al. relate this assumption to the practice by ― it is possible to implement the infinite summation as a summation over a finite number of array elements” [33]. Finally, if a Convolutional Network has for example a two-dimensional image I as an input it only makes sense to use a two-dimensional kernel K. That's why convolution is often used over more than one axis at a time:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)\, K(i-m, j-n) \quad (17)$$

The kernel relative to the input was flipped in the above equation, in the sense that as m increases, the index into the input increases, but the index into the kernel decreases. This only happened so convolution gained commutativity. Despite this property 's usefulness for writing proofs, it doesn't have an impact on the training process of a NN. Cross-correlation is a related function that has the same utility of convolution but without needing to flip the kernel, and it's used in many NN libraries [33].

$$S(i , j) = (I * K) (i , j) = \sum_m \sum_n I (i + m , j + n) K (m, n) \quad (18)$$

II.4.2 Layers in CNN

A CNN comprises of multiple layers of operations, each with its own function. The CNN takes an image as input and feeds it to the first layer. Feature information is propagated through the hidden layers. For each layer, the activation functions perform an element-wise activation to the output produced by the previous layer. The output of the final layer is then compared against the targeted output. This process is known as forward pass.

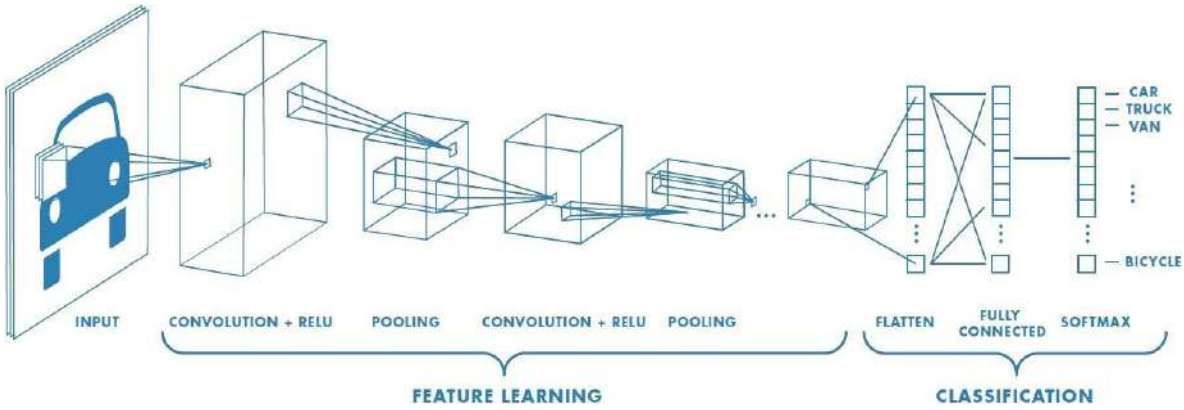


Figure II-8 A simple CNN architecture [38]

II.4.2.1 Input Layer

The input can be an image input or a video stream. Both are basically a collection of pixels which are placed in an array to form an image. Each pixel has a numeric value in the range of -255 to +255. The numbers represent the color value of the pixels in the image or video. The image or video stream is first converted to feature maps which consist of an array of numbers stacked together using NumPy library.

II.4.2.2 Convolutional Layer

The convolutional layer plays an important role in how CNNs operate. This layer's parameters focus on small n × n windows called kernels, filter matrices or simply filters. Given input data, the convolutional layer convolves each filter or kernel across the spatial dimensionality of the input to produce a 2D feature map (or an activation map after computing the activation function). These activation maps can be visualized in the next figure.

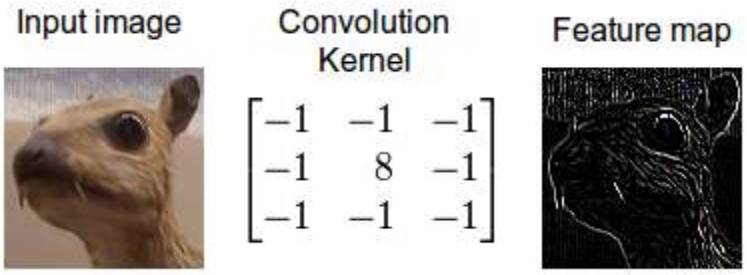


Figure II-9 Feature maps after convolution filtering [39]

Classically, the convolution operation is defined as an integral that expresses the amount of overlap of one function g as it is shifted over another function f, given as:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau \quad (19)$$

where f and g are the arbitrary continuous functions in domain t. In case of an image, the convolution operation is defined as a discrete function. The discrete convolution operation between an image f and a filter matrix g is defined as:

$$f[n] * g[n] = \sum_{m=-\infty}^{\infty} f[m]\, g[n - m] \quad (20)$$

The feature map is the output matrix h obtained by aligning the filter successively with every sub-image of f (with same dimensions as g) centered on coordinates x, y. The convolutional layer of a neural network is a combination of multiple convolutional filters, whose matrix values are treated as neuron parameters. Successive repetition of the convolutional layers along with some other types of layers (such as the pooling layer) results in a Convolutional Neural Network (CNN). [36]

II.4.2.3 Activation Layer

Feature maps which are produced by the convolutional layers are much smaller in size than the input matrix. Since the CNNs resemble NNs, they as well, adopt activation functions which are

arranged in the layer called the activation layer. Though this layer is considered to be a successive part of the preceding convolutional layer, it is not shown explicitly in topological schemes since it is almost always used after a convolutional layer.

Activation functions determine the output of the neuron, based on the input value. Various activation functions exist for CNNs such as linear function, sigmoid, tanh, binary step function and ReLu (Rectifying Linear Unit). Non-linear activation functions are the most popular ones since they have localized outputs thus reducing the complexity of the input and making it possible for the network to generalize or adapt with data. They better suit with the backpropagation learning algorithms which allow the network to train itself making it capable to learn and perform more complex tasks.

ReLu is the most common activation function recently, which is defined as:

$$Relu(x) = \max(x, 0) \tag{21}$$

It is simply performing a threshold operation on the input and can be implemented and calculated more easily than tanh or sigmoid functions. Thresholding operation causes only a few neurons to be activated at a time. These attributes of the ReLu function make it possible to reduce overall computational cost effectively.

II.4.2.4 Subsampling Layer

The next layer is a subsampling layer in CNN architecture. It is also called a down sampling layer. The subsampling layer operates on the output of the activation layer. The intuition for the subsampling operation is that the exact location of a feature is not important if it has been found. The result of the subsampling operation is reducing the size of feature maps while keeping the important information. That means there would be fewer parameters to be calculated. Thus, subsampling, helps to reduce the computational cost. Subsampling also helps to prevent overfitting.

Usually, types of spatial pooling are used in the subsampling layers. These might be one of the followings:

- Max Pooling
- Average Pooling
- Sum Pooling

Max pooling is taking the largest element of the non-overlapping rectangular regions of the output of the activation layer. Average pooling and sum pooling take the arithmetic average or the sum of elements of the same region, respectively. Consequently, for the given pooling region of size (Px, Py), the input image is down sampled by a factor of Px and Py along each direction. [40]

Max pooling is the most commonly used type of subsampling that supplies position invariance over larger local regions. Scherer, Müller, & Behnke [41] states that 19 max pooling can converge faster, yield better generalization and select superior invariant features. General feature pooling and max pooling's theoretical analysis can be found in the work of Boureau, Ponce, & LeCun [42].

II.4.2.5 Dropout Layer

Dropout layer functions only in the training phase of the network. Main purpose of using it is to prevent overfitting. This may occur especially when the training dataset volume is relatively low. When we train a network with a low quantity of samples, the network might not find enough samples to generalize the data and learn the specific features of the given training data which are thought to be "noise" since these features don't represent the true nature of the data source. This effect can be thought to be "memorizing" the training data. This is sometimes called a "generalization error". Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov [43] show that dropout is an efficient regularization technique for reducing this sort of error in many cases.

As its name implies, this technique simply drops out neurons at given probability in the training phase. Dropped out neurons' outputs 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 which 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. However, it is advised to use it only for fully connected layers and not for convolutional layers.

II.4.2.6 Batch Normalization Layer

Stochastic gradient descent algorithm is one of the most used algorithms for training the deep neural networks. In this algorithm, training dataset volume is divided into mini-batches over which the loss is calculated instead of using all the training data at once or using them one by one. This technique increases the training speed effectively. But this may give rise to a problem which

is called an “internal covariate shift” [44]. The reason for this problem is having mini-batches with different distributions. These distributions could have different means and deviations which may cause imbalanced gradients therefore cause an “exploding gradient” effect. This brings about the instability and decrease in the learning speed since each layer tries to learn a new distribution at every training step.

The batch normalization layer aims to solve this problem and thus increase the learning speed and stability. Generally speaking, normalization puts all the data points on the same scale. Similarly, to the preprocessing of inputs for normalization before the input layer, the batch normalization layer normalizes the previous layer’s outputs for the next layer by subtracting the batch mean and dividing by the batch standard deviation. This process forces the data points to have zero mean and one standard deviation, according to the following calculations [44].

- Activations for a mini-batch:

$$B = \{x_1..m\} \tag{22}$$

- Mini-batch mean:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} xi \tag{23}$$

- Mini-batch variance:

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x - \mu)^2 \tag{24}$$

- Normalized activation:

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \varepsilon}} \tag{25}$$

where ε is an arbitrarily small constant for numerical stability. Batch normalization output:

$$y_i = y\hat{x}_i + \beta \tag{26}$$

where, ϒ and β are hyperparameters to be learnt in the training phase. Each unit has their own set of those parameters.

Batch normalization process helps to avoid weights and biases to become imbalanced with extremely high and low values which will over-influence the training process. It also prevents

nonlinearities e.g., sigmoid to become saturated thus helps to solve the "vanishing gradient" problem.

Increased stability makes the network less sensitive to how weights are initialized and makes possible to increase the learning rate. Moreover, since the scaling results in adding some noise within that mini-batch, it gives a similar effect to dropout, therefore providing some regularization and reducing the generalization error. It is not recommended to use batch normalization and dropout layers together since the normalized activations could be noisier after dropout. Batch normalization layer may take place before or after the activation layer. This process is at batch basis hence the name "batch" normalization.

II.4.2.7 Fully Connected Layer

Fully-connected layers are the final layers of a CNN that contain several neurons, all connected between two adjacent layers [36]. A fully-connected layer generally acts as a classifier at the end of the CNN and is analogous to a traditional ANN.

II.4.2.8 Softmax Layer

CNN's layer structure generally ends with a consecutive softmax layer and a classification output layer, for the mutually exclusive multiple class problems. Softmax function acts as an activation function which is applied at this layer to the preceding fully connected layer's output. Softmax layer's size is also equal to the total number of classes of the dataset as the final fully connected layer's size. Softmax layer assumes each input belongs to one and only one class. Each softmax layer's output will be in the interval (0, 1), and they will add up to 1, so that they can be interpreted as probabilities. Softmax function is defined as:

$$Softmax(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} for\ i = 1, .., K\ and\ z = (z_1, .., z_k) \in \mathbb{R}^k \quad (27)$$

where K is the number of classes of the dataset, z is the output of the previous layer. This function enlarges differences in the input data and is also known as normalized exponential function. It is fully differentiable therefore is appropriate for training algorithms that use gradients such as gradient descent. The main objective of this layer is to turn the network's output to probabilities of each input element belonging to a class therefore output represents categorical distribution over class labels.

II.4.2.9 Classification Output Layer

Remaining other calculations are carried out in a layer which is called a classification output layer that follows the softmax layer. Generally, two functions attached to this layer. First, it assigns outputs a label according to the output of the softmax layer. Second function runs only in the training phase to calculate loss function or a.k.a. cost function. Sometimes this layer is called a loss layer because of this functionality. Loss function is used in the training of the network by the backpropagation algorithm which tries to reduce the output of this function, or namely the "loss" …. There are several loss functions used in accordance with the specific application at hand, such as Mean Absolute Error (MAE or L1 loss), Mean Squared Error (MSE or L2 loss), 0/1 loss, Huber loss, Hinge loss, cross entropy, etc. Softmax layer is usually used with the cross-entropy loss function for multi-class classification problems with mutually exclusive classes. Cross entropy loss is a measure of total distance between the network label predictions and the original labels over all the samples.

Cross entropy for the correct output y and network's prediction for a sample is defined as:

$$\mathrm{H}(y, \hat{y}) = -\sum_{m=1}^{M} y_m \log \hat{y}_m \qquad (28)$$

where M is the number of classes of the dataset. Here the network's prediction ŷ is the value from the softmax function in this case.

Then the average total loss is calculated as:

$$loss = \frac{1}{n}\sum_{1}^{N} H(x_n, \hat{y}_n) = -\frac{1}{N}\sum_{1}^{N}\sum_{1}^{M} y_{m,n} \log \hat{y}_{m,n} \qquad (29)$$

where N is the number of samples, $y_{m,n}$ is the correct output for the nth sample, $\hat{y}_{m,n}$ is the network's prediction for the nth sample.

For the problems involving prediction and classification, cross entropy loss function has some advantages over the other types of losses especially when gradient descent-based learning algorithms are used. MSE loss function for example, penalizes the outliers strongly. Badly misclassified samples over-emphasized and shift the training too much.

II.4.3 Performance optimization of CNN

There are several techniques that can improve the performance of a CNN. The optimization techniques help in following ways:

- Faster CNN convergence.
- Avoiding the local minima.
- Prevent overfitting.
- Improving accuracy or reducing error rate.

There are several factors that affect the Convolution Neural Networks performance as discussed below.

II.4.3.1 Data sets

Before going further two terms need to be understood: the training and testing set. Generally, the accuracy of a neural network depends on the size of the dataset [45], which should be as large as possible. The dataset can be expanded by adding several forms of distorted data. This data helps in learning using different patterns to generalize the neural network. The available examples are divided into two sets, training and testing set in the ratio of 80: 20. Each set consists of examples and the class information to which the respective example belongs, also called labeled data for supervised learning.

II.4.3.2 Backpropagation

The main idea of backpropagation is to compute gradients for a network consisting of more than two processing layers. The layers between the input and output layers are known as hidden layers. If there are only two layers i.e., input and output layer, the error computation is simply the difference between output value and the target value and depending upon the error the weights can be changed. However, if a network has more than two layers the computation of error gradient is difficult for the hidden layers, but can be done using back propagation. In backpropagation the gradients can be computed by propagating the output to input through the hidden layers, adjusting the weights and biases along the way. The number of adjustments of weights and biases is calculated through gradient descent algorithm, where gradient descent seeks to minimize the error function by changing the weights and biases.

II.4.3.3 Momentum

Momentum improves the convergence speed, where the current weight change depends on the previous weight change [46]. A momentum is added to improve the convergence speed. It controls the previous weight change on the current weight change from oscillating. The following shows the weight update formula. The weight update as follows:

$$w(n+1) = w(n) + \alpha\Delta w(n) + \eta\Delta w(n-1) \quad (30)$$

Where α is the momentum which lies between $0 \leq \alpha < 1$ and η is the learning rate [47]. Learning rate is a crucial parameter which determines the rate of convergence and generalization of the neural network. It influences the speed at which the network attains minimum value of cost function. It is used in updating the hyperparameters or weights during each iteration. The learning rate affects the quality of the network, so optimal selection of learning rate is a crucial design constant. A small learning rate takes more time to converge and can lead to divergence. The optimal rate is given by:

$$\eta_{opt} = \left(\frac{\partial^2 J}{\partial w^2}\right)^{-1} \quad (31)$$

Where η_{opt} is optimal learning rate, J is the cost function and $\frac{\partial^2 J}{\partial w^2}$ represents the double differentiation of cost function with the hyper parameters or weights.

II.4.3.4 Cross validation

Cross validation is a technique which separates data into disjoint sets, training sets and test sets. The available labelled examples are randomly divided into training and test sets [48]. The motivation here is to train the CNN with the training data, which results in a model. The obtained model is tested with the test set i.e., to assess the performance and choose the best model. The training and test sets are disjoint, that new examples are given as input to the model and accuracy is obtained. This helps in generalization of the model.

II.4.3.5 Weight initialization

Initialization of CNN weights is a critical task to achieve good performance. Generally initializing all weights to 0 is ineffective. The initial weights are to be initialized for fast and uniform learning, which helps weights reach equilibrium simultaneously. In practice the weights should be chosen randomly from the same distribution function to ensure uniform learning [49].

II.4.3.6 Dropout method

Convolutional neural networks contain many parameters due to multiple nonlinear hidden layers; these parameters model the complicated relationship between the inputs and outputs and make it a very powerful machine learning system. A model with a large number of parameters is prone to overfitting. Over-fitting is a problem in neural networks where the error rate on new test sets is much higher than on training data.

Dropout helps to prevent overfitting and stops training a neural network when the error rate decreases in the test data set. Overfitting can be reduced by randomly omitting half of the kernels on each training set or omitting with a predefined ratio. Each kernel unit is retained with a fixed probability p which is independent of other kernel units, where p can be set using a validation set or simple 0.5, which seems to be optimal for a wide range of architectures [43]. The probability for the input layer feature maps must be higher than 0.5 because the lower probability makes the input feature maps which further reduces the information in the input.

After reducing the number of nodes on the training data, dropout reduces overfitting and also improves the speed of training. The training of a neural network with a dropout layer shows a better error rate when compared to the network without dropout layer [50].

II.4.4 CIFAR CNN Example

II.4.4.1 Dataset

CIFAR dataset consists of 60000 small square 32 by 32 pixels RGB images of one of the 10 classes (Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck), in which 50000 images are for training and 10000 images are for testing [51].

II.4.4.2 Training

For the CIFAR CNN Model, we tested two CNN architectures: architecture #1 that uses VGG3 model, and architecture #2 that is an enhancement of the VGG3 model [52].

We used the CPU based computation option to train this, the epoch was 20 and the batch size was 128.

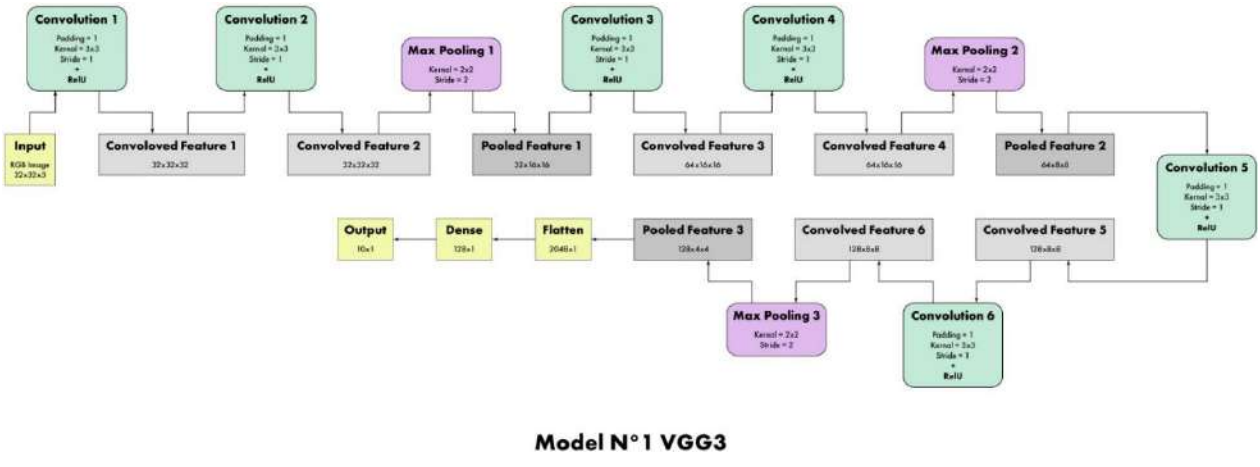


Figure II-10 CIFAR CNN Architecture #1

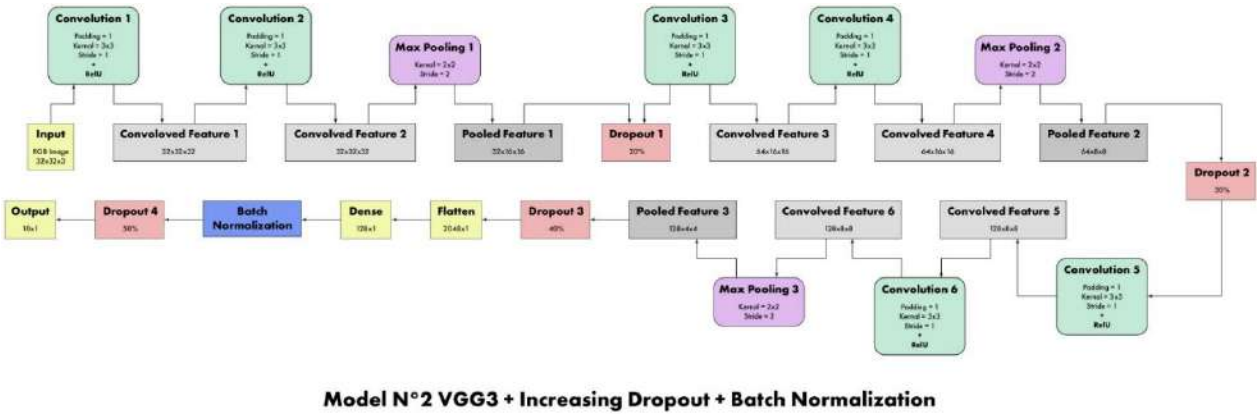


Figure II-11 CIFAR CNN Architecture #2

The next terms influenced our model so we they will be explained briefly:

II.4.4.2.1 Categorical Crossentropy

Categorical crossentropy is a loss function that is used in multi-class classification tasks. These are tasks where an example can only belong to one out of many possible categories, and the model must decide which one.

II.4.4.2.2 Adam

Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.

II.4.4.2.3 Accuracy

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right.

II.4.4.2.4 Epoch

One epoch = one forward pass and one backward pass of all the training examples.

II.4.4.2.5 Batch Size

It is the number of training examples in one forward/backward pass. The higher the batch size, the more memory space you'll need.

II.4.4.2.6 Number of Iteration

It is the number of passes, each pass using [batch size] number of examples. To be clear, one pass = one forward pass + one backward pass (we do not count the forward pass and backward pass as two different passes).

II.4.4.2.7 Validation Split

It is a fraction of the training data to be used as validation data. The model will set apart this fraction of the training data, will not train on it, and will evaluate the loss and any model metrics on this data at the end of each epoch, it floats between 0% and 100%.

For our CIFAR model we have 50,000 training examples, our batch size is 128, our validation split is set to 10%, then it will take 352 iterations to complete 1 epoch out of 20 epochs (50,000 = 128 x 352 + 9.88% x 50,000).

II.4.4.2.8 He Weight Initialization

It's the current standard approach for initialization of the weights of neural network layers and nodes that use the rectified linear (ReLU) activation function.

II.4.4.2.9 Batch Normalization

Batch normalization is a layer that allows every layer of the network to do learning more independently. It is used to normalize the output of the previous layers. The activations scale the input layer in normalization. Using batch normalization learning becomes efficient also it can be used as regularization to avoid overfitting of the model. The layer is added to the sequential model to standardize the input or the outputs. It can be used at several points in between the layers of the model. It is often placed just after defining the sequential model and after the convolution and pooling layers.

II.4.4.3 Results

We will be discussing the results from the architecture #1 compared to architecture #2.

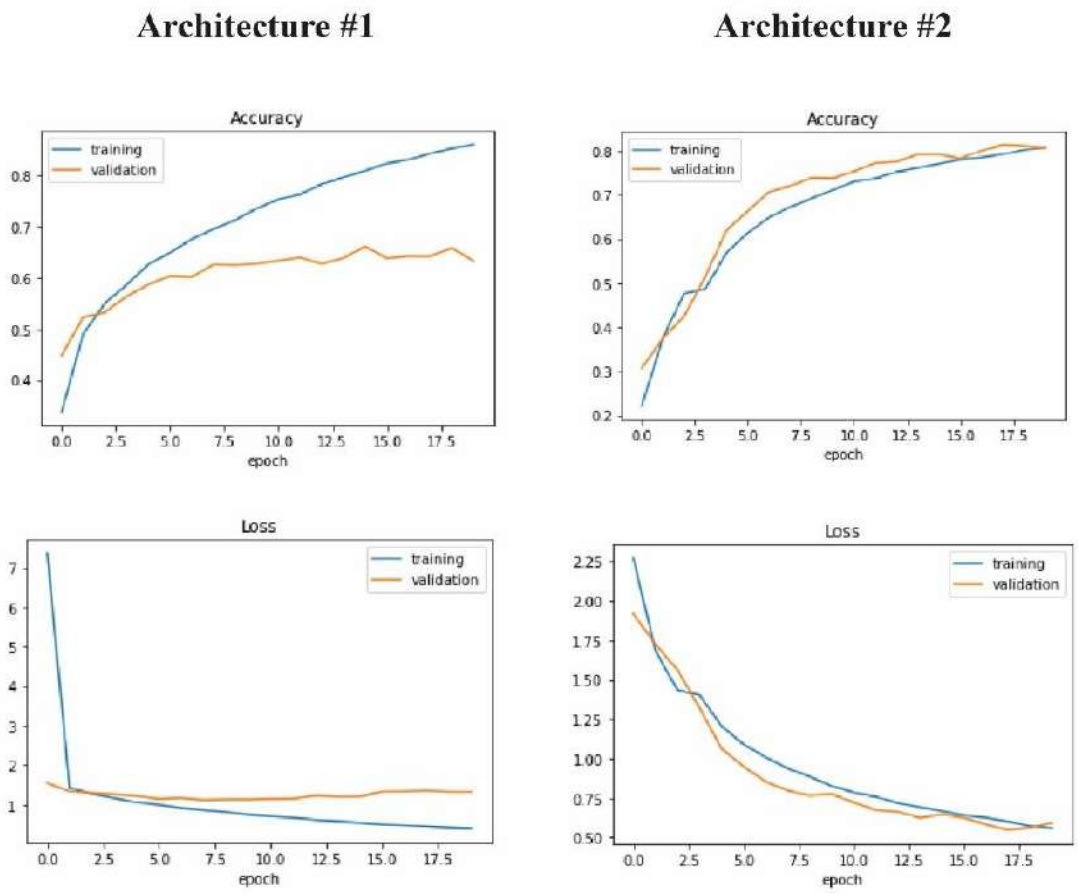


Figure II-12 CIFAR CNN Training Loss and Accuracy Results

From the above two plots (figure II-12 Architecture #1), we can see that the validation accuracy became flat after a couple of epochs and didn't increase enough after. In the beginning, the validation accuracy was linearly increasing for a while, but then it did not increase much.

Since our model has high number of parameters, for the architecture #1 model it causes overfitting (Overfit the training dataset by "memorizing" it, so the test error will be far higher than the training error). For avoiding overfitting, we added a dropout layer in Architecture #2 which randomly ignore (Input neurons are not considered during a particular forward or backward pass) a percentage of the input neurons feeding it in each training set (epoch).

From Figure II-12 (Architecture #2) we see that adding dropout in our mode worked, we can see that the validation loss and validation accuracy both are in sync with the training loss and accuracy. It shows that our model is not overfitting: the validation loss is decreasing instead of increasing, and the gap between training and validation accuracy is minimum.

Exposing Architecture #2 to test data yielded an accuracy rate of 79.51% which is 18.12% higher than the Architecture #1 accuracy which is 61.39%.

Classification accuracy alone can be misleading if we have an unequal number of observations in each class or if we have more than two classes in our dataset.

A confusion matrix is a technique for summarizing the performance of a classification algorithm.

Calculating a confusion matrix can give us a better idea of what our classification model is getting right and what types of errors it is making.

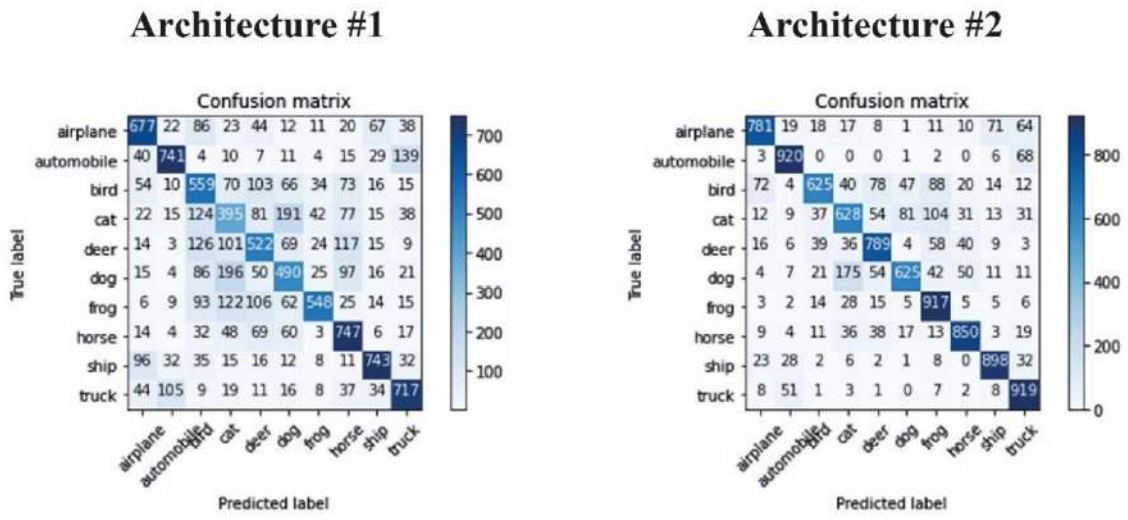


Figure II-13 CIFAR Model Confusion Matrix

The diagonal elements represents where the predicted label is the same as the true label, while off-diagonal elements represent the mislabeled elements by our classifier. The higher the diagonal values the better which is the case in our model.

The off-diagonal elements refer to the unbalance of our model, both of our models did misclassify a god number of samples, so Architecture #2 is much better than Architecture #1 when it comes to our model's preciseness and balance, although the model can be enhanced more to get an accuracy above 97%.

II.5 Conclusion

After we had an idea of how ANNs and CNNs basically work and their role in the deep learning field, we know that CNNs are the most used and the best neural networks to deal with image recognition and pattern detection because of their features.

The next chapter; deals with the various materials used in our project; software and hardware.

Chapter III : Materials Used

III.1 Introduction

This chapter provides a brief overview of the most important technologies used in developing our project. First, we'll go over the hardware we used, including what it is and its features. Then we'll go over the major software technologies we used to build the Graphical User Interfaces, database, and paper check verification framework.

III.2 Hardware Used

III.2.1 Raspberry pi 3 Model B

The Raspberry Pi is a low cost, credit-card sized computer. It enables people of all ages to explore computing, and learn how to program. It runs on an operating system based on Linux, so it can do everything you'd expect a desktop computer to do but at a lower performance [53].

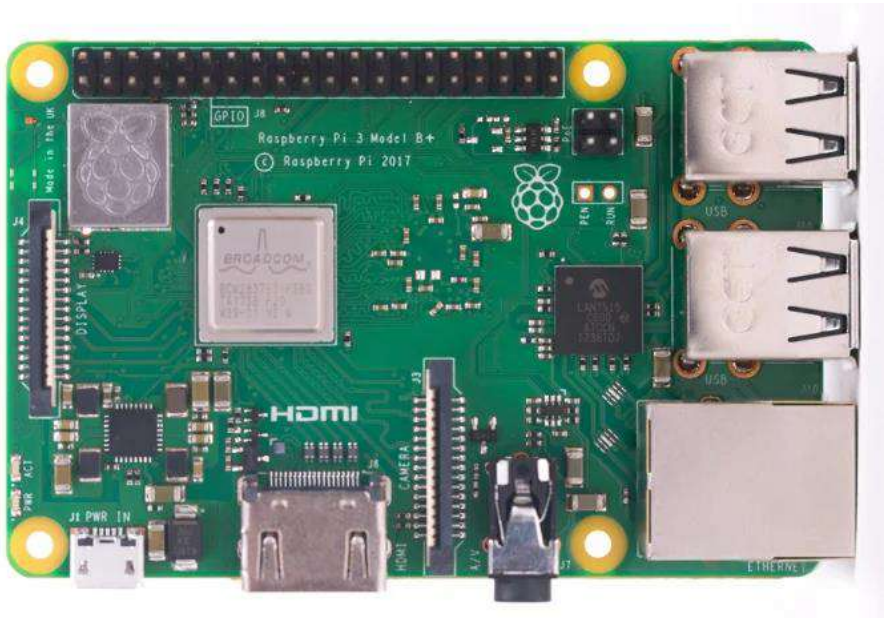


Figure III-1 Raspberry Pi 3 Model B [53]

Raspberry PI 3 Model B has the following specifications:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU.
- 1GB RAM.
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board.
- 100 Base Ethernet.
- 40-pin extended GPIO.
- 4 USB 2 ports.

- 4 Pole stereo output and composite video port.
- Full size HDMI.
- CSI camera port for connecting a Raspberry Pi camera.
- DSI display port for connecting a Raspberry Pi touchscreen display.
- Micro SD port for loading your operating system and storing data.
- Upgraded switched Micro USB power source up to 2.5A.

III.3 Software Used

III.3.1 Python

Python is considered one of the most powerful and appealing programming languages, especially in machine learning and data science. There are bundles of python packages that can be used to solve ML complex problems. The syntax of python is very straightforward to learn, as its syntax is closely related to natural languages such as English. The code in python is easy to write and it can provide faster execution than in other programming languages. The popularity of python is due to its more expressive and conciseness language compared to the other programming languages such as C#, C++, and Java, as it takes less time and resources to perform the same tasks. Python is the only programming language that can easily support any kind of platform such as windows, mac, Linux and provide open-source packages for any platform. Python can be used in various fields of computing, as it can be used to develop web applications, it can be used to develop mobile applications, it can be used to develop desktop applications, it can be used for scripting and automation processes, it can be used in computer vision for different purpose such as object detection, clustering, classification, text recognition (handwritten as well as printed text) and pattern recognition, etc. In this project, we have used python as the main programming language to write our source code for the paper check verification.



Figure III-2 Python Programming Language Logo

III.3.2 TensorFlow

Neural Networks have been heavily used to solve different and challenging problems in the last few years. Neural Networks have been heavily used to solve different and challenging problems in the last few years. Since then, many frameworks that facilitate the implementation of these networks have appeared. TensorFlow is one of them. Google developed it. It was only used internally in the company, then Google made it open source, and it now includes a large and active community that helps improve it every day. This framework is built to support every aspect of the process, from training models to deploying them to servers or mobile devices, making it stand out from other frameworks [54].



Figure III-3 TensorFlow Logo

III.3.3 Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

Keras is [55]:

- Simple -- but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.
- Flexible -- Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned.
- Powerful -- Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, or Waymo.



Figure III-4 Keras Logo

III.3.4 OpenCV

OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection. It was originally developed in C++. In addition to it, Python and Java bindings were provided. OpenCV runs on various Operating Systems such as Windows and Linux.

Using OpenCV library, we can:

- Read and write images
- Capture and save videos
- Process images (filter, transform)
- Perform feature detection
- Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it.



Figure III-5 OpenCv Logo

III.3.5 SQLite

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is a database, which is zero-configured, which means like other databases you do not need to configure it in your system.

SQLite engine is not a standalone process like other databases, you can link it statically or dynamically as per your requirement with your application. SQLite accesses its storage files directly.



Figure III-6 SQLite Logo

III.3.6 Tkinter

Tkinter is a cross-platform GUI framework built into the Python standard library. Since it is a cross-platform framework, the same code can be used on any operating system, such as Linux, macOS, and Windows. Tkinter provides an object-oriented interface to the Tk GUI toolkit.

Compared to other available options, Tkinter is lightweight and easy to use. Therefore, it is the go-to choice for quickly building applications that can operate cross-platform and do not require a modern look.

III.3.7 Jupyter and Google Collaboratory

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

Google Collaboratory is a hosted Jupyter notebook environment that is free to use and requires no setup. It’s a free cloud service and it supports free GPU!

Some libraries like Keras, TensorFlow, PyTorch, and OpenCV are already installed. So, the user doesn't need to install them for developing their deep learning applications for example.

The most important feature that distinguishes Colab from other free cloud services is that it provides GPU and is totally free. No membership or a credit card is needed to use it. Colab provides a free CPU and a GPU.

III.3.8 IP Webcam

IP Webcam is an app that coverts an Android device into an internet camera with multiple view options that can be seen on any platform using VLC player or an internet browser

IP Webcam allows the recording of videos in Webm, MOV and MPEG4 (only offered for devices with the Android 4.1 operating system or later). On the other hand, the audio transmission can be as a Wav, Opus or ACC file (the last of these requires Android 4.1 or later).

The movement detection options allow leaving the phone pointing somewhere, and it will start recording when something moves in front of it. This mode, of course, consumes a lot of the phone battery, so it's only a good idea to use it when the device is plugged into a power supply [56].



Figure III-7 IP Webcam Android App

III.4 Conclusion

After we had an idea of the most important technologies used in our project development, we could see that a considerable part of our solution depends on software.

The next chapter; deals with the technical and practical implementation of our proposed solution.

Chapter IV : Implementation, Tests and Results

IV.1 Introduction

After having presented the different hard and soft used tools, in this chapter, we will explain in detail our proposed solution's technical and practical implementation.

Before starting the project's design and implementation, we should first create a detailed description of our system, known as the specification book; we should then conduct some experiments and preliminary testing to achieve acceptable practical results. To better understand our project's general functioning, we will present our system's flow diagram and the specification book. Then, after doing the segmentation phase, we used four different CNNs that we worked on enhancing their efficiency; once we have those four models, we will proceed to the implementation of our project.

IV.2 Specification Book

In order to carry out our automated verification system, we have drawn outlines that are considered to be requirements for our project, making its purpose clear and precise.

IV.2.1 Purpose

The purpose of this project is to automate the check verification process. Thus, preface the teller line's automatization, increase their profitability by decreasing the costs associated with the additional staff necessary to maintain compliance, and improve the customer experience by speeding up the transaction process more accurately because the human factor was eliminated.

IV.2.2 Product Features

- The check verification must be robust and fast to replace the traditional teller line.
- The system must have a high accuracy rate, since it deals with money transactions.
- The system must be extensible and easily expandable (E.g., the administrator should be able to add new users to the database).
- The system text recognition approach should be versatile for it to be used in other types of financial institution documents.

IV.2.3 Interfaces Requirements

- The administration should be able to add new users to the database, and retrain the signature model using a graphical interface running in a windows machine.

- The bank cash counter should have light indicators showing the state of the check verification (a blue LED light when the check verification is confirmed, and a red LED light when the check verification is denied).

IV.3 Global System Flow Diagram

The synoptic diagram corresponding to the overall operation of our system is shown in Figure IV-1. It's composed of three parts: segmentation, processing and database part. A user puts their check on the cash counter and presses a push button to withdraw money (1a). Since the push button becomes high, it triggers the processing unit to activate the camera (1b), so it captures a photo of the check (2a). The captured image is then forwarded to the processing unit (2b), and passed to a check segmentation block where four segments are extracted (ID Number Segment, Signature Segment, Money Number Segment, & Money Text Segment) (3). The ID Number Segment Pre-Processing & Segmentation block (4) do the image treatment for this image then segment the ID number on 20 digits. The ID Number Digits Prediction block (5) take each digit image and pass it through the pre-trained ID Number CNN .h5 model to identify each digit (It can be 0, 1, 2 ... or 9), then all the predicted digits are arranged to form the predicted ID number which will be forwarded to the processing unit (6a) to access the users database (6b) and check if the predicted ID is a valid user, so if the predicted ID isn't for a valid user, the Red LED will light (6c) to indicate that the money withdraw process is denied; but if the predicted ID is for a valid user, The Signature Segment Pre-Processing block (7) will treat the signature image, then the Signature Image Prediction block (8) take the signature image and pass it through the pre-trained Signature CNN .h5 model to get the probability of this signature of being related to the user of the previously predicted ID, then this probability number is forwarded to the processing unit (9a) to see if it's above a pre-defined threshold (The threshold eliminate the possibility if fake signatures to be validated, our system threshold is 97%, which means only signatures with an accuracy of 97% and more are validated), if the signature isn't valid, the Red LED will light (9b) to indicate that the money withdraw process is denied; but if the signature is valid, then the Money Number Segment Pre-Processing & Segmentation block (10) will be treated and the digits composing the handwritten number will be segmented, then the Money Number Digits Prediction block (11) take each digit image and pass it through the pre-trained Money Number CNN .h5 model to identify each digit (It can be 0, 1, 2 ... or 9), then all the predicted digits are arranged to form the predicted Money number which will be forwarded to the processing unit (12), after that the Money Text Segment Pre-

Processing & Segmentation block (13) will be treated and each black pixels that are linked together will be segmented individually, then the Money Text Digits Prediction block (14) take each segment and pass it through the pre-trained Money Text CNN .h5 model to identify its class, then all the predicted segments are arranged to form the predicted Money phrase which will be translated to a number, then the Money Text value will be forwarded to the processing unit (15a), for it to compare it with the Money Number value, and if both Money Number and Money Text aren't equal, the Red LED will light (15b) to indicate that the money withdraw process is denied, but if they are equal then the users database is accessed (15c) to make sure the amount of money to withdraw doesn't exceed the current users balance, so if it does exceed the current balance then the Red LED will light (15b) to indicate that the money withdraw process is denied, but if not then the current balance is updated and the Blue LED will light (16) to indicate that the money withdraw process is accepted.

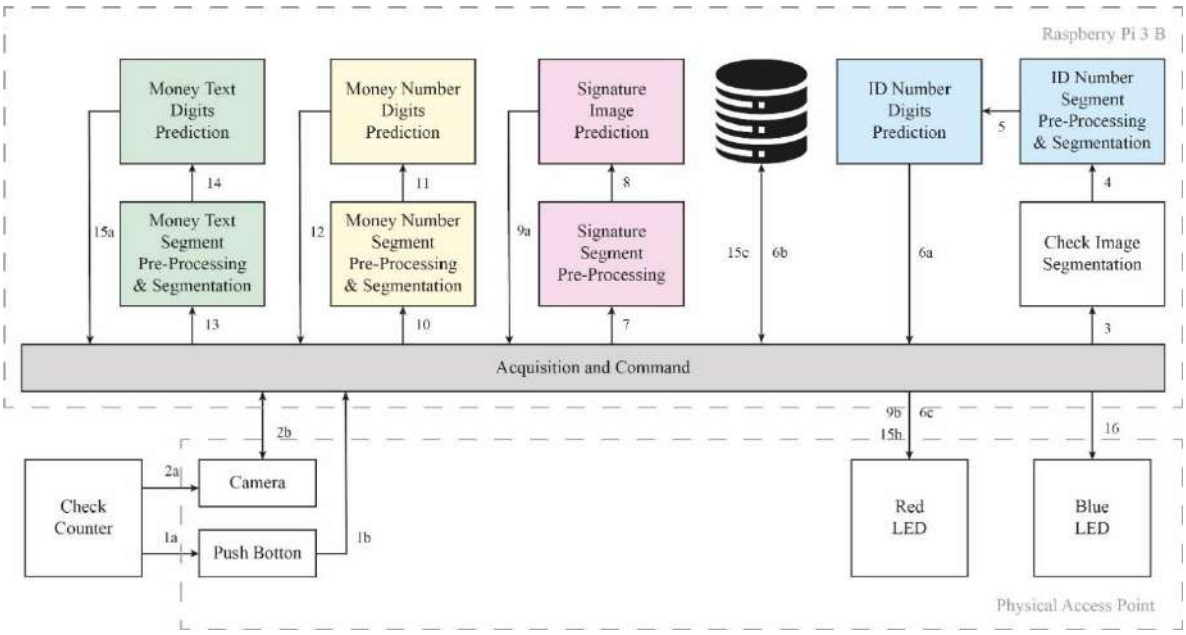


Figure IV-1 Global System Flow Diagram

IV.4 Check Verification Model

Before we started our experiments on building a valid check verification system, we needed to identify the important check features that must be verified for the user to withdraw money, and our reasoning was to:

- Check the user ID number since it's the key for us to identify if the user belongs to the client system (Therefore, identify the user's Full name in Latin and Arabic, and other financial information).
- Verify if the written signature belongs to the user.
- Identify the written amount of money in numbers format the user want to withdraw.
- Identify the written amount of money in text format the user wants to withdraw.

After the segmentation phase, our check verification system will depend on four different CNN models, to verify the mentioned four check features:

IV.4.1 ID Number Verification CNN

This CNN aims to identify all the 20 digits composing the ID number, then accessing the user's database to verify the existence of the tested check ID number. The success of this model depends on the steps in the figure IV-2:

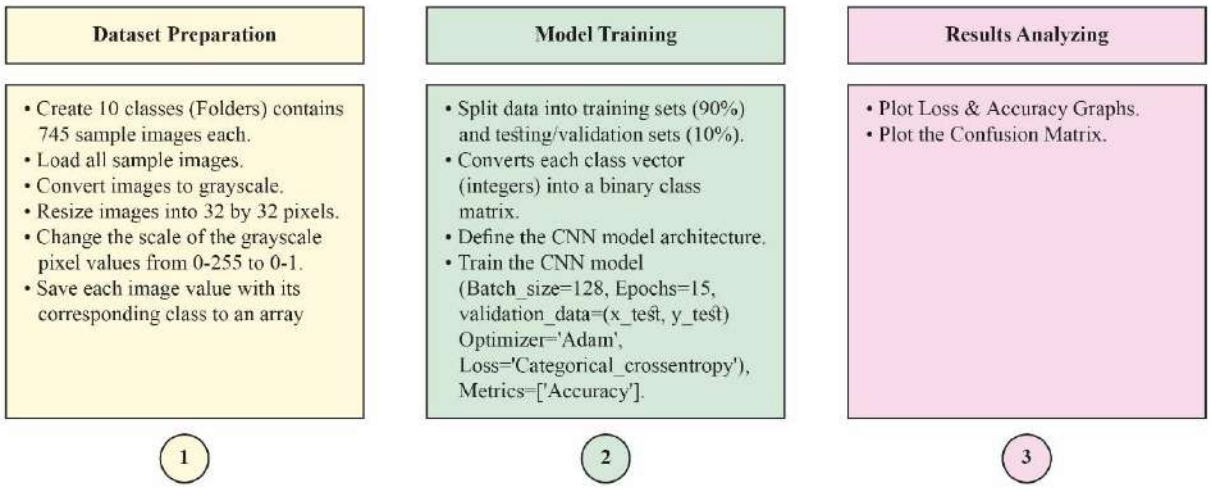


Figure IV-2 ID Number CNN Training Flow Diagram

IV.4.1.1 Dataset

Since we aim to identify a series of printed digits (it can be one of the following: 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9) we used a digit's samples from an existing dataset, namely Chars74K dataset [57], we have 10 classes, each class contains 745 images.

After collecting enough samples for training, we pre-process each image to the right dimensions and format, for our model we decided to scale all training data to 32 by 32 pixels grayscale images, then save each image with its corresponding class in an array.

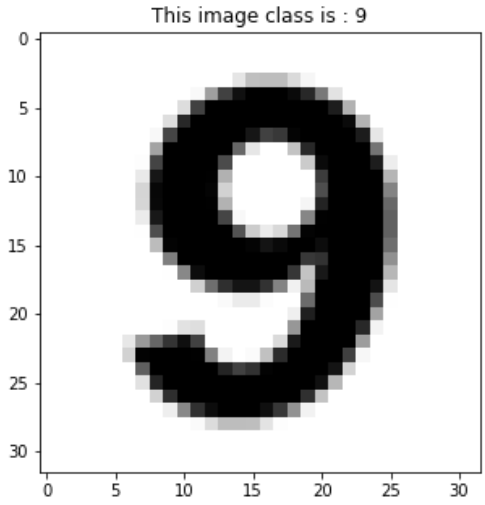


Figure IV-3 ID Number CNN Dataset Sample Image After Pre-Processing

IV.4.1.2 Training

All the 4 CNN models are developed in Keras on top of TensorFlow. The decision to use Keras was made because it supports convolution networks and its ease of use.

When defining the network layer configuration, it is important to note that there is no exact formula for optimal layer configuration. Instead, the effective approach is trial and error, where a few layers configuration is explored and benchmarked to see how well they work. However, for this ID Number CNN, the 1st configuration we tested yielded the best results which proves that our research was solid since we inspired our model architecture from a model shared on Kaggle used for Sudoku game digits classification [58].

We used the CPU based computation option to train this model since it's not a complex model, the epoch was 15 and the batch size was 128.

The labelled dataset was divided into two, where 90% of the data is used to train the network, and 10% is used to validate the network. After a sufficient validation accuracy is achieved, additional test sets were used to see how well the network performs.

Table IV-1 ID Number CNN Configuration

Architectures #1		
Convolution Filter Filter 5*5 32		
Convolution Filter Filter 3*3 16		
Dropout 25%		
Flatten		
Dense 128		
Dense 50		
BatchNormalization		
Softmax		
Precision (%)	Loss (%)	Execution Time
100%	0.26%	11ms

IV.4.1.3 Results

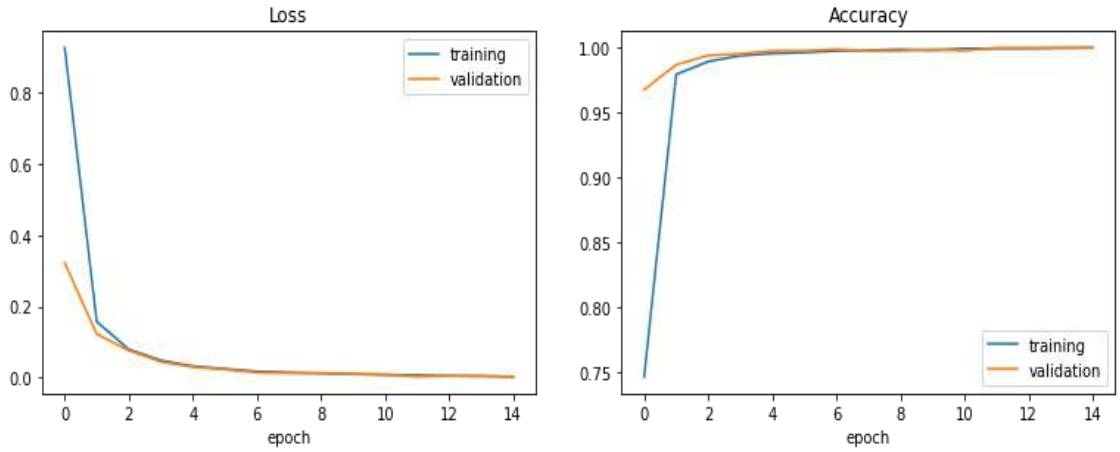


Figure IV-4 ID Number CNN Validation Loss and Accuracy Results

The accuracy rate is 100% which represents the ultimate result we can get for just 15 epochs and a training time of 84s.

Classification accuracy alone can be misleading since our data has more than 2 classes, we may have an accuracy of 100% but we aren't sure if all classes are being predicted equally well.

Calculating a confusion matrix should give us a better idea on our model's pros and cons.

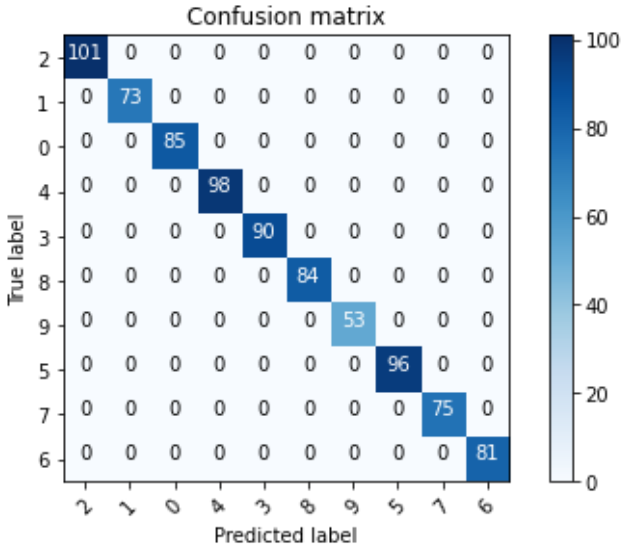


Figure IV-5 ID Number Model Confusion Matrix

The diagonal elements represent where the predicted label is the same as the true label, while off-diagonal elements represent the mislabeled elements by our classifier. The higher the diagonal values the better the model is, which is the case in our model.

The off-diagonal elements refer to the unbalance of our model. For our model, the misclassification is zero. Thus, our model is accurate and balanced.

IV.4.2 Signature Verification CNN

The aim of this CNN is to verify if the written signature is for the user identified with the previously tested ID number, the success of this model depends on the steps in the figure IV-7:

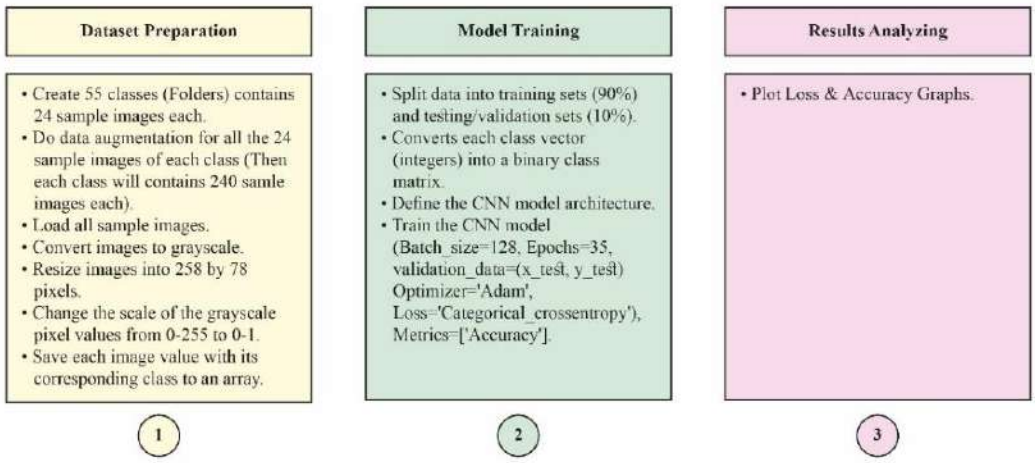
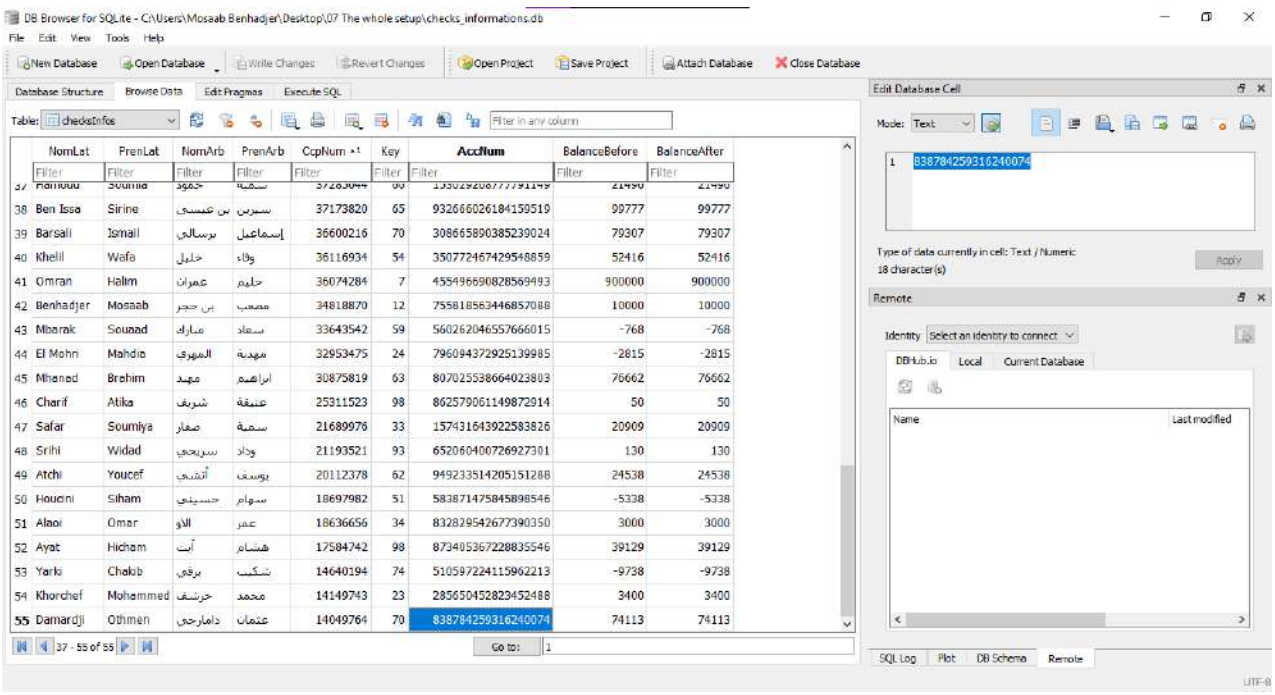


Figure IV-6 Signature CNN Training Flow Diagram

IV.4.2.1 Dataset

Since we aim to identify the user signature, we 1st created an SQLite database with 55 users (Each user have a full name written in both Latin and Arabic, a PCC number, a Key number, an Account number "known as the ID number", and a balance before & after state numbers), then we used each user's ID number as the name of his/her signature class; each class contains about 240 sample images; for each class we started with 24 images we token from an existing database, namely CEDAR dataset [59], then we used data augmentation techniques (Scaling, Translation, Rotation, & Lighting condition) to rise our samples to about 240 sample images for each class.



	NomLat	PrenLat	NomArb	PrenArb	CcpNum	Key	AccNum	BalanceBefore	BalanceAfter
38	Ben Issa	Sirine	بن عيسى	سيرين	37173820	65	932666026184159519	99777	99777
39	Barsali	Ismail	برسالي	إسماعيل	36600216	70	308665890385239024	79307	79307
40	Khelil	Wafa	خليل	وفاء	36116934	54	350772467429548859	52416	52416
41	Omran	Halim	عمران	حليم	36074284	7	4554966908285694 93	900000	900000
42	Benhadjer	Mosaab	بن حجر	مصعب	34818870	12	755818563446857088	10000	10000
43	Mbarak	Souaad	مبارك	سعاد	33643542	59	560262046557666015	-768	-768
44	El Mohri	Mahdia	المهري	مهدية	32953475	24	796094372925139985	-2815	-2815
45	Mhaned	Brahim	مهند	ابراهيم	30875819	63	807025538664023803	76662	76662
46	Charif	Atika	شريف	عتيقة	25311523	98	8625790611498729 14	50	50
47	Safar	Soumiya	صفار	سمية	21689976	33	1574316439225838 26	20909	20909
48	Srihi	Widad	سريحي	وداد	21193521	93	652060400726927301	130	130
49	Atchi	Youcef	أتشي	يوسف	20112378	62	949233514205151288	24538	24538
50	Houcini	Siham	حسيني	سهام	18697982	51	583871475845898546	-5338	-5338
51	Alaoi	Omar	الاو	عمر	18636656	34	832829542677390350	3000	3000
52	Ayat	Hicham	آيت	هشام	17584742	98	873405367228835546	39129	39129
53	Yarki	Chakib	برقي	شكيب	14640194	74	5105972241159622 13	-9738	-9738
54	Khorchef	Mohammed	خرشف	محمد	14149743	23	2856504528234524 88	3400	3400
55	Damardji	Othmen	دامارجي	عثمان	14049764	70	838784259316240074	74113	74113

Figure IV-7 SQLite based Users Database

After getting enough samples for training, we pre-process each image to the right dimensions and format, for our model we decided to scale all training data to 258 by 78 pixels grayscale images, then save each image with its corresponding class in an array.

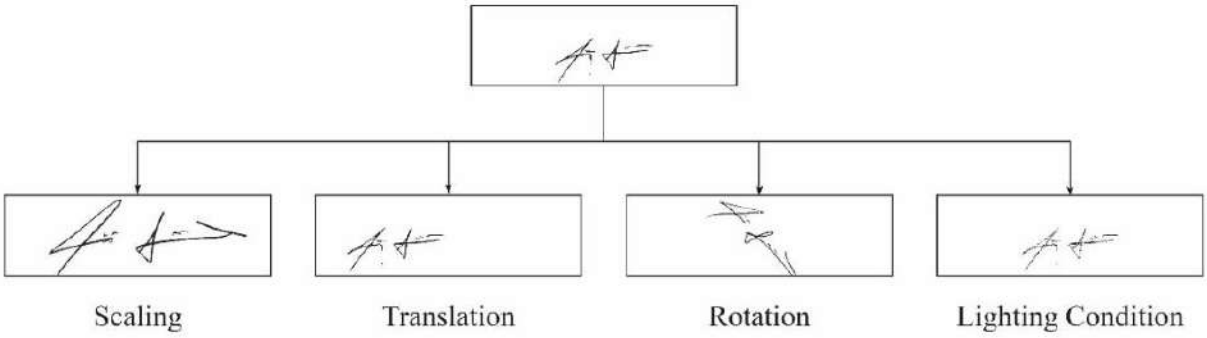


Figure IV-8 Signature Sample Images after Data Augmentation

IV.4.2.2 Training

After our trial-and-error improvement of the signature verification CNN, we will show only the configuration that yielded the best results before comparing it to a Signature CNN model architecture posted in a journal article [60].

For our developed model, we used Google's cloud-based CPU to train this model since our laptop can't handle the complexity of this model. Still, for the CNN model derived from the journal article, we used a CPU-based computation option to train this model since we figured out a better PC machine than our laptop.

The labeled dataset was divided into two, where 90% of the data is used to train the network, and 10% of the data is used to validate the network. After a sufficient validation accuracy is achieved, an additional tests set were used to see how well the network performs.

Since we are dealing with multiple classes for outputs, the categorical cross entropy loss function was used. The architecture that yielded the best results is architecture 3.

Table IV-2 Signature CNN Configurations

Architectures #2			**Architectures #3**		
Convolution Filter Filter 5*5 32			Convolution Filter Filter 5*5 32		
MaxPooling			MaxPooling		
Convolution Filter Filter 5*5 32			Convolution Filter Filter 5*5 64		
MaxPooling			MaxPooling		
Convolution Filter Filter 5*5 64			Convolution Filter Filter 5*5 128		
MaxPooling			MaxPooling		
Convolution Filter Filter 5*5 64			Convolution Filter Filter 5*5 256		
MaxPooling			MaxPooling		
Convolution Filter Filter 5*5 128			Convolution Filter Filter 5*5 512		
MaxPooling			MaxPooling		
Convolution Filter Filter 5*5 256					
MaxPooling					
Dropout 25%			Dropout 25%		
Flatten			Flatten		
Dense 128			Dense 256		
Dense 64			Dense 60		
Softmax			Softmax		
Precision (%)	Loss (%)	Execution Time	Precision (%)	Loss (%)	Execution Time
96.39%	16.53%	16ms	98.61%	6.14%	17ms

IV.4.2.3 Results

We will be discussing the results from the architecture #3 compared to architecture #2.

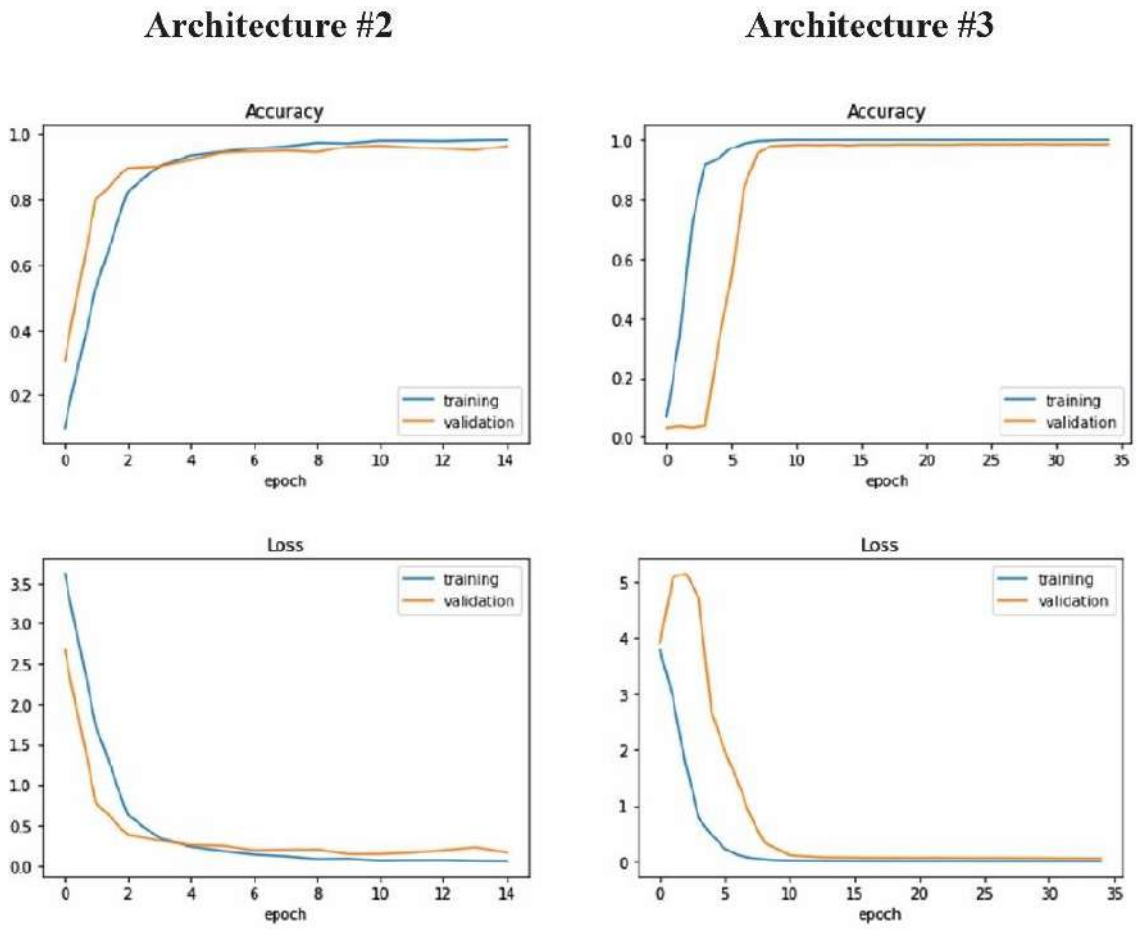


Figure IV-9 Signature Training Loss and Accuracy Results

For architectures #2 and #3, the accuracy plots are accelerating well, and neither is overfitting nor underfitting. Architecture #3 plots are a more stable (almost no vibrations) compared to architecture #2; architecture #3 is 2% better in accuracy, 10% better in Loss, and quite the same in the execution time (1ms time difference) compared to architecture #2, so it's evident for us to go with architecture #3.

IV.4.3 Money Number Verification CNN

The aim of this CNN is to identify all the handwritten digits composing the number of the amount of money the user wants to withdraw, the success of this model depends on the steps in the figure IV-2:

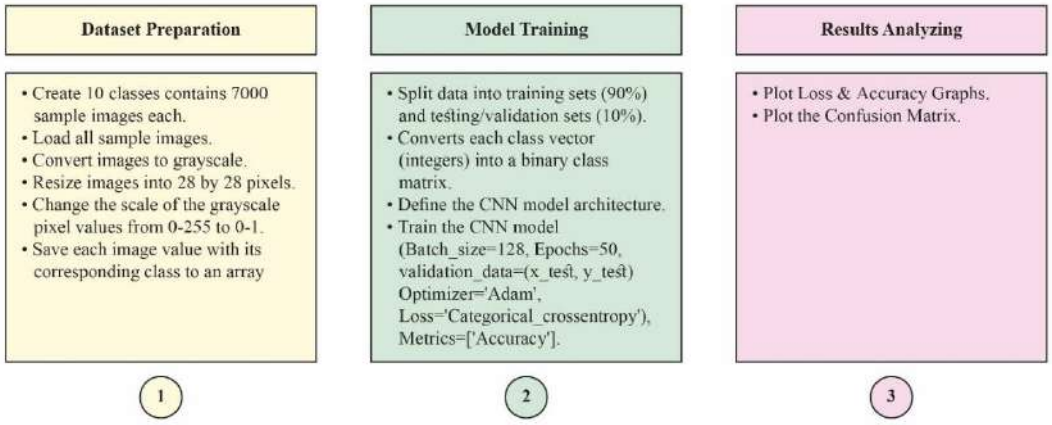


Figure IV-10 Money Number CNN Training Flow Diagram

IV.4.3.1 Dataset

Since we aim to identify a series of handwritten digits (it can be one of the following: 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9) we used a famous dataset, namely MNIST dataset [61], this dataset consists of 70000 small square grayscale images of handwritten single digits between 0 and 9 (10 classes).

After loading the training samples, we pre-process each image to the right dimensions and format, for our model we decided to scale all training data to 28 by 28 pixels grayscale images, then save each image with its corresponding class in an array.

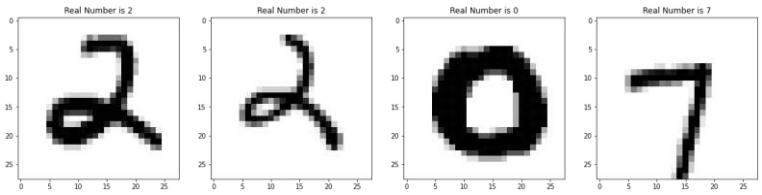


Figure IV-11 Money Number Dataset Sample Images after Pre-Processing

IV.4.3.2 Training

For this Money Number CNN, the 1st configuration we tested yielded the best results, since many people worked on enhancing MNIST based CNN model accuracy we used an existing architecture posted in Kaggle [62].

We used the CPU based computation option to train this, the epoch was 50 and the batch size was 128.

The labeled dataset was divided into two, where 90% of the data is used to train the network, and 10% of the data is used to validate the network. After a sufficient validation accuracy is achieved, an additional tests set were used to see how well the network performs.

Table IV-3 Money Number CNN Configuration

Architectures #4		
Convolution Filter Filter 3*3 64		
Convolution Filter Filter 3*3 64		
MaxPooling		
BatchNormalization		
Convolution Filter Filter 3*3 128		
Convolution Filter Filter 3*3 128		
MaxPooling		
BatchNormalization		
Convolution Filter Filter 3*3 256		
MaxPooling		
BatchNormalization		
Flatten		
Dense 512		
Softmax		
Precision (%)	Loss (%)	Execution Time
99.28%	3.48%	14ms

IV.4.3.3 Results

The accuracy rate is 99.28% which is a great result. Classification accuracy alone can be misleading since our data has more than 2 classes, we may have an accuracy of 99.28% but we aren't sure if all classes are being predicted equally well.

Calculating a confusion matrix should give us a better idea on our model's pros and cons.

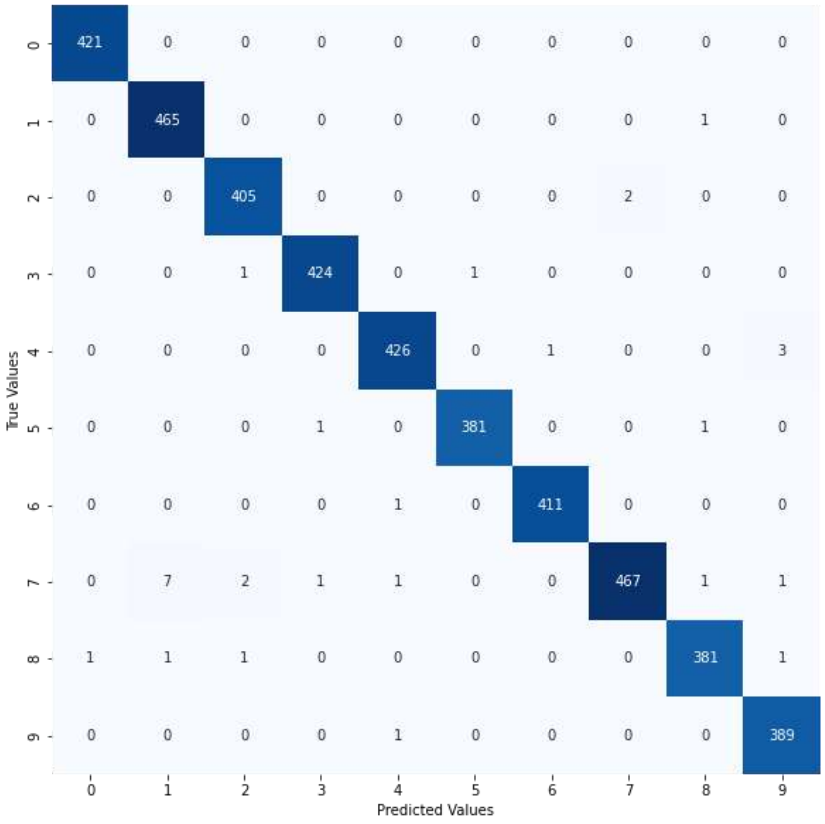


Figure IV-12 Money Number Confusion Matrix for architecture #4

Our model did misclassify some samples, but since they are very minimal, we can say that our model is not only accurate but also precise and balanced.

IV.4.4 Money Text Verification CNN

The aim of this CNN is to identify the Arabic handwritten phrase referring to the amount of money the user wants to withdraw, then translate it into a number so we can compare it with the result of the previous CNN (Money number CNN), so since we want to identify a phrase, we will

1st need to identify each word in the phrase (The Arabic words that can be used to describe the amount of money the user can withdraw are 73), so to identify each word used an approach based on OpenCV called Connected component labeling, which is an algorithmic application of graph theory used to determine the connectivity of "blob"-like regions in a binary image, in other words we are going to divide each word based on its main parts where the pixels are connected together (Ex: the word ثمانية, will be divided into 2 sub-words: ثما and نية); after we applied this approach on the 73 words that can be used, we come up with a 63 sub-words (Classes), that will be used to compose the right words, so those words will be composed to come up with the written phrase and thus translate that phrase into a number, and finally compare this number with the previously identified number from the Money Number CNN; the success of this model depends on the steps in the figure IV-14:

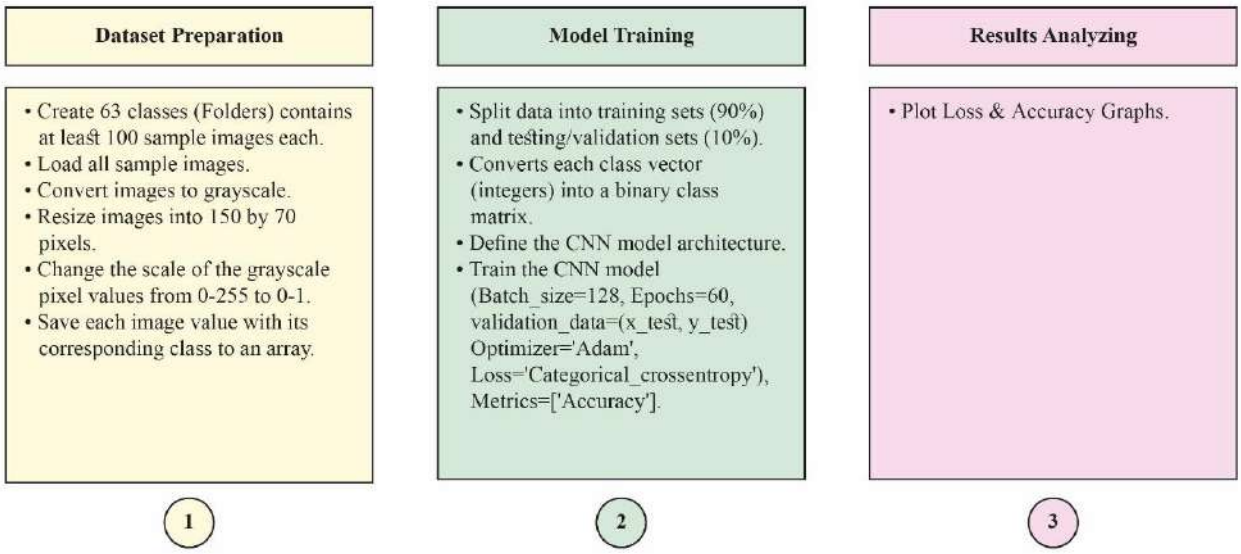


Figure IV-13 Money Text CNN Training Flow Diagram

IV.4.4.1 Dataset

Since we aim to identify a handwritten phrase, we first need to identify the words composing this phrase, part of the words we used to create our dataset were created by Master students in 2019 [63] and the other part we created, then we extracted the main parts of each word to create the needed dataset that has 63 class, and each class contains at least 100 samples.

After collecting enough samples for training, we pre-process each image to the right dimensions and format, for our model we decided to scale all training data to 150 by 70 pixels grayscale images, then save each image with its corresponding class in an array.

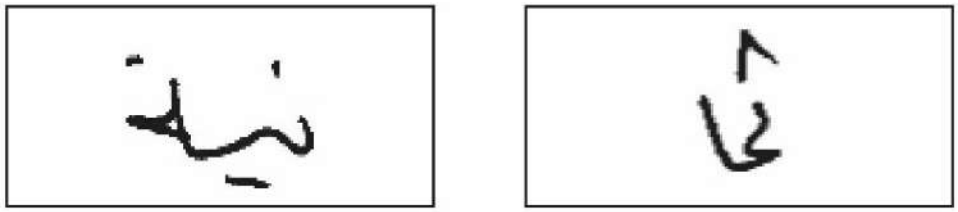


Figure IV-14 Money Text Dataset Sample Images after Pre-Processing

IV.4.4.2 Training

For this Money Text CNN, we tested two CNN architectures: architecture #5 is for the best model a master student gets when working on a handwritten Arabic image's identification [63]. and we compared it with the architecture #6 that we used in the signature verification.

We used the CPU based computation option to train model #5, the epoch was 60 and the batch size was 128.

The labeled dataset was divided into two, where 90% of the data is used to train the network, and 10% of the data is used to validate the network. After a sufficient validation accuracy is achieved, an additional tests set were used to see how well the network performs.

Table IV-4 Money Text CNN Configurations

Architectures #5			**Architectures #6**		
Convolution Filter Filter 7*7 32			Convolution Filter Filter 5*5 32		
Convolution Filter Filter 7*7 32			MaxPooling		
MaxPooling			Convolution Filter Filter 5*5 64		
Convolution Filter Filter 7*7 64			MaxPooling		
Convolution Filter Filter 7*7 64			Convolution Filter Filter 5*5 128		
MaxPooling			MaxPooling		
Convolution Filter Filter 7*7 128			Convolution Filter Filter 5*5 256		
MaxPooling			MaxPooling		
Dropout 25%			Convolution Filter Filter 5*5 512		
			MaxPooling		
			Dropout 25%		
Flatten			Flatten		
Dense 128			Dense 256		
Dropout 50%			Dense 60		
Softmax			Softmax		
Precision (%)	Loss (%)	Execution Time	Precision (%)	Loss (%)	Execution Time
93.33%	30.56%	16ms	96.35%	12.96%	14ms

IV.4.4.3 Results

We will be discussing the results from the architecture #5 compared to architecture #6.

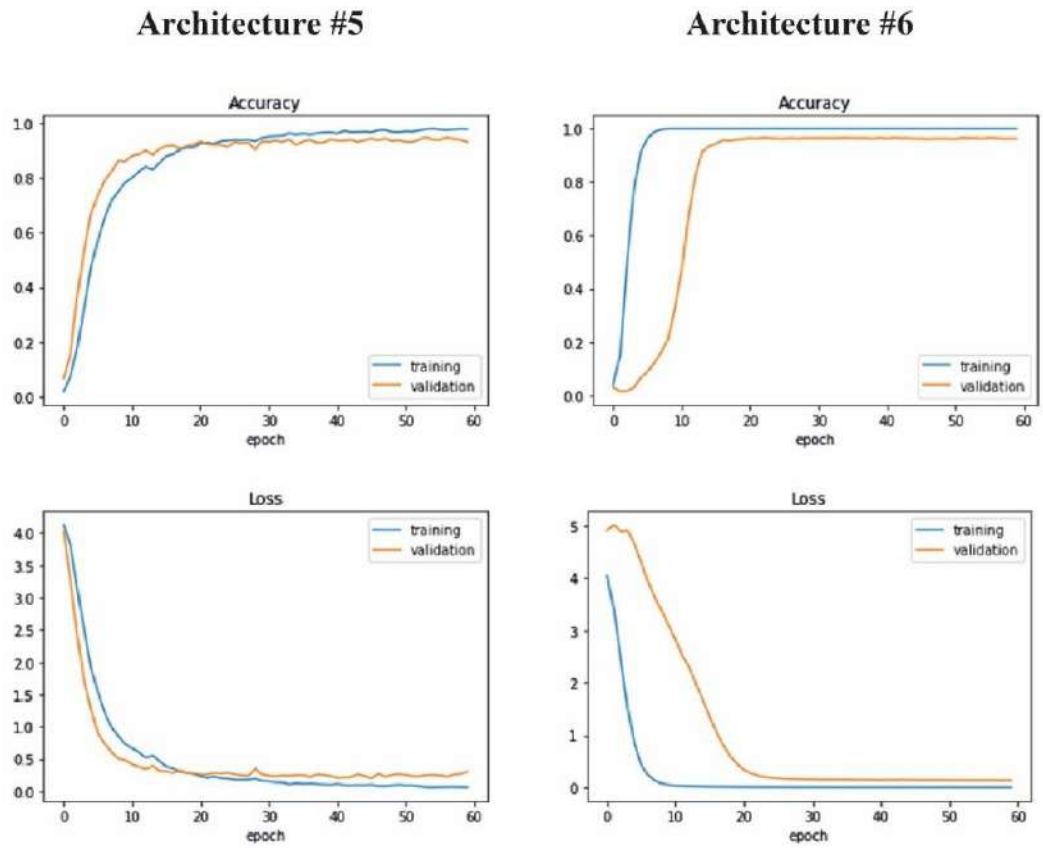


Figure IV-15 Money Text Training Loss and Accuracy Results

For both architecture #5 and #6 the accuracy plots are accelerating well, and no one of them is overfitting or underfitting, architecture #6 plots are a more stable (almost no vibrations) compared to architecture #5, architecture #6 is 3% better in accuracy, 17.5% better in Loss and quite the same in the execution time (2ms better) compared to architecture #5, so it's very clear for us to go with architecture #6.

IV.4.5 Discussion

In this section we have tried one or multiple CNN models to get the higher accuracy value and stability for our four CNN models, thus make the a good check recognition system, for our 1st CNN (ID number CNN) after we did a solid research and dataset preparation the 1st CNN model got us a perfect result (100% accuracy); for our 2nd CNN (Signature CNN) we followed the trial and error approach to come up with a good model but after extra research we found a CNN model that is dedicated for signatures that a team of researchers come up with that give us slightly better

results so we implemented in our system; for the 3rd CNN (Money Number CNN) it was an easy problem to solve since a lot of researchers enhanced the handwritten digits CNNs accuracy with the help of the MNIST Dataset, se we used one of the best models out there; for our 4th and final CNN (Money Text CNN) the challenge was to come up with a model with a good accuracy that deals with Arabic handwritten sentences, and since the used words for making these sentences are already known (73 words), we used OpenCV's connected component labeling algorithm to split each word into its main parts, the result was a 63 sub-words that we build our model based on, then for the models we tried: the 1st was for a Master students who developed a CNN model in 2019 for that case (Different dataset is used) and the 2nd model is the one we chosen for our signature identification, both models did well but the second one was the most stable and give us the best results.

Table IV-3 is a summary of the results we come up with.

Table IV-5 Comparison between the obtained CNN results

CNNs	Tested Architectures	Accuracy (%)	Loss (%)	Execution (ms)	Chosen Architecture
ID Number	Archi #1	100	0.26	11	Archi #1
Signature	Archi #2	96.39	16.53	16	Archi #3
	Archi #3	98.61	6.14	17	
Money Number	Archi #4	99.28	3.48	14	Archi #4
Money Text	Archi #5	93.33	30.56	16	Archi #6
	Archi #6	96.35	12.96	14	

IV.5 Implementation

We are set to start implementing our check verification system after developing the best CNN architectures to recognize the user's ID number, signature, and Money written as a sentence. First we created an administrator GUI for the financial institution's worker to add new user information to the SQLite users database, then built the user's check verification system to 1st take the check image then extracting the required segments and then test if: the user's ID is valid, the written signature belongs to the identified users, identify the amount of money the user want to

withdraw, the user can get the requested amount of cash based on his/her balance and finally inform the user if the check verification is confirmed or denied via a blue and red LEDs.

IV.5.1 Administration Graphical User Interface

Designing the administration graphical user interface is intuitive using the TKinter library that provides a higher level abstraction layer to create elements with no boiler-plate code, meaning everything was declared in immediate mode.

The administration worker needs to add new users to the users database, add the signature samples of the new user (Then data augmentation will be used to get 240 signature image samples out of 24), and to retain the model when a new user is added. To do so we created this GUI, the new user information te worker will add are:

- Name in Latine.
- Surname in Latine.
- Name in Arabic
- Surname in Arabic
- PCC number
- Key
- Account Number (ID number)
- Current Balance.

And after the information is typed into their specific areas, the worker must press on the "Save info" button to add the new user information to the database, the "Clear" button is used to clear what he did typed on the GUI.

For the worker to add the signature samples of the new user to the signature dataset, he will need to type the Client ID again and press on "Add Signature Images" button, so a new class (Floder)

with the name of the ID number will be created and the selected 24 signature images will be augmented into 240 image and put inside the created class folder.

Since a new user is added the worker must retrain the Signature CNN model by pressing the "Train Sig CNN" button.



Figure IV-16 Administration Graphical User Interface

IV.5.2 Check Verification

When a user approaches the cash counter to withdraw money, after filling the check with his/her name, the amount of money to withdraw in number and Arabic text and sign it, the next step is to press the push botton so the check verification system will proceed:

For our system to extract the four mentioned check features the following steps were executed:

1. Preparing the input image:
    - Take a picture of the check on a black background.



Figure IV-17 Check Segmentation Part 1

- Invert the taken image colors.



Figure IV-18 Check Segmentation Part 2

- Convert the inverted image into a binary image.

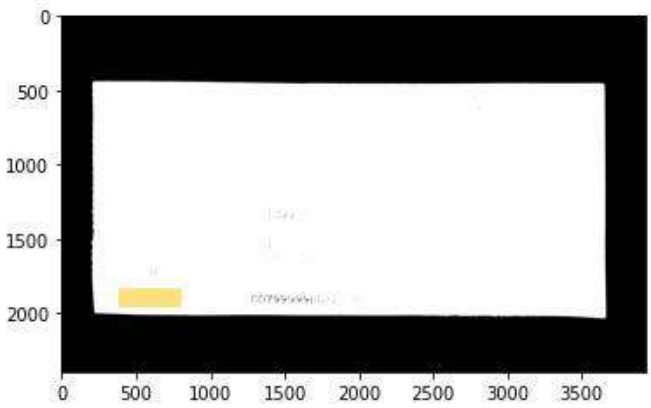


Figure IV-19 Check Segmentation Part 3

2. Isolating the foreground (check) from the input image:
    - Locate the check's contour using OpenCv Contour Detection technique.



Figure IV-20 Check Segmentation Part 4

- Get the check's top-left and bottom-right coordinates.
- Cut the foreground (check) from the input image based on the defined coordinates.



Figure IV-21 Check Segmentation Part 5

3. Isolating specific segments of the check to be feeded into their CNNs later:
    - Prepare the masks of each segment, using a design software.



Figure IV-22 Check Segmentation Part 6

- Merge each mask with the foreground (check) image.



Figure IV-23 Check Segmentation Part 7

- Crop each segment based on the coordinates of the non-isolated part of the mask.

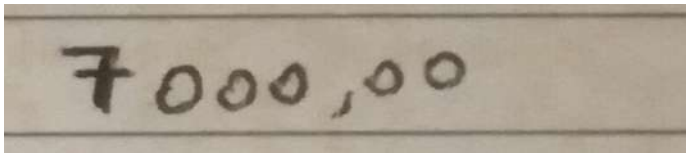


Figure IV-24 Check Segmentation Part 8

The next step will be verified if the check ID number is valid by predicting it based on ID Number CNN model and check if it exists in the user's database as the figure IV-18 explains it details.

If the ID number is valid the written signature will be identified based on the Signature CNN model, so if the similarity between it and the original user signatures passed a threshold of 97%, as the figure IV-19 explains in details, then next step will proceed.

Now the Money number will be predicted based on the Money Number CNN model, as the figure IV-19 explains in details.

After identifying the amount of money written in digits, the next step will be identifying the amount of money written as an Arabic sentence based on the Money Text CNN model, and finally the two number will compare to make sure they are equal, if so then this number must be less or equal to the user current balance for him/her to withdraw the money, as the figure IV-20 explains in details.

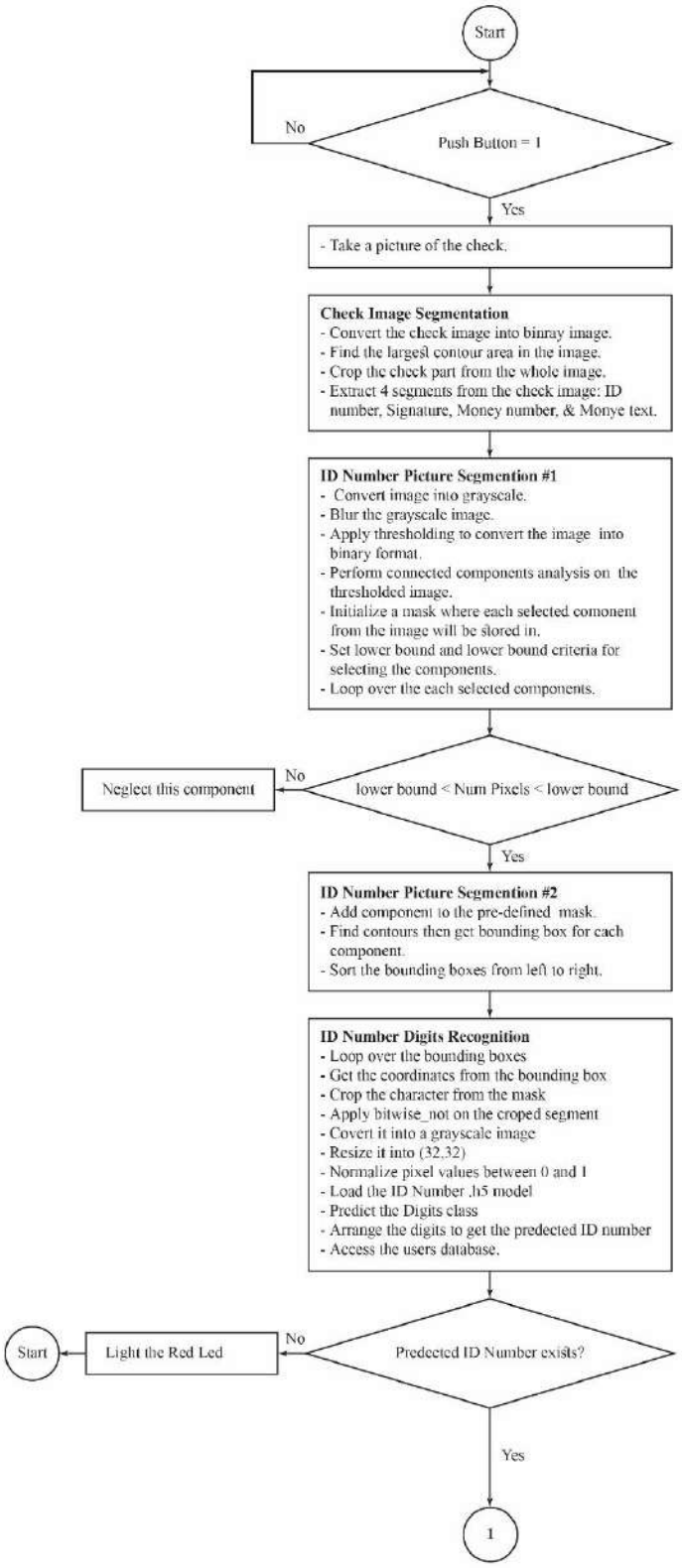


Figure IV-25 Detailed Flowchart of the Check Verification System Part 1

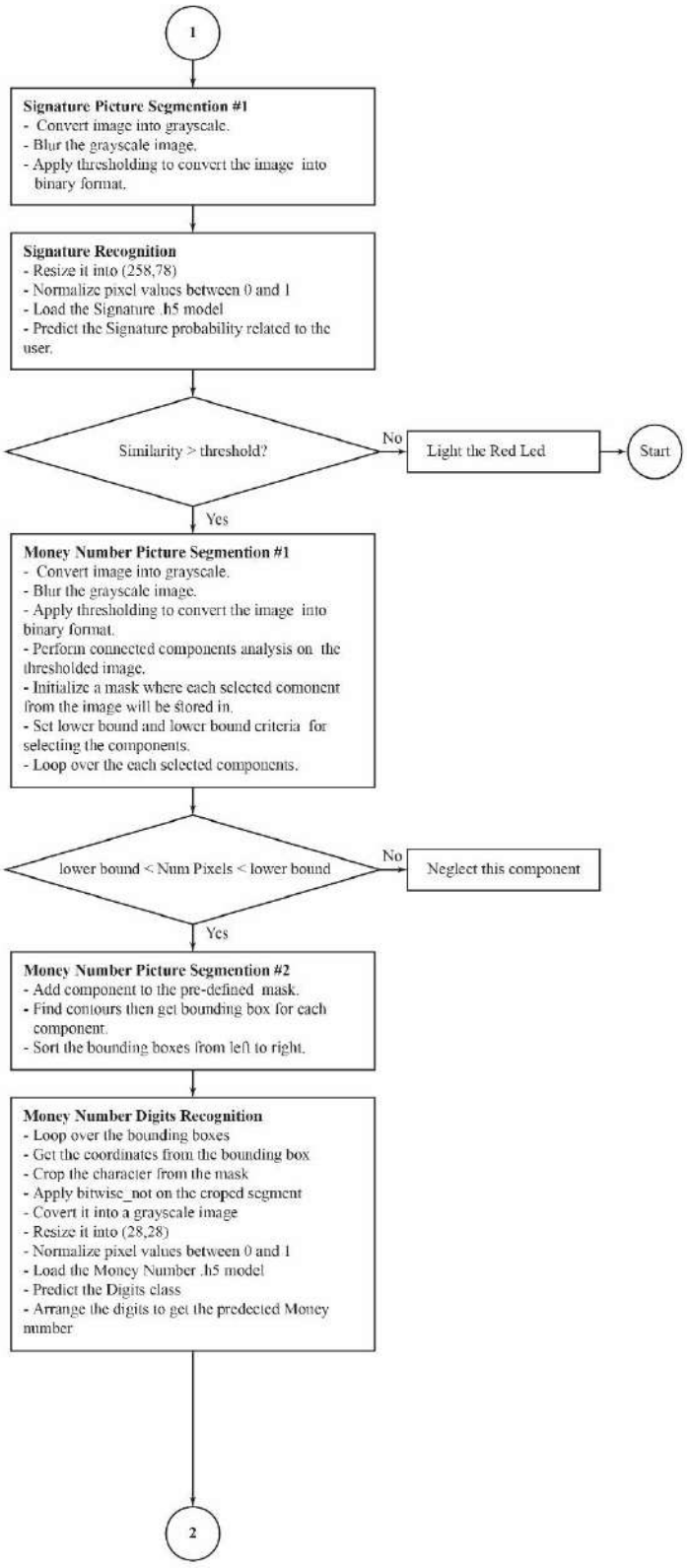


Figure IV-26 Detailed Flowchart of the Check Verification System Part 2

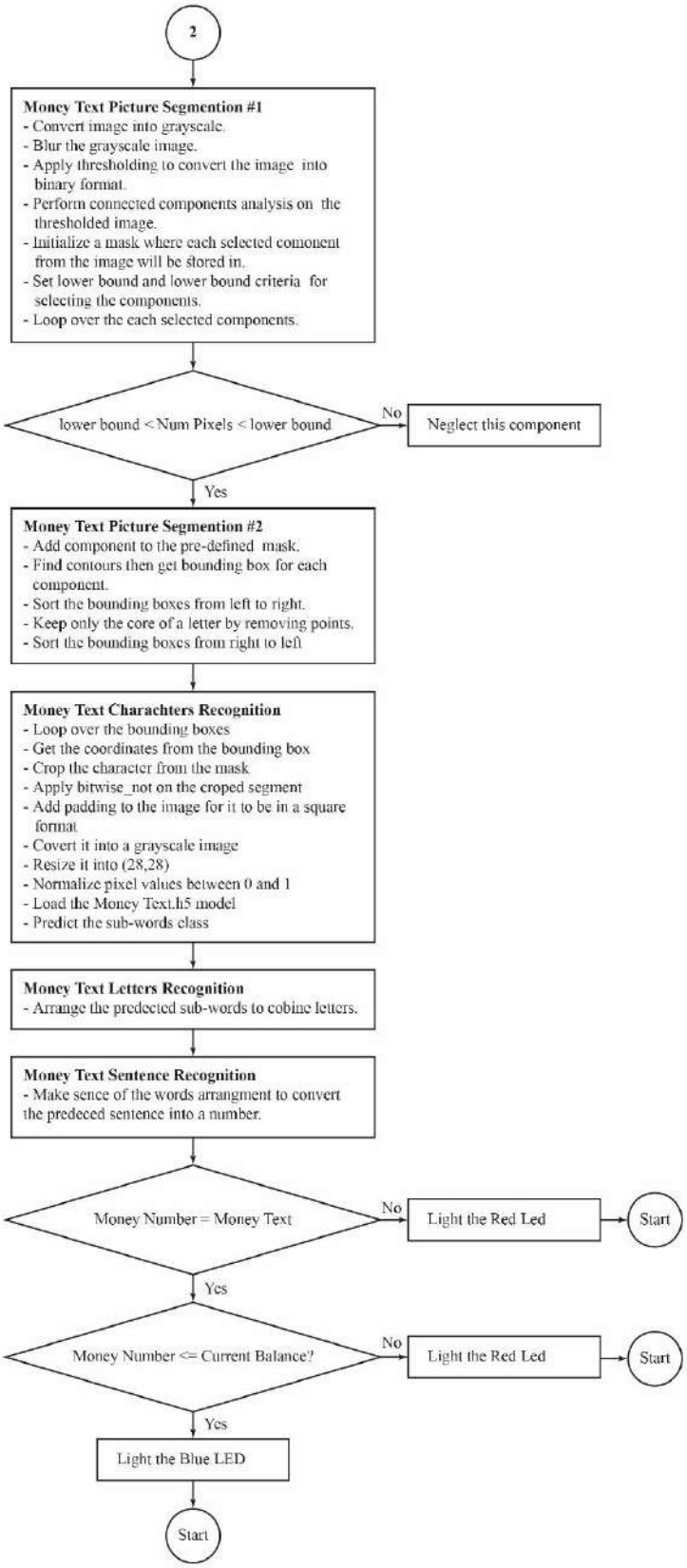


Figure IV-27 Detailed Flowchart of the Check Verification System Part 3

IV.5.3 Hardware Implementation

Finally, that we have all the software side figured, we pass to integrating it with hardware to finalize the product (the entire system). We run our system on a raspberry pi 3 Model B, and give remote access to our Android smartphone camera for it to take the check picture.

To start the check verification, process a push button is wired to GPIO pin 22, and for the user to know if the check verification process was accepted (Blue LED) or denied (Red LED), the Blue led is wired to the GPIO pin 17 through a 200-ohm resistor, and the Red led is wired to the GPIO pin 27 through a 200-ohm resistor too.

The 10k-ohm resistor wired to the push botton is a pull-down resistor to ensure our push button is pulled to a low logical level in the absence of an input signal.

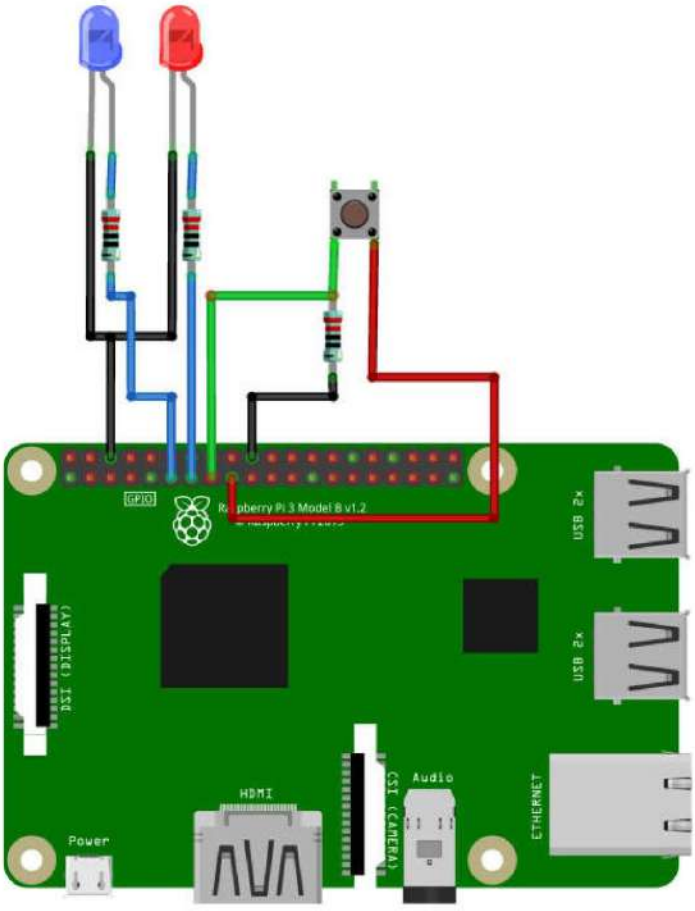


Figure IV-28Electrical wiring diagram of our system

IV.5.4 Final Product Assembly

To accomplish this, we used the materials discussed earlier in Chapter 3 and then followed in instructions stated in the sub-subsection above (IV.5.3). (Figure IV.22) depicts the entire assembly of our product.



Figure IV-29 Assembled Check Verification System

IV.5.5 Results

Now that the solution is finalized, we can showcase its functionality. We fill in a check (User's full name, amount of money to withdraw as a number and an Arabic sentence, and signature), then we put in on the right black area and press the push button to start the check verification process. After the segmentation of our check the 4 check feature segments (ID Number, Signature, Money Number, and Money Text) are verified based on the pre-trained CNN models, since we fill in the check with the right information related to what is in the user's database, the Blue LED light after about 9s to indicate that our check verification is accepted. Of course after we will get the requested money, our balance will be updated.

The next figures showcase in details the progress of the check verification process as it was described in (IV-5-2):

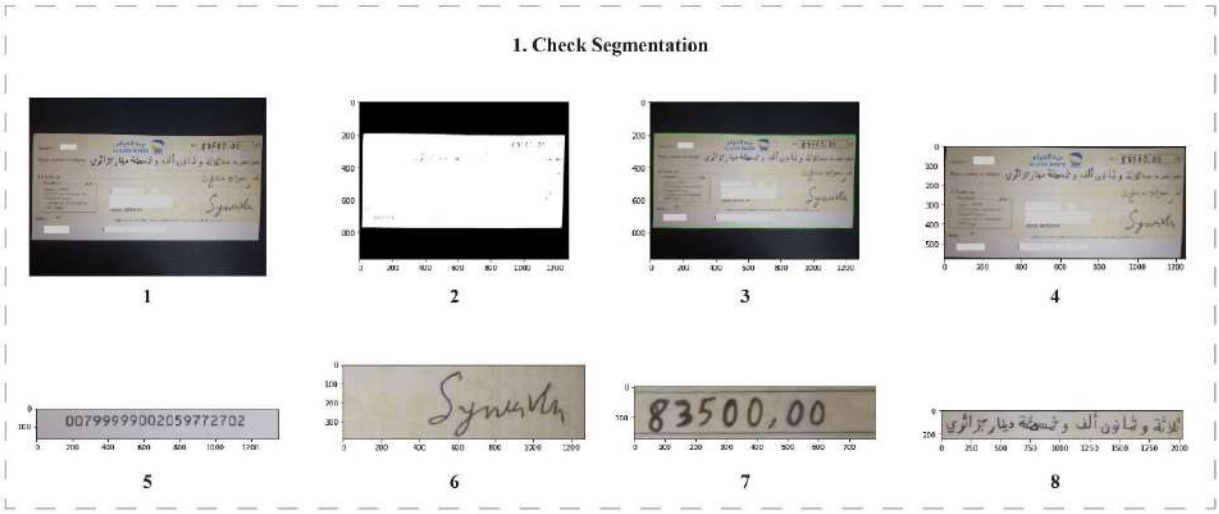


Figure IV-30 Check Segmentation Progress

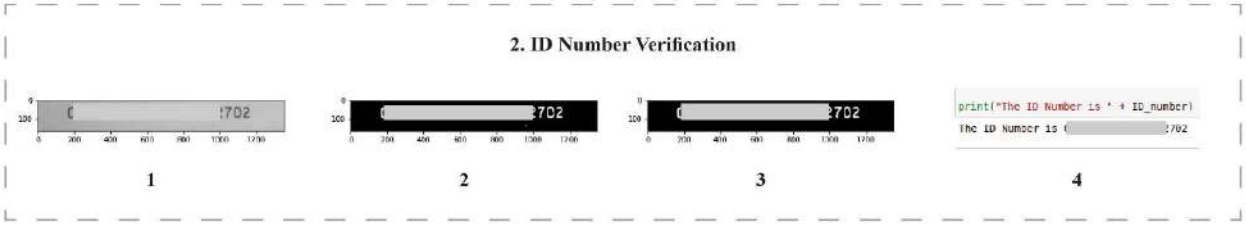


Figure IV-31 ID Number Verification Progress

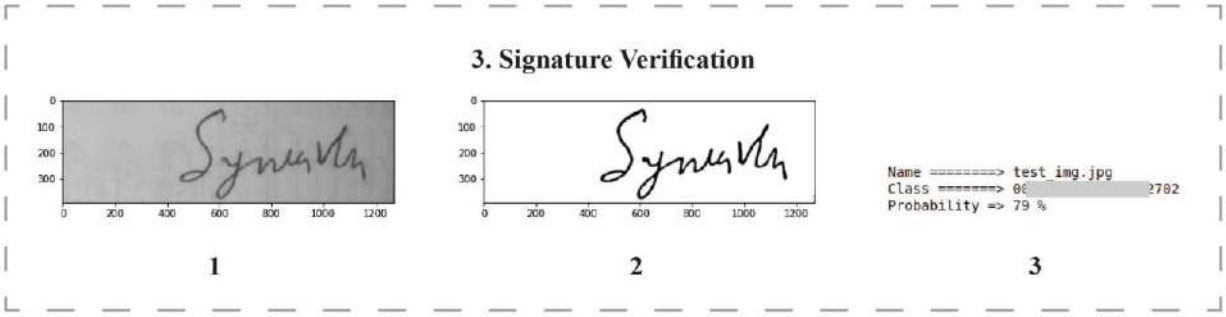


Figure IV-32 Signature Verification Progress

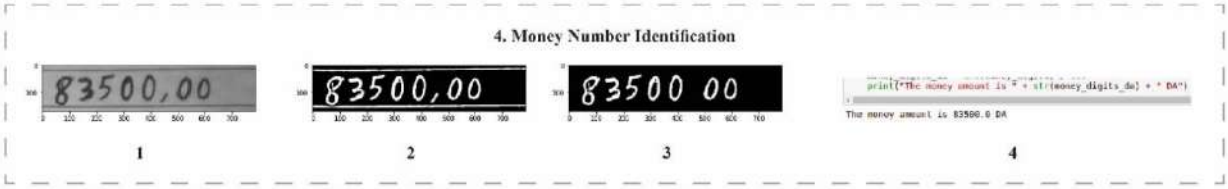


Figure IV-33 Money Number Identification Progress

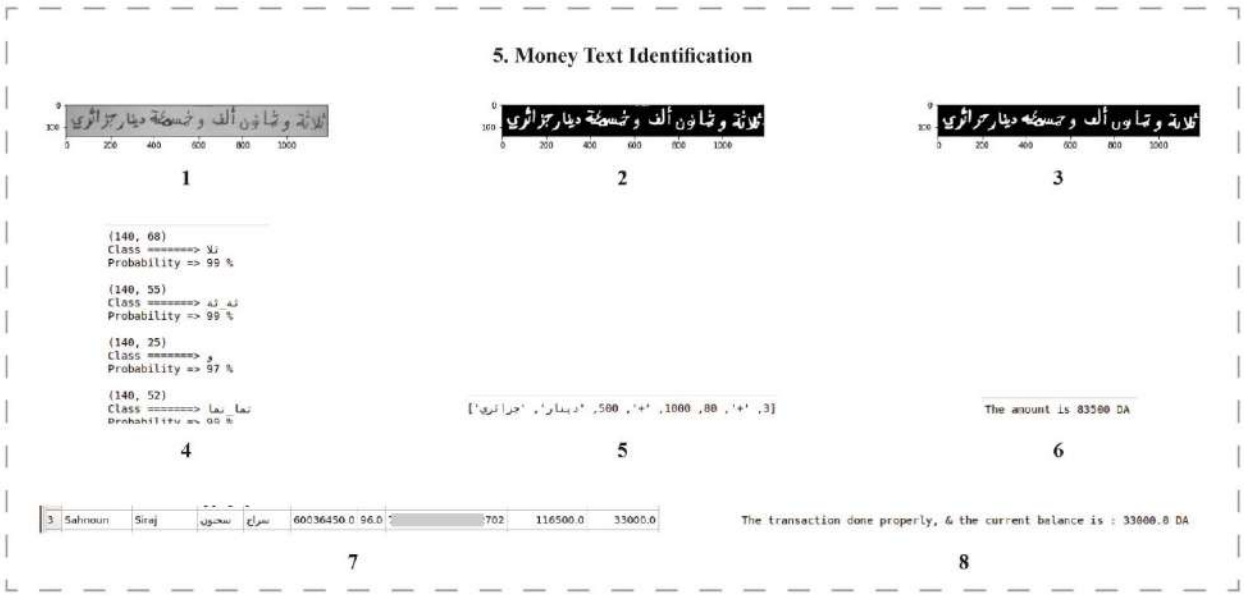


Figure IV-34 Money Text Identification Progress

Table IV-4 showcase the amount of time the whole and main parts of our system toke.

Table IV-6 Check Verification System Execution Time

The Main System Parts	The Main System Parts Execution Time (s)	The Whole System Execution Time (s)
Check Image Capturing	0.54	7.83
Check Segmentation	0.70	
ID Number Verification	0.26	
Signature Verification	0.70	
Money Number Identification	0.46	
Money Text Identification	5.17	

IV.6 Conclusion

In this chapter, we presented the detailed operation of our solution to realize an intelligent check reader system. This system is based on four convolutional neural networks. The objective we have set ourselves is to obtain the best performance in terms of precision and execution time. To do this we have implemented and evaluated several architectures. We started by a segmentation

phase then created a four CNNs to verify the most important check features, we even created a GUI to the financial institutions administration worker to create the user's database. And with the final architectures we have obtained an intelligent check reader system that can read any ID number with an accuracy of 100%, verify the user's signature with an accuracy of 98.61%, read the handwritten money number with an accuracy of 99.28% and finally read the Arabic handwritten phrase that represents the money number with an accuracy of 96.35%. Our systems performance has been demonstrated with the presentation of the check verification example that we have written by hand on a real check.

General Conclusion

In this thesis, we made an intelligent automation solution for check verification based on four convolution neural networks. We identified the check features that must be verified for the user to withdraw money. The first CNN verify the check ID number by recognizing the 20 digits composing it based on a CNN model with perfect accuracy of 100%. Then, users' existence in the financial institution's database will be verified. The second CNN verify if the signature belongs to the identified user when passing a threshold of 97% and it's based on a CNN model with a high accuracy of 98.61%. The third CNN identify the amount of money written in numbers the user wants to withdraw by recognizing the digits composing the full number based on a CNN model with a very height accuracy of 99.28%. The fourth CNN identifies the amount of money written in the Arabic sentence the user wants to withdraw by recognizing the sub-words composing the complete sentence based on a CNN model with very high accuracy of 96.35%.

After choosing the best CNNs we tested we moved to create a GUI that was made with Tkinter (in python), which allow the financial institution's administration worker to use add their clients/users information in, then for user part a push button was used for him to start the check verification process, the system used Raspberry pi 3 Model B as a base processing unit which in turn was connected remotely to a phone webcam and wirelessly to the mentioned push button and other two LEDS which indicates the state of the check Verification process (Accepted or denied), so if the check verification process is accepted a Blue LED will light and the user's balance will be updated since successfully now withdrawer money but if the check verification process is denied a Red LED will light.

This project allowed us to learn more about computer vision, specifically machine learning, it also gave us the chance to work with programming languages and libraries new to us, such as python, SQLiten Tkinter, OpenCv, Keras, and so on.

For future work we would like to develop this product even further by verifying the Arabic full name written on the check, also we would like to be able to verify the check when it's written in Latin, we would like also to verify if the check paper is original or not. Finally, we would also to add a face recognition feature to further enhance the security level even further.

Bibliography

[1] S. F. Bisbee, J. J. Moskowitz, E. R. Sheehan, D. H. Trotter and M. W. White, "Document authentication system and method," Google, 03 April 1997. [Online]. Available: https://patents.google.com/patent/WO1997012460A1/en. [Accessed 06 June 2022].

[2] R. H. Nagel, "System and method for production and authentication of original documents," Google, 19 December 2006. [Online]. Available: https://patents.google.com/patent/US7152047. [Accessed 06 June 2022].

[3] K. E. Moore, "Document authentication using the physical characteristics of underlying physical media," Google, 11 April 2006. [Online]. Available: https://patents.google.com/patent/US7028188. [Accessed 06 June 2022].

[4] I. Lancaster, F. Tuffy and J. Winchcombe, "Physical to Digital: A Revolution in Document Security | Reconnaissance Intl Online Store," [Online]. Available: https://estore.reconnaissance.net/product/physical-to-digital-a-revolution-in-document-security/. [Accessed 31 May 2022].

[5] R. MacDonald, "Online Document Verification: How Does It Work?," 15 April 2022. [Online]. Available: https://www.1kosmos.com/authentication/online-document-verification/. [Accessed 31 May 2022].

[6] "Document Verification," [Online]. Available: https://shuftipro.com/knowledgebase/digital-document-verification/. [Accessed 31 May 2022].

[7] "Machine Learning Algorithm Cheatsheet by SAS," 7 April 2020. [Online]. Available: https://coursebapu.com/data-science-resources/machine-learning-algorithm-cheatsheet-by-sas/. [Accessed 5 June 2022].

[8] U. Malmendier and J. Bradford, "Historical Echoes: Cash or Credit? Payments and Finance in Ancient Rome - Liberty Street Economics," 22 February 2013. [Online]. Available: https://libertystreeteconomics.newyorkfed.org/2013/02/historical-echoes-cash-or-credit-payments-and-finance-in-ancient-rome/. [Accessed 31 May 2022].

[9] "Cheque dates through the ages," [Online]. Available: https://www.chequeandcredit.co.uk/information-hub/history-cheque/cheque-dates-through-ages. [Accessed 31 May 2022].

[10] "NEWLY DISCOVERED OLDEST SURVIVING AMERICAN CHECKBOOK," 12 July 2014. [Online]. Available: https://rarebookbuyer.com/webuyoldbooks/newly-discovered-oldest-surviving-american-checkbook/. [Accessed 31 May 2022].

[11] S. Quinn and W. Roberds, "The Evolution of the Check as a Means of Payment: A Historical Survey," *FEDERAL RESERVE BANK OF ATLANTA,* vol. 93, no. 1, pp. 19-20, 2008.

[12] K. R. Schmitt, "Check Definition," 10 August 2021. [Online]. Available: https://www.investopedia.com/terms/c/check.asp#citation-4. [Accessed 31 May 2022].

[13] "Understanding Check Basics," [Online]. Available: https://www.centralbank.net/learning-center/understanding-check-basics/. [Accessed 31 May 2022].

[14] K. Severance, "10 Advantages of Using Personal Checks," 10 May 2022. [Online]. Available: https://www.checkcity.com/personal-finance/the-advantages-of-using-personal-checks. [Accessed 31 May 2022].

[15] D. Witten, R. Tibshirani, T. Hastie, G. James and G. James, An Introduction to Statistical Learning: With Applications in R, vol. 11, Springer New York, 2013.

[16] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of research and development,* vol. 3, no. 3, pp. 210 - 229, 1959.

[17] P. Lison, *An introduction to machine learning,* Language Technology Group, 2012.

[18] D. Soni, "Supervised vs. Unsupervised Learning | by Devin Soni ," 22 March 2018. [Online]. Available: https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d. [Accessed 5 June 2022].

[19] I. Salian, "Difference Between Supervised, Unsupervised, & Reinforcement Learning," 2 August 2018. [Online]. Available: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/. [Accessed 5 June 2022].

[20] P. Lison, *An introduction to machine learning,* Language Technology Group, 2012.

[21] R. S. Sutton, "Introduction: The Challenge of Reinforcement Learning," *Springer, Boston, MA,* vol. 173, no. 1, pp. 1-3, 1992.

[22] "Machine Learning Algorithm Cheatsheet by SAS," 7 April 2020. [Online]. Available: https://coursebapu.com/data-science-resources/machine-learning-algorithm-cheatsheet-by-sas/. [Accessed 5 June 2022].

[23] "What is Supervised Learning?," 19 August 2020. [Online]. Available: https://www.ibm.com/cloud/learn/supervised-learning. [Accessed 5 June 2022].

[24] A. Singh, M. Elhoseny, A. A. Elngar and K. K. Singh, "Machine Learning and the Internet of Medical Things in Healthcare," Elsevier Science, 2021.

[25] V. J. Schmid and A. Meyer-Baese, Pattern Recognition and Signal Analysis in Medical Imaging, Elsevier Science, 2016.

[26] S. Grimnes and O. G. Martinsen, Bioimpedance and Bioelectricity Basics, Elsevier Science, 2014.

[27] D. Steen, "Precision-Recall Curves. Sometimes a curve is worth a thousand… | by Doug Steen," 19 September 2020. [Online]. Available: https://medium.com/@douglaspsteen/precision-recall-curves-d32e5b290248. [Accessed 5 June 2022].

[28] K. D. Foote, "Artificial Neural Networks: An Overview - DATAVERSITY," 11 February 2021. [Online]. Available: https://www.dataversity.net/artificial-neural-networks-overview/. [Accessed 6 June 2022].

[29] T. COUR, G. GIRAUD, A. KODSI, T.-A. LUONG, R. LAURANSON, C. MARCOVICI and K. SADEGHI, *RECONNAISSANCE DE FORMES PAR RESEAU DE NEURONES,* 2002.

[30] D. J. Hemanth, D. Gupta and V. E. Balas, Intelligent Data Analysis for Biomedical Applications: Challenges and Solutions, Elsevier Science, 2019.

[31] S. Razavi and B. A. Tolson, "A New Formulation for Feedforward Neural Networks," *IEEE,* vol. 22, no. 10, pp. 1588 - 1598, 2011.

[32] Y. Y, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *IEEE,* vol. 86, no. 11, pp. 2278 - 2324, 1998.

[33] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, M. Press, Ed., 2016.

[34] F.-F. Li, A. Karpathy and J. Jonhson, "Stanford University CS231n: Convolutional Neural Networks for Visual Recognition," [Online]. Available: http://cs231n.stanford.edu/2016/. [Accessed 6 June 2022].

[35] H. Pokharna, "The best explanation of Convolutional Neural Networks on the Internet!," 28 July 2016. [Online]. Available: https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8. [Accessed 6 June 2022].

[36] K. T. O'Shea and R. Nash, *An Introduction to Convolutional Neural Networks,* 2015.

[37] S. NECIB, "Fusion De Face 3d Couleur, Profondeur Et Profil Pour Srv3d," 2015. [Online]. Available: https://www.theses-algerie.com/2693188195199307/memoire-de-master/universite-mohamed-khider---biskra/fusion-de-face-3d-couleur-profondeur-et-profil-pour-srv3d. [Accessed 6 June 2022].

[38] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way," 15 December 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed 6 June 2022].

[39] "Convolution," [Online]. Available: https://developer.nvidia.com/discover/convolution. [Accessed 6 June 2022].

[40] D. C. Ciresan, U. Meier, L. M. Gambardella and J. r. Schmidhuber, *Convolutional Neural Network Committees for Handwritten Character Classification,* vol. 1, 2011, pp. 1135-1139.

[41] D. Scherer, A. C. Müller and S. Behnke, *Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition,* vol. 1, 2010, pp. 92-101.

[42] J. Ponce and Y. Lecun, *A Theoretical Analysis of Feature Pooling in Visual Recognition,* vol. 1, 2010, pp. 111-118.

[43] N. Srivastava, G. E. Hinton', A. Krizhevsky, I. Sutskever and R. Salakhutdinov, vol. 15, 2014, pp. 1929-1958.

[44] Ioffe, Sergey, Szegedy and Christian, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,* vol. 37, 2015, pp. 448--456.

[45] C. M. Bishop and P. o. N. C. C. M. Bishop, Pattern Recognition and Machine Learning, Springer, Ed., 2007.

[46] Y. Y, L. Bottou, Y. Bengio and P. Haffner, *Gradient-based learning applied to document recognition,* vol. 86, pp. 2278 - 2324.

[47] Krizhevsky, Alex, Sutskever, Ilya and Hinton, *ImageNet Classification with Deep Convolutional Neural Networks,* vol. 25, 2012, p. 1097–1105.

[48] S. O. Haykin, Neural Networks and Learning Machines, P. Education, Ed., 2011.

[49] R. O. Duda, P. E. Hart and D. G. Stork, Pattern classification, 2nd edition ed., Wiley-Interscience, Ed., 2000.

[50] Hinton, Geoffrey, Srivastava, Nitish, Krizhevsky, Alex, Sutskever, Ilya, Salakhutdinov and Ruslan, *Improving neural networks by preventing co-adaptation of feature detectors,* 2012.

[51] A. Krizhevsky, V. Nair and G. Hinton, "The CIFAR-10 & CIFAR-100 dataset," 2009. [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html. [Accessed 06 June 2022].

[52] P. Koukaras, C. Berberidis and C. Tjortjis, *A Semi-supervised Learning Approach for Complex Information Networks,* Springer Singapore, 2021.

[53] "What is a Raspberry Pi?," [Online]. Available: https://opensource.com/resources/raspberry-pi. [Accessed 6 June 2022].

[54] P. Warden, "Building Mobile Applications with TensorFlow [Book]," [Online]. Available: https://www.oreilly.com/library/view/building-mobile-applications/9781491988435/. [Accessed 6 June 2022].

[55] "About Keras," [Online]. Available: https://keras.io/about/. [Accessed 2022 June 6].

[56] P. Khlebovich, "IP Webcam," Pavel Khlebovich, 26 Jul 2010. [Online]. Available: https://ip-webcam.en.uptodown.com/android. [Accessed 14 June 2022].

[57] "The Chars74K image dataset - Character Recognition in Natural Images," 15 October 2012. [Online]. Available: http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/#download. [Accessed 06 June 2022].

[58] K. DHAMA , "Printed Digits Dataset," Kaggle, 2022. [Online]. Available: https://www.kaggle.com/datasets/kshitijdhama/printed-digits-dataset?resource=download. [Accessed 06 June 2022].

[59] S. AKSHMIGPRAKASH, "CEDAR-Dataset," 2021. [Online]. Available: https://www.kaggle.com/datasets/shreelakshmigp/cedardataset. [Accessed 06 June 2022].

[60] E. Alajrami, B. A. Ashqar, B. S. Abu-Nasser, A. J. Khalil, M. M. Musleh, A. M. Barhoom and S. S. Abu-Naser, "Handwritten Signature Verification using Deep Learning," *International Journal of Academic Multidisciplinary Research (IJAMR)*, vol. 3, no. 12, pp. 39-44, December 2019.

[61] Y. LeCun, C. Cortes and C. J. Burges, "THE MNIST DATABASE of handwritten digits," [Online]. Available: http://yann.lecun.com/exdb/mnist/#:~:text=The%20MNIST%20database%20was%20constructed,1%20as%20their%20test%20set.. [Accessed 06 June 2022].

[62] K. "MNIST: Simple CNN keras (Accuracy : 0.99)=>Top 1%," 2022. [Online]. Available: https://www.kaggle.com/code/elcaiseri/mnist-simple-cnn-keras-accuracy-0-99-top-1/notebook. [Accessed 06 June 2022].

[63] M. Abderrahmane and H. Amira, *Implémentation d'un réseau de neurone covolutionnel CNN pour la reconnaissance automatique des mots manuscrits en arabe*, UNIVERSITE YAHIA FARES DE MEDEA, 2019.

[64] S. Farrell, "3 Reasons Financial Institutions Need to Automate their Teller Lines," 9 May 2014. [Online]. Available: https://blog.qualitydatasystems.com/3-reasons-financial-institutions-need-to-automate-their-teller-lines. [Accessed 31 May 2022].