PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

YAHIA FARES UNIVERSITY OF MEDEA



Faculty of **T**echnology

Master's Thesis

Department: Electrical Engineering

Specialty: Automatic and systems

Design and Implementation of an Intelligent Authentication System based on Face Recognition

---

Presented by

Mr. SAFAR ZITOUN Zine eddine

Mr. BRAHIM BELHAOUARI Yasser

Proposed and Directed by

Dr. KARA Reda

Dr. BENYEZZA Hamza

Promotion September 2021

Dedication



بسم الله الرحمان الرحيم و الحمد لله

We dedicate this project thesis to our family and many friends. A special feeling of gratitude to our loving parents, who tirelessly encouraged us and counseled us, and to all our siblings who have never left our sides and are very special.

We also dedicate this project thesis to our friends who have supported us throughout the process. We will always appreciate their support and all the things they have done.

We dedicate this work and give special thanks to our best friends for being there for us throughout the master's program. All of you have been our best cheerleaders.

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



Zine Eddine & Yasser

Acknowledgement

بسم الله الرحمان الرحيم و الحمد لله

We would like to thank our dear teacher and supervisor Dr. Kara as well as our co-supervisor Mr. Benyezza for supporting us throughout this tough year and kept motivating us to give the best we could despite this covid-19 pandemic.

Thank you to the jury members for giving us the honor of judging our work. We also deeply thank all the Electrical engineering department teachers

Finally, we thank our family, friends, colleagues and all the class of 2021

ملخص:

تواجه مؤسسات اليوم تحديات أمنية كبيرة. لذلك فهُم بحاجة إلى العديد من رجال الأمن. لكن هؤلاء الأفراد, كبشر, يرتكبون أخطاء يمكن أن تؤثر على مستوى أمن المنطقة.

أحد الحلول المقترحة لهذه المشكلة هو نظام أمان يعمل بالتعرف على الوجه, و الذي يُمَكِنْ بإكتشاف المتسللين إلى المناطق المحظورة أو عالية الأمان, و يساعد في تقليل الخطأ البشري. يتكون النظام من جزأين: جزء المعدات وجزء برمجي. يتكون جزء المعدات من كاميرا متصلة بقفل كهربائي, بينما يتكون جزء البرمجي من خوارزميات لإكتشاف الوجوه و التعرف عليها, المرتكزة على الشبكات العصبية الإلتفافية.

عندما يدخل شخص ما إلى المنطقة المعنية, يجب عليه أولاً المصادقة باستخدام شاشة اللمس, حيث يتم أخذ صورة لوجهه و يتم إرسالها إلى برنامج التعرف على الوجوه لتحليلها و مقارنتها بقاعدة بيانات المستخدمين المسجلين؛ إذا تم العثور على المستخدم, يتم فتح الباب (إذا لم يتم العثور عليه, فيظل الباب مغلقاً), ويتم تسجيل محاولة الدخول في قاعدة بيانات و الذي يمكن عرضها على لوحة التحكم (عبارة على موقع ويب)

كلمات مفتاحية: الكشف عن الوجه, التعرف على الوجه, معالجة الصور الرقمية, CNN, القياسات الحيوية.

Résumé :

Les institutions d'aujourd'hui sont confrontées à des problèmes de sécurité majeurs. Elles ont donc besoin de plusieurs personnels spécialement formés pour atteindre la sécurité souhaitée. Ce personnel, en tant qu'êtres humains, commet des erreurs qui peuvent affecter le niveau de sécurité.

Une solution proposée à ce problème est un système de sécurité par reconnaissance faciale, qui peut détecter les intrus dans les zones à accès restreint ou de haute sécurité, et aider à minimiser les erreurs humaines. Ce système est composé de deux parties : une partie matérielle et une partie logicielle. La partie matérielle consiste en une caméra reliée à une serrure électrique, tandis que la partie logicielle comprend un algorithme de détection et de reconnaissance facial basé sur CNN (Réseau neuronal convolutif).

Lorsqu'une personne entre dans la zone en question, elle doit d'abord s'authentifier à l'aide d'un écran tactile, où un cliché de son visage est pris et envoyé au logiciel de reconnaissance pour être analysé et comparé à une base de données d'utilisateurs inscrits ; si l'utilisateur est trouvé, la porte est déverrouillée (sinon, elle reste verrouillée), et une tentative d'accès est enregistrée dans une base de données pour être affichée sur le panneau de contrôle (site web).

Mots clés : CNN, Reconnaissance facial, Détection facial, Biométrie, Traitement d'image.

Abstract :

Today's institutions face major security challenges; as a result, they require a large number of specially trained personnel to achieve the desired level of security. As humans, these employees make mistakes that compromise the degree of security.

As a solution to the aforementioned issue, a face recognition security system that can detect intruders to high-security areas and aid in the reduction of human error has been proposed. This system is comprised of two parts: software and hardware. The software consists of face detection and recognition algorithms based on CNN (Convolutional neural networks), while the hardware consists of a camera linked to an electric door lock.

When a person wants to enter the zone in question, he must first authenticate using a touch screen, where a snapshot of his face is taken and sent to the recognition software to be analyzed and compared with a database of enrolled users; if the user is found, the door is unlocked (if not, it remains locked), and an access attempt is recorded in a database to be displayed on the control panel (website).

Keywords : CNN, Face Detection, Face Recognition, Biometrics, Digital Image Processing.

TABLE OF CONTENTS

LIST OF FIGURES ..... **I**

LIST OF TABLES ..... **III**

LIST OF ACRONYMS ..... **IV**

GENERAL INTRODUCTION ..... **V**

CHAPTER I : BIOMETRICS FUNDAMENTALS **1**

1. Introduction **1**

2. History on biometrics **1**

2.1. Definition of biometrics 2
2.2. Advantages of biometrics 3
2.3. Biometric types overview 3
2.3.1 Physiological 4
2.3.1.1 Fingerprints 4
2.3.1.2 Hand shape 4
2.3.1.3 Eye's iris 5
2.3.1.4 Facial features 6
2.3.2 Behavioral 7
2.3.2.1 Voice 7
2.3.2.2 Hand signature 8

3. Face recognition **9**

3.1. History 9
3.2. Fields of application 10
3.3. Face detection and recognition techniques 11
3.4. Problematic 12
3.5. Advantages and disadvantages of face recognition 13

4. Conclusion **13**

CHAPTER II : ARTIFICIAL INTELLIGENCE **1**

1. Introduction **14**

2. Basic Concepts of Machine Learning **14**

2.1. Machine Learning Systems 14
2.2. Supervised Learning 15
2.3. Learning Algorithms 16
2.3.1. Parametric models 16
2.3.2. Cost functions 17
2.3.3. Generalization 17
2.3.4. Regularization 17

2.4. Evaluation of a Binary Classifier 18
2.4.1. Confusion Matrix 18
2.4.2. Receiver Operating Characteristic Curve 19
2.4.3. Precision Recall Curve 19

3. Artificial Neural Networks 20

3.1. Basic Concepts 21
3.2. Feedforward Neural Networks 23
3.2.1. Single-Layer Perceptron 23
3.2.2. Multilayer Perceptron 24
3.3. ANN Learning 25
3.3.1. Gradient Descent Methods 25
3.3.2. Back-propagation Algorithm 27
3.3.3. Regularization 27

4. Convolutional Neural Networks 28

4.1. Convolution Operation 28
4.1.1. Convolution in Mathematics and Image Processing 28
4.1.2. Convolution in Image Processing 29
4.1.3. Convolution in Neuroscience 30
4.2. Convolutional Network Architecture 30
4.2.1. Convolutional Layer 30
4.2.2. Pooling 31
4.2.3. Batch Normalization 33
4.3. Main Concepts Behind CNNs 33
4.3.1. Local Receptive fields 33
4.3.2. Parameter Sharing 34
4.4. Popular CNN Architectures 35

5. Conclusion 36

CHAPTER III : MATERIALS USED 37

1. Introduction 37

2. Hardware used 37

2.1. Raspberry pi 4 37
2.2. Relay 38
2.3. Solenoid lock 39
2.4. Raspberry Pi Camera Board v1.3 39

3. Software used 40

3.1. Python 40
3.2. TensorFlow 41
3.3. Keras 41
3.4. OpenCV 42
3.5. Sqlite 42
3.6. Php 43
3.7. Tkinter 43
3.8. Jupyter and Google Colaboratory 43

CHAPTER IV : REALISATION AND TESTS **45**

1. Introduction **45**

2. Specification book **45**

2.1. Purpose 45
2.2. Product features 45
2.3. Interface requirements 46

3. Global System flow diagram **46**

4. Face recognition model **47**

4.1. First attempt: CNN Face Classifier 47
4.1.1 Dataset 48
4.1.2 Training 49
4.1.3 Results 50
4.1.4 Additional tests 53
4.2. Second attempt: CNN Face Descriptor 56
4.2.1 Hypothesis 56
4.2.2 Model used 56
4.2.3 Similarity distance threshold fine-tuning 61
4.2.3.1 Fine-tuning using Euclidean distance 62
4.2.3.2 Fine-tuning using Cosine distance 65
4.3 Discussion 67

5. Implementation **68**

5.1 Graphical User Interface 69
5.2 Face detection and recognition 71
5.3 Website 72
5.3.1 Login page 73
5.3.2 Access reports page 73
5.3.3 Add users page 74
5.4 Hardware implementation 74
5.4.1 Electrical wiring 75
5.4.2 Final product assembly 76

6. Results **77**

7. Conclusion **79**

GENERALE CONCLUSION **80**

BIBLIOGRAPHY **81**

LIST OF FIGURES

Figure I.1. *Fingerprint Recognition Process* 4
Figure I.2. *Hand Geometry Reader* 5
Figure I.3. *Different shapes of the Iris* 6
Figure I.4. *Facial Landmarks for face recognition[5]* 7
Figure I.5. *Voice Passphrase Spectrogram example* 8
Figure I.6. *Hand Signature Terminal* 8
Figure I.7. *Block Diagram of Face Recognition System* 12
Figure II.1. *An example of the ROC curve* 19
Figure II.2. *An example of the PR curve* 20
Figure II.3. *A biological neuron (a) compared to an artificial neuron (b).* 21
Figure II.4. *Popular activation functions[42]* 22
Figure II.5. *A simple artificial neural network example.* 23
Figure II.6. *Edge detection with convolution [53].* 29
Figure II.7. *Illustration of the 2x2 max pooling function [55]* 32
Figure II.8. *CNN illustration [58].* 33
Figure II.9. *Local receptive fields* 34
Figure II.10. *Deep indirect interactions* 34
Figure II.11. *Parameter sharing* 35
Figure III.1. *Raspberry Pi 4 board* 37
Figure III.2. *Single-Channel Relay Module Pinout* 38
Figure III.3. *Solenoid lock* 39
Figure III.4. *Camera RPi v1.3 Module* 40
Figure III.5. *Python Programming Language Logo* 41
Figure III.6. *TensorFlow Logo* 41
Figure III.7. *Keras Logo* 42
Figure III.8. *PHP Logo* 43
Figure IV.1. *System flow diagram* 47
Figure IV.2. *Dataset sample images after pre-processing* 48
Figure IV.3. *CNN Configuration with the best accuracy rate* 50
Figure IV.4. *Training loss and accuracy results* 51
Figure IV.5. *Confusion matrix for configuration 1* 52
Figure IV.6. *Test results on known subjects* 54
Figure IV.7. *Test results on unknown subjects* 55
Figure IV.8. *Visualization of VGGface architecture.[76]* 58
Figure IV.9. *LFW ROC curve comparison[76]* 60
Figure IV.10. *Histogram of threshold distribution for positive/negative pairs using Euclidean distance and the 2622-d feature vector* 62
Figure IV.11. *Threshold accuracy in detecting negative and positive pairs, for Euclidean distance and 2622-d feature vector* 63
Figure IV.12. *Histogram of threshold distribution for positive/negative pairs using Euclidean distance and the 4096-d feature vector* 64
Figure IV.13. *Threshold accuracy in detecting negative and positive pairs, for Euclidean distance and 4096-d feature vector* 64
Figure IV.14. *Histogram of threshold distribution for positive/negative pairs using Cosine distance and the 2622-d feature vector* 65
Figure IV.15. *Threshold accuracy in detecting negative and positive pairs, for Cosine distance and 2622-d feature vector* 66
Figure IV.16. *Histogram of threshold distribution for positive/negative pairs using Cosine distance and the 4096-d feature vector* 66
Figure IV.17. *Threshold accuracy in detecting negative and positive pairs, for Cosine distance and 4096-d feature vector* 67
Figure IV.18. *Graphical user interface of our authentication system* 69
Figure IV.19. *Flowchart of the main functionality of the GUI application* 70
Figure IV.20. *Detailed flowchart of the inner workings of the recognition function* 72
Figure IV.21. *Login page of the website* 73
Figure IV.22. *Authentication report page of the website* 73

Figure IV.23. *User management page of the website* 74
Figure IV.24. *Electrical wiring diagram of our system* 75
Figure IV.25. *Assembled authentication system* 76
Figure IV.26. *Authentication test before adding the user* 77
Figure IV.27. *User successfully added to enrolled users database* 77
Figure IV.28. *Access Granted to user that is present in enrolled users database* 78
Figure IV.29. *Access Granted to user with artificial beard and glasses that is present in enrolled users database* 78
Figure IV.30. *Authentication report populated by access attempts* 79

LIST OF TABLES

Table II.1. *Confusion matrix for binary classification* 18
Table III.1. *Offered specs by Google Colaboratory* 44
Table IV.1. *F1-Score for configuration 1* 53
Table IV.2. *The accuracy of different methods evaluated on the LFW dataset [75]* 57
Table IV.3. *Racial bias in existing commercial recognition APIs and face recognition algorithms [75]* 58
Table IV.4. *VGGface Dataset curation process[76]* 59
Table IV.5. *LFW unrestricted setting results[76]* 60
Table IV.6. *Comparison between the results of different distance algorithms* 68

LIST OF ACRONYMS

ACC : Accuracy
AdaGrad : Adaptive Gradient
ADAM : ADAptive Moment estimation
ANN : Artificial Neural Networks
BP : Back-Propagation
CNN : Convolutional Neural Network
CPU : Central Processing Unit
CSI : Camera Serial Interface
DB : Database
DL : Deep Learning
EER : Equal Error Rate
FC : Fully-Connected
FN : False Negative
FP : False Positive
FPR : False Positive Rate
GD : Gradient Descent
GPU : Graphics Processing Unit
GUI : Graphical User Interface
KNN : K-Nearest Neighbor
LBP : Local Binary Pattern
LDA : Linear Discriminant Analysis
LFW : Labeled Faces in the Wild
ML : Machine Learning
MLP : Multi-Layer Perceptron
PPV : Positive Predictive Value
PR : Precision Recall
ReLU : Rectified Linear Unit
ROC : Receiver Operating Characteristic
SGD : Stochastic Gradient Descent
SVM : Support Vector Machine
TN : True Negative
TP : True Positive
TPR : True Positive Rate
TPU : Tensor Processing Unit
YTF : YouTube Faces

GENERAL INTRODUCTION

Nowadays, there is more and more insecurity in various sectors caused by an increasingly industrialized society, making forgery easier and easier. For this reason, several computerized means are implemented to counter this trend. One of these modern means is face recognition, which acts like a secondary defense line for its ability to keep detailed logs of everyone accessing the establishment. Our goal in this thesis is to create a fully automated system to manage access control to restricted areas, from identifying faces to log keeping.

A face recognition system is a software/application that automatically recognizes a person's face. It is a field of computer vision that consists of automatically recognizing a person from an image of his face from a set of faces stored in a database. It is a particularly studied subject in computer vision, with many publications and patents.

Face recognition systems have many applications in video surveillance, biometrics, robotics, image and video indexing, image search by content, etc. They are widely used because they are considered to be low cost. They are used because they are considered to be less invasive than other biometric systems (fingerprints, iris recognition...). The operation of these systems is based on one or more cameras to recognize the user.

In this Master thesis, we will propose an architecture of an authentication system that will keep intruders out of high security zones by enforcing a facial verification process on every employee when they seek to enter a restricted sector. It will be managed from a control panel in a form of a website, where the administrator of the system will have the ability to add new employees as well as check detailed descriptions of every access attempt that happened before.

This thesis is made of four chapters, the first one is a presentation of the state of the art on biometry and face recognition technology, on the next chapter we will dive into the world of artificial intelligence; from ML concepts to CNN architectures, the third chapter will be a general description on 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 : BIOMETRICS FUNDAMENTALS

1. Introduction

During our daily life, each of us identifies different faces throughout the day. So when we meet a person, our brain searches our memory and checks if this person is listed or not, this is an easy task for humans.

Is it the same for a machine? In this chapter, we are going to present the main lines of our work, we will start with some history on biometrics then we are going to explain the general notions about face recognition, the functioning of a face recognition system, we are also going to talk about some techniques used, some examples and the fields of application.

2. History on biometrics

The oldest anthropometric recognition process, well known to the general public, is the analysis of fingerprints, which explains its importance today among the many identification processes based on physiological characteristics. This type of analysis is now very advanced, but research on it is not new.

Although the handprints left by our ancestors on the walls of caves, have not given us all their secrets, the thumbprint was already used as a signature in trade-in Babylon (3000 BC) and ancient China (7th century). Closer to home, Marcello Malpighi, an Italian physician, and biologist, described the components of numerous organs such as the liver and kidneys, a research that was highly criticized at the time. He described the basal layer of the epidermis, to which he gave his name, and studied the patterns and curves of fingerprints as early as the 17th century. The Czech physiologist Jan Evangelista Purkinje, a professor of embryology and pharmacology, worked on different parts of the human body, including fingerprints, and published a thesis around 1820 in which he studied different types of prints.

However, it was not until 1870 that fingerprints were used for identification purposes by Dr. Henry Faulds, then a surgeon in a Tokyo hospital. He published an article in the journal "Nature", in which he developed the idea of using fingerprints to identify people. Faulds was later the first to establish a classification using printing ink and informed Francis Galton. This British scientist, renowned for his work in anthropology and heredity, was passionate about the theory of evolution, although he was interested in many other scientific fields such as meteorology and statistics. He did important work on the measurements of human beings, and collected statistics on the height, weight, dimensions and other characteristics of many people. Galton paid particular attention to fingerprints and developed a method of

identification in his book "Fingerprints", where he established the uniqueness and permanence of skin patterns. His global approach to populations allowed him to note the important variation of fingerprints in the human species but also within different ethnic groups. He also demonstrated that there was only one chance in 64 billion that two individuals would have the same fingerprints and affirmed the permanence of fingerprints in the same individual throughout his life. Thanks to him, in Bengal, as early as 1858, the thumbprint was used as proof of identity.

The first prototypes of automatic fingerprint processing terminals arrived in the mid-1970s, and the first commercial products were put on the market in the early 1980s. These terminals were initially used for access control and/or time management for clients of "governmental organization" type (prisons, for example).[1]

2.1. Definition of biometrics

Biometry is the application of statistical analysis to biological data or take body measurements and calculations related to human characteristics, there exist three ways to prove one's identity:[2]

1. What you have (card, badge, document)
2. What you know (a name, a password)
3. What we are (fingerprints, hand, face...) - This is biometrics

There are three types of biometric technologies; the first is based on biological analysis, which focuses on reading biological characteristics unique to each human such as odor, blood, saliva, urine, and DNA. This technology is rarely used for public reasons since it's very intrusive and not very practical. The second type is based on behavioral analysis, this method focuses on identifying individuals based on their behavior, such as their signature dynamics (speed of movement of the pen, acceleration speed, pressure, inclination), or the way of using a computer keyboard (speed of typing or pressure exerted on the keys), their voice, or their way of walking (gait). The final type of technology is based on morphological analysis, which scans for elements in the human body that tend to be stable in their life but also unique to each individual such as fingerprints, hand shape, facial features, vein pattern, eye's iris and so on.

In general, biometric systems are used for purposes of authentication, or identification:

1. Authentication (1:N): in this mode, the question is asked: "Am I Mr. X? ". Technically, the device checks against a code (identifier) entered on a

keyboard, or read by the passage of a badge (smart card, magnetic card, proximity, etc.) that the biometric sample provided corresponds to the template designated by the identifier.

2. Identification (1:1): for this mode, the simple question is asked: "Who am I? From the biometric sample provided, the device searches for the corresponding template in its database.

The characteristics of biometric identifiers must be universal and exist in all individuals; it should be unique to allow differentiation between one individual from another, permanent and stable throughout one's lifetime, and measurable to allow for comparison between individuals.

2.2. Advantages of biometrics

Biometrics is an emerging field where technology is improving our ability to identify a person. Protecting consumers from fraud or theft is one of the goals of biometrics. The advantage of biometric identification is that each individual has their own physical characteristics that cannot be changed, lost or stolen. The biometric identification method can also be used to complement or replace passwords. [3]

There are several reasons for using biometrics:

- High security - By associating it with other technologies such as encryption
- Comfort - By replacing just the password, for example for login in to an operating system, biometrics allow to respect the basic rules of security (plus no password to remember). And when these rules are respected, biometrics saves network administrators from having to answer the many calls for lost passwords (which are sometimes given over the phone, thus without security).
- Security / Psychology - In some cases, especially for e-commerce, the user wouldn’t know how trustworthy a store is. It is important for the managers of this market to convince the consumer to make transactions. A known means of authentication such as fingerprints could change consumer behavior.

2.3. Biometric types overview

There are two main types of biometric identifiers. A first type is physiological characteristic that relate to the shape or composition of the body (fingerprint, hand, face, iris, retina or ear features). A second type is behavioral characteristic that relate to the behavior of a person

(typing rhythm, gait, gestures and voice). This section summarizes the most used modalities in recognition.

2.3.1 Physiological

2.3.1.1 Fingerprints

Fingerprint recognition is the oldest biometric technique and one of the most mature. It is based on the fact that each person has unique fingerprints. After capturing the image of the fingerprint, the fingerprint is enhanced. Then we identify and extract the fingerprints (Figure I.1.), which will be compared with all the fingerprints saved by other users. This is one of the most studied and used biometric technologies, especially in access control [4].

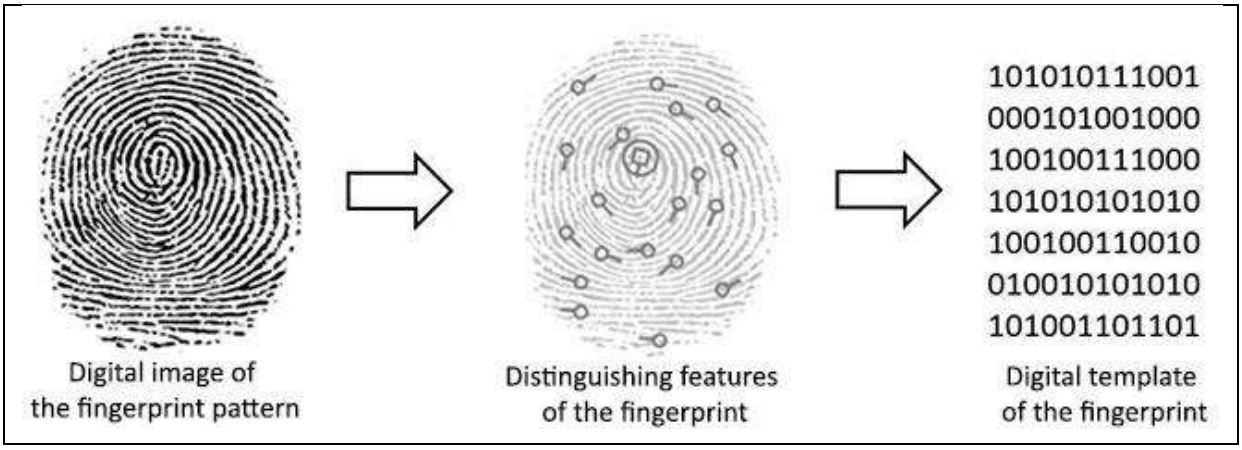


Figure I.1. *Fingerprint Recognition Process*

2.3.1.2 Hand shape

Palm biometrics is a popular technology that is widely used for physical access control or time and attendance. It is very simple and inexpensive. The accuracy of this biometric system is quite reasonable. The elements taken into account are only based on the geometry of the hand and not on the palm print.

The system takes a picture of the hand and examines 90 features, including the three-dimensional shape of the hand, the length and width of the fingers and the shape of the joints.

To use the hand geometry, the user places his or her hand on a stage that has guides for positioning the fingers.

Hand geometry readers (Figure 1.2) offer a very reasonable level of accuracy, but can have high false acceptance rates for twins or other family members.



Figure I.2. *Hand Geometry Reader*

2.3.1.3 Eye's iris

The use of the iris to identify a person was originally proposed in 1936 by ophthalmologist Frank Burch. In 1987 ophthalmologists Aran Safir and Leonard Flom patented the idea and in 1989 they asked John Daugman (then teaching at Harvard University), to try to create algorithms for iris identification. These algorithms (based on Gabo waves), which Daugman patented in 1994, are the basis of all iris identification systems.

Iris identification uses more parameters than other identification methods and the resulting reliability is sufficient to do not authentication but identification.

The probability of finding 2 sufficiently identical irises is 1 in 10 to the power of 72 (according to Daugman's estimates).

Two identical twins have enough distinctive elements on their irises to distinguish them from each other.

This formation is chaotic (Figure I.3.), so it generates patterns with a high variability (244 degrees of freedom).

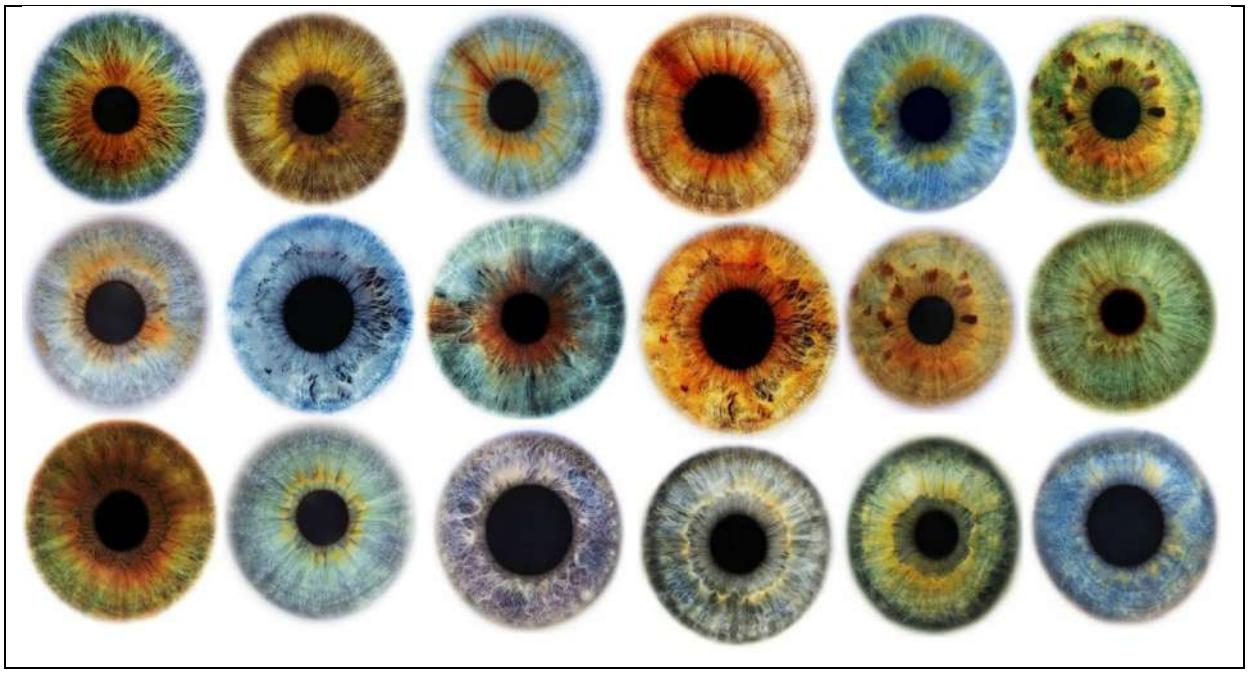


Figure I.3. *Different shapes of the Iris*

2.3.1.4 Facial features

Nothing is more natural than using the face to identify a person. Facial images are probably the most common biometric feature used by humans for personal identification.

The use of a camera captures the shape of an individual's face and identifies certain features. Depending on the system used, the individual must be positioned in front of the camera or may be moving at a certain distance. The biometric data obtained is then compared to the reference file.

In the early 1970s, facial recognition was mainly based on measurable facial attributes (Figure I.4.) such as eye width, eyebrows, lips, chin position, shape, etc. Since the 1990s, the various technologies used have exploited all the discoveries made in the field of image processing, and more recently neural networks.

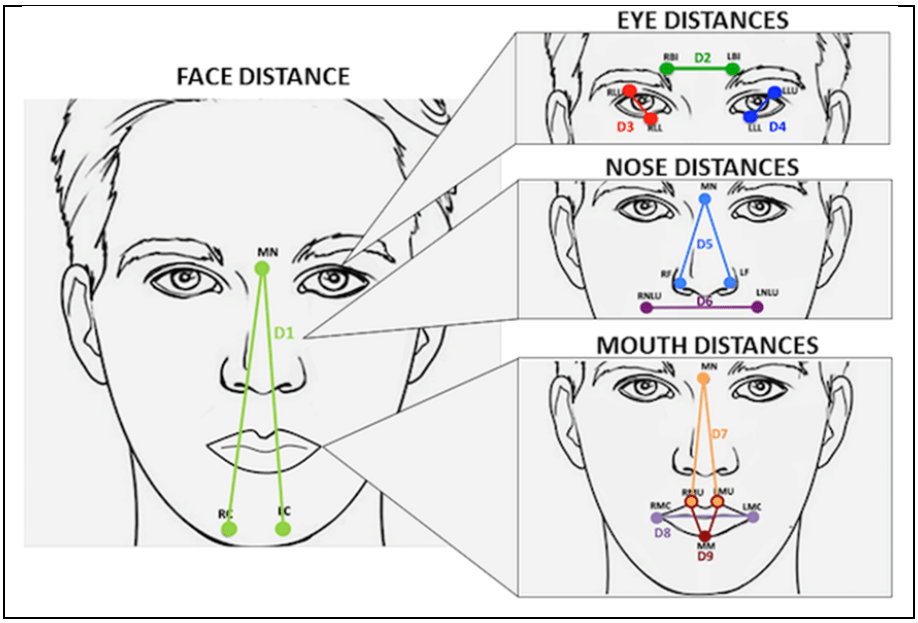


Figure I.4. *Facial Landmarks for face recognition[5]*

2.3.2 Behavioral

2.3.2.1 Voice

Voice identification is considered by users to be one of the most normal forms of biometric technology, as it is non-intrusive and does not require physical contact with the system's reader.

Voice analysis technology is successfully applied where other technologies are difficult to employ. It is used in areas such as call centers, banking, access control, home PCs, network access and forensic applications.

Most voice identification systems use a text display, specific words must be read and then spoken to verify that the person being authenticated is present and not a recording.

Voice identification systems focus on only those voice features that are unique to an individual's speech pattern (Figure I.5.). These speech patterns are made up of a combination of behavioral and physiological factors.

Imitators usually try to reproduce the speech features that are most obvious to the human auditory system and do not recreate the less accessible features that an automated voice identification system analyzes. Therefore, identity theft is not easy.

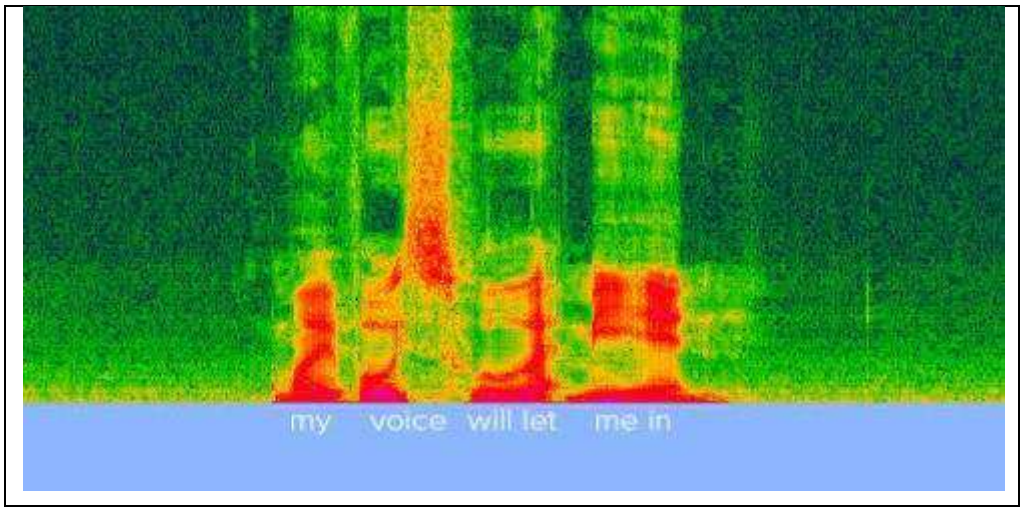


Figure I.5. *Voice Passphrase Spectrogram example*

2.3.2.2 Hand signature

Signing a document to identify oneself is a natural gesture. Whether it's to confirm a transaction on a credit card account or simply to mark a birthday card, we're all used to putting our signature on something.

Each person has a unique writing style. We can therefore define, from a person's signature, a model that can be used to make an identification. Moreover, the signature is used in many countries as a legal or administrative element. It allows to prove the good faith of a person.

The (dynamic) signature verification is based on the differentiation between the parts of a signature that are usual and those that change with almost every signature.

Signature authentication systems usually include a pen reader and a tablet (Figure I.6.). Verification is accomplished by analyzing a number of variables, including the speed with which the signature is performed, accelerations, pressure exerted…



Figure I.6. *Hand Signature Terminal*

3. Face recognition

Facial recognition is a technique that allows to:

- Authenticate a person: i.e., verify that a person is who he or she claims to be (in the context of an access control)
- To identify a person: that is, to find a person within a group of individuals, in a place, an image or a database.

In practice, recognition can be performed from still images (photos) or animated images (video recordings) and this takes place in two phases:

- Phase 1**:** From the image, facial features are extracted. The data extracted from this image are to be represented as biometric data.
- Phase 2**:** The recognition phase is then carried out by comparing this biometric data (the candidate image) with the biometric data of known users from a database.

In the case of authentication, the system checks whether the face features presented matches one of the features contained in the database. The results of the comparison correspond to the one or those with the highest similarity score among those exceeding a certain predetermined threshold.

Facial recognition should not be confused with face detection, which characterizes the presence or absence of a face in an image independently of the person to whom it belongs [6].

3.1. History

Facial recognition is a relatively new biometric technique. While fingerprinting is the oldest biometric technique in use since 1903 to search for criminals, face recognition was developed by "Benton and Van Allen" in 1968 to assess the ability to identify unfamiliar faces. It is not a test of recognition of familiar or unfamiliar faces, but a test consisting of matching photographs of unfamiliar faces presented under different lighting and from different angles and requiring good Visio-spatial integration ability [7].

The use of facial recognition techniques has been developing on a large scale since the mid-1990s, with the effective use of new technologies, particularly computers and their image processing capabilities. The use of these techniques has existed since computers were able to understand what they "see" when connected to one or more cameras, i.e., the first attempts date back to the early 1970s (Benton and Van Allen in 1968), and are based on heuristic

methods, based on measurable facial attributes such as the distance between the eyes, eyebrows, lips, the position of the chin, the shape, etc [8].

These methods were not very robust, because they require a specific set of conditions (frontal face, good lighting conditions, etc.) for the recognition to be correct. One of the first attempts at face recognition was made by Takeo Kanade in 1973 during his PhD thesis at Kyoto University [9] [10].

3.2. Fields of application

Security remains the main field of application. In this field, face recognition is necessary for identification and authentication. A good example of this is In 2017, Gemalto (now Thales) signed the equipment for the new automated control of PARAFE (Passage Automatisé Rapide aux Frontières Extérieures) system within the Paris airports. The solution evolved from fingerprint recognition to facial recognition in 2018 and reduced the average passage time from 30-45 seconds to 10-15 seconds [11].

Police control can make use of it, although it is still highly regulated in Europe. In 2016, the "man in a hat" of the Brussels attacks was identified thanks to an FBI facial recognition software [12].

Camera-equipped drones offer an interesting combination for facial recognition at large events, for example. The drones, in a fixed position up to 100 meters high, can carry heavy optics (10kg) and identify a person at 800 meters. Moreover, as they are connected by cable, the electrical autonomy is unlimited and the communications (also by cable) are secure [12].

Google and Facebook have implemented algorithms to identify people in their photo databases. In 2014 Facebook announced the launch of its program called DeepFace, capable of determining whether two photographed faces belong to the same person, with an accuracy of 97.25%. When subjected to the same test, humans respond correctly 97.53% of the time, just 0.28% better than Facebook's program. In June 2015, Google followed suit with FaceNet, a new facial recognition system with unprecedented scores: 99.63% accuracy on the Labeled Faces in The Wild benchmark test; 95% on the YouTube Faces DB. By using an artificial neural network and a new algorithm, the Mountain View firm has managed to bring a face and its owner closer to near perfection. A technology integrated in Google Photos to sort the pictures and tag them automatically according to the people recognized and, as a proof of its relevance, quickly became available online in an unofficial open-source version, OpenFace [12].

Health has also seen considerable advances in this field. Thanks to Deep Learning and facial and emotional recognition, it is already possible To follow more precisely the consumption of medication by a patient; To detect a genetic disease such as DiGeorge syndrome with a success rate of 96.6% [12].

There are many other areas of application of facial recognition such as: Commerce, unlocking systems, snapchat filters...etc.

3.3. Face detection and recognition techniques

There are a multitude of techniques dedicated to face detection. But for the most part it is of interest that these techniques are based on elements of the face that are least susceptible to change: The large upper features of the eye sockets, the areas around the cheekbones, the sides of the mouth and other similar features so as to ignore changes such as haircut [8]. The most commonly used basic methods are:

- Automatic face processing, It is a rudimentary technology, characterizing faces by distances and proportions between particular points such as the eyes, the nose, the corners of the mouth. As proven as the other technologies, automatic face processing is most effective in low-light settings [13].
- Eigenfaces were the first type of characterization successfully used in facial processing such as face detection and recognition [14]. It uses a representation of the characteristic features of a face from gray scale level images. Variants of Eigenface are frequently used as the basis for other recognition methods [15].
- LDA (Linear discriminant analysis) fisherface is an appearance-based technique used for the dimensionality reduction and recorded a great performance in face recognition. This method works on the same principle as the eigenface method (PCA).it performs dimensionality reduction while preserving as much of the class discriminatory information as possible [16].
- HAAR Cascade filter is an Object Detection Algorithm used to identify faces in an image or a real time video. The algorithm uses edge or line detection features proposed by Viola and Jones in their research paper “Rapid Object Detection using a Boosted Cascade of Simple Features” published in 2001. The algorithm is given a lot of positive images consisting of faces, and a lot of negative images not consisting of any face to train on them [17].

- Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. Due to its discriminative power and computational simplicity, LBP texture operator has become a popular approach in various applications, one of which is facial recognition [18].
- Neural networks can theoretically lead to an increased ability to identify faces under difficult lighting/position conditions. Neural networks employ an algorithm to determine the similarity between facial features of live subjects and templates previously learned [19].

3.4. Problematic

The problem of face recognition can be formulated as follows: given one or more images of a face, the task is to find or verify the identity of a person by comparing his or her face to the set of face images stored in a database. In general, a facial biometrics system consists of two modules: a face detection module (face detection or localization), and a recognition module that proceeds in three steps: normalization or preprocessing, facial feature extraction, and classification (Figure 1.7).

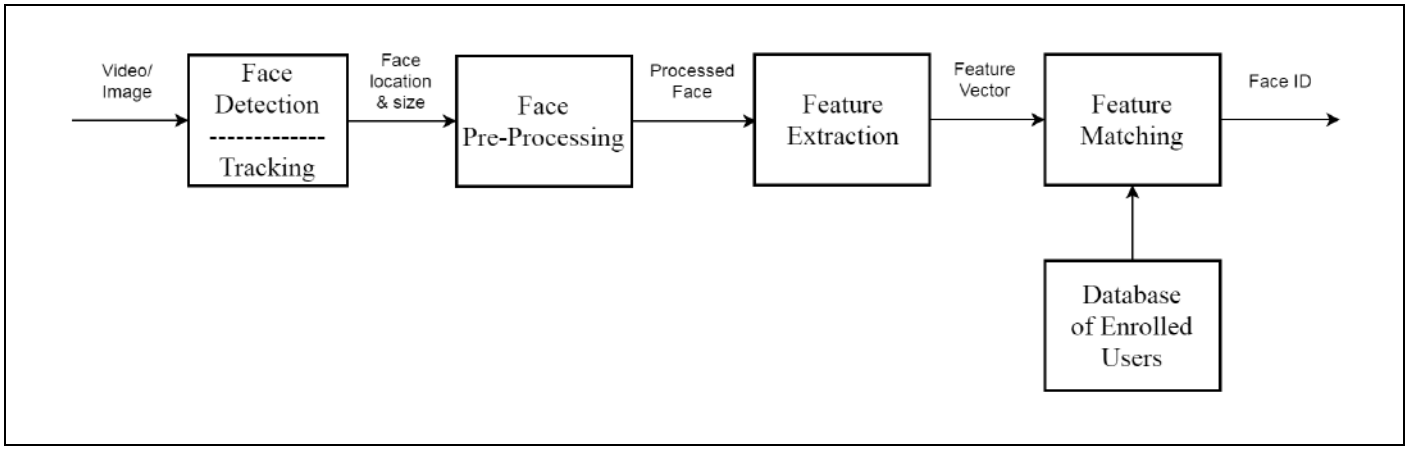


Figure I.7. *Block Diagram of Face Recognition System*

- Capture: image or video this is the first step in the process. It is necessary to succeed in capturing the relevant information without noise [20].
- Face detection: Face detection is an essential step before the recognition phase. Indeed, the face recognition process can never become fully automatic if it is not preceded by an efficient detection step [21]. The quality can be filtered or improved by pre-processing the extracted face. Geometric and photometric normalizations can be performed. These preprocessings are necessary to eliminate or limit pose variations, to

standardize the illumination, to minimize the influence of illumination, as well as to adjust the face to have a particular dimension and to be horizontal [20].

- Feature Extraction: The goal is to extract features from the face that can make it both different from other people's faces and robust to variations in the person itself. This is the information needed to make a person's face not look like another person and at the same time look like itself under other acquisition conditions [20].
- Comparison: Depending on the features extracted previously, the comparison algorithms differ. Several approaches are found in the literature: distance calculation, similarity calculation. Other methods are based on the classification of features by a single classifier (SVM, Baysian classifier, etc.) or by several classifiers (Adaboost) [20].

3.5. Advantages and disadvantages of face recognition

Several factors make face recognition attractive for large-scale use, it is acceptable by people since it is an apparent part of the body, easily verifiable by any operator to have a decision, non-intrusive because of the easy verification. In addition, image sensors are less expensive in the market [22]. Despite its advantages, it has disadvantages that affect the quality of recognition. The following aspects can be mentioned [22]:

- Change in illumination.
- Facial expressions.
- Presence or absence of structural components, such as (beard, mustache and glasses).
- Identical twins.

4. Conclusion

Biometrics aims to imitate the mental pattern recognition process in the manner that it identifies persons. It is a more secure and reliable alternative to the classical authentication schemes based on secrets and tokens.

In this chapter we have presented an overview on biometry and face recognition. We have also highlighted the critical points to be taken care of and identify the parts to be addressed during the next steps of our project. In the next chapter, we will introduce the basic concepts concerning artificial intelligence and the building blocks of convolutional neural networks.

CHAPTER II : ARTIFICIAL INTELLIGENCE

1. Introduction

In the last decades the field of Machine Learning (ML) has been undergoing dramatic changes. It has witnessed the rise of Deep Learning (DL) and Artificial Neural Networks (ANN) in particular. Some concepts that can be attributed to DL have been known since the end of XX century, but for various reasons they did not get much appreciation by the ML community. It has changed with the adoption of Convolutional Neural Networks (CNNs) in image processing problems.

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

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 [23]. It covers topics that are essential to understanding complex learning systems like neural networks. Starting from the brief overview of most important machine learning concepts, this section concludes with evaluation basics.

2.1. Machine Learning Systems

The area of ML is creating a system that can perform a specific task without being explicitly programmed [24]. Instead, it should be sufficient to provide such system with empirical data of the process which behavior is being studied. This can be stated formally [25]:

> "A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improves with experience **E**."

This definition captures all important aspects of a ML problem.

- **T,** the task – the problem a ML system is intended to solve. The most common tasks, regression and classification. Regression models are trained to estimate unknown continuous variable based on given inputs. In classification a ML model learns to specify which of k categories some object belongs to.
- **P,** the performance measure – is a quantitative measure of how well the model performs. Optimizing the performance measure is one of the main goals of machine

learning. Usually the choice of the metric depends on the type of the task that machine learning system is dealing with. A a common practice is to use several metrics that give different perspectives on the model performance.

- **E,** the experience – In most ML algorithms experience is represented by a dataset – a collection of data points, that are represented by real-valued vectors $x \in \mathbb{R}^m$. Generally, ML problems require collecting empirical evidence and transforming it into the values that can be accepted by the learning algorithm. This process is commonly referred to as feature engineering. Although some branches of machine learning, like deep learning, deviate from this approach and make use of as much information as possible. For instance, a gray-scale image could be represented by a feature equal to the sum of its left-part pixel values subtracted from the sum of its right-part pixel values. Deep learning system would use the whole image as an input. When it is possible, it is convenient to represent the dataset with a matrix. A dataset consisting of $n \times m$-dimensional samples forms a matrix $X \in \mathbb{R}^{n \times m}$. Then *i*-th row of the X denotes a data point x_i.

A learning algorithm is an algorithm that is able to produce ML models using input data. Learning algorithms can be classified as supervised and unsupervised [23], albeit sometimes this boundary is too vague. Apart from that, there is also reinforcement learning, in which learning algorithms interact with dynamical systems "in real time" and receive feedback to evaluate the successfulness of their actions [26].

In supervised learning, each data point is associated with a target value, which provides information for learning and which the model tries to estimate when given a new, unseen input. There are no target values provided in unsupervised learning problems. Presence of the target value impacts learning methodology to a great extent. This thesis concentrates on supervised learning algorithms.

2.2. Supervised Learning

In supervised learning the task is to estimate (or to predict, which is the same in this context) one or more outputs (or targets) based on one or more inputs using empirical evidence (a dataset) that is collected beforehand [27].

Let $\mathbf{x} \in \mathbb{R}^p$ and $\mathbf{y} \in \mathbb{R}^q$. An input **x** causes a response in a form of the output **y**. This relationship can be described in a general form:

$$y = f(x) + \epsilon \quad \text{(II.1.)}$$

ϵ is a random error term, and $f: \mathbb{R}^p \rightarrow \mathbb{R}^q$ is an arbitrary mapping from the input space to the output space. Assuming that there is a dependence of **x** on **y**, it can be modeled with some function $\hat{f}: \mathbb{R}^n \rightarrow \mathbb{R}^m$ in a way that a chosen performance measure **J** is optimized with respect to the dataset $\mathbb{D} = \{(x_i, y_i) | x \in \mathbb{R}^p, y \in \mathbb{R}^q\}$

2.3. Learning Algorithms

Learning is a process of improving the performance measure with respect to the given dataset and generalizing knowledge beyond that particular dataset [28].

2.3.1. Parametric models

In case of supervised learning, training algorithms are provided with data that include the desired output of the system. Supervised learning algorithms are designed to make use of this information. Model is created or gradually improved through receiving explicit feedback on its performance in a form of quantitative performance measure.

Models can be classified as parametric and non-parametric based on the assumptions that should be made to apply them

- Parametric models are based on the assumption that the unknown target function f(·) lies in hypothesis space – a set of parameterized functions $\{f(x; \theta) | \theta \in \Theta\}$. Approximating function $\hat{f}(\cdot)$ is chosen from this set by finding a parameter configuration which optimizes the performance measure. Linear regression [29] and ANNs [30] are examples of parametrized models.
- Non-parametric learning algorithms make almost no assumptions about the unknown function of interest, and do not put constraints on the hypothesis space. Non-parametric models include KNN (K-Nearest neighbors) [23] and decision trees [23].

While the family of non-parametric algorithms is diverse, parametric learning is usually stated and solved in terms of mathematical optimization. The problem of finding optimal θ values with respect to the cost function $J : \Theta \rightarrow \mathbb{R}$ and the dataset $\mathbb{D}$ can be formulated as:

$$min_{\theta \in \Theta} J(\theta; \mathbb{D}) \quad \text{(II.2.)}$$

The solution to this problem depends on the special properties J might or might not possess [31].

2.3.2. Cost functions

The choice of the cost function depends on the specific ML task and sometimes on the properties of the given dataset. Most common cost functions for regression and classification are the mean squared error and the cross entropy.

- Mean squared error is generally used in regression problems [32] and measures the squared euclidean distance between the vector consisting of target values and the vector of estimated values.
- Cross entropy [33] is popular in classification tasks, where the output of the model simulates the simple or multivariate Bernoulli distribution [34]. That is, the model outputs probabilities of the input belonging to some classes. Loosely speaking, it measures the difference between two probability distributions.

2.3.3. Generalization

Aside from just improving performance measure it is desirable to achieve generalization [28]. A model is able to generalize if it performs well when given previously unseen inputs. To measure generalization error, dataset is divided into training and test parts. Data for testing do not participate in the learning process and are intended solely for model evaluation. A supervised learning model is underfitting if its error on the training dataset is undesirably high [30]. A model is overfitting if it achieves low training error but the test error is far higher than the training error [30]. Models with high number of parameters are able to overfit the training dataset by "memorizing" it. On the other hand, models with fewer number of parameters can be insufficient and prone to underfitting.

Test set error is only an approximation of the true error, which denotes the error on all previously unknown inputs. Sufficiently big test set can provide an adequate approximation, but if the test set does not contain enough samples this estimate is not reliable. Cross-validation [35] is a family of algorithms addressing this issue. The dataset is randomly divided into train and test (validation) parts, model is trained and evaluated. This process is repeated several times and the final test error is approximated with the average validation error. In the area of DL, cross-validation might be unfeasible to perform, because repeated training of a DL model can take unacceptable amount of time.

2.3.4. Regularization

Regularization is a way of constraining the hypothesis space of a model for the purpose of reducing the generalization error without increasing the training error.

In supervised parametric learning tasks regularization takes form of regularization penalty function $\Omega : \Theta \rightarrow \mathbb{R}$:

$$min_{\theta \in \Theta} L(\theta; \mathbb{D}) = J(\theta; \mathbb{D}) + \lambda \Omega(\theta) \quad \text{(II.3.)}$$

where $L\Theta \rightarrow \mathbb{R}$ is the total cost function and λ is a hyperparameter controlling the regularization strength. Hyperparameters are variables which does not participate in the learning process and should be predefined when constructing the model. Higher values of λ give the regularization part of the loss function bigger weight.

Common penalties are:

- L_2 loss $\Omega(\theta) = \|\theta\|_2^2 = \theta^T \theta$
- L_1 loss $\Omega(\theta) = \|\theta\|^1 = \sum_i |\theta_i|$

These penalties force optimization process to prefer solutions in which some parameter values are low or zero, thereby dynamically reducing the number of active parameters.

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 classifier's performance from different perspectives.

2.4.1. Confusion Matrix

The classification setting, designed to show classifier's performance from different perspectives. Binary classifier is a supervised learning model which maps input features to one of the two class labels, which are commonly called positive and negative labels. Often a binary classifier outputs values from the [0, 1] interval. This value can be interpreted as a probability of the input falling into the positive class. A specific evaluation system has been developed for binary classification. It is based on the confusion matrix, which displays all possible outcomes of the label estimation:

Table II.1. *Confusion matrix for binary classification*

	True label positive	True label negative
Predicted label positive	# of true positives, TP	# of false positives, FP
Predicted label negative	# of false negatives, FN	# of true negatives, TN

- Accuracy**,** indicates how many samples were classified correctly:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \quad \text{(II.4.)}$$

- Precision, or positive predictive value, measures the correctness of predicting positive condition:

$$PPV = \frac{TP}{TP + FP} \quad \text{(II.5.)}$$

- Sensitivity, also called recall, or true positive rate, measures the proportion of correctly classified true samples:

$$TPR = \frac{TP}{TP + FN} \quad \text{(II.6.)}$$

- False alarm, or false positive rate, measures the proportion of incorrectly classified positive samples to all negative samples:

$$FPR = \frac{FP}{TN + FN} \quad \text{(II.7.)}$$

2.4.2. Receiver Operating Characteristic Curve

ROC curve is created by computing true positive rate and false positive rate for different thresholds applied on the output of the binary classifier. ROC curve can be represented with a set of pairs:

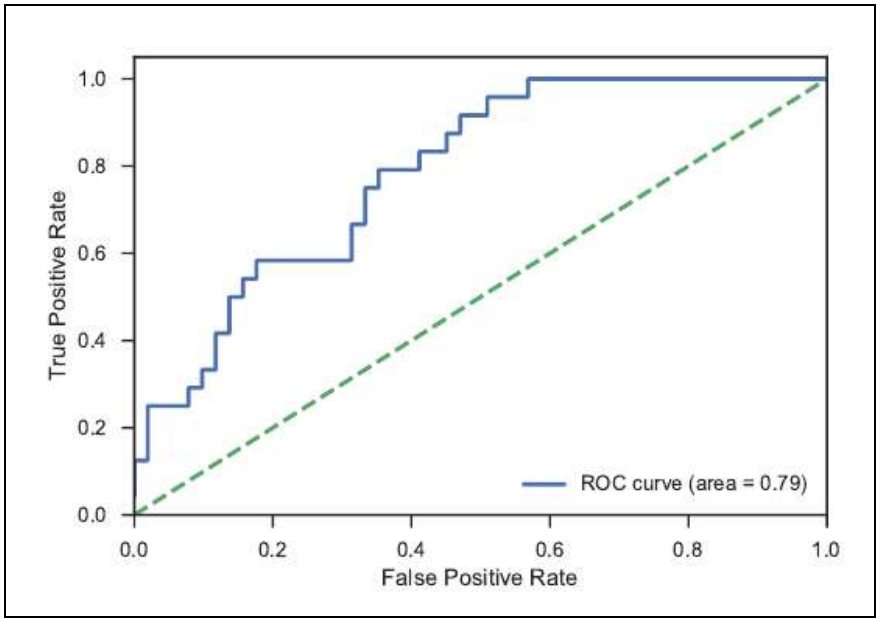


Figure II.1. *An example of the ROC curve*

2.4.3. Precision Recall Curve

PR curve is very similar to the ROC curve. PR curve uses precision (positive predictive value) instead of false alarm, and differs in the axis relative positions.

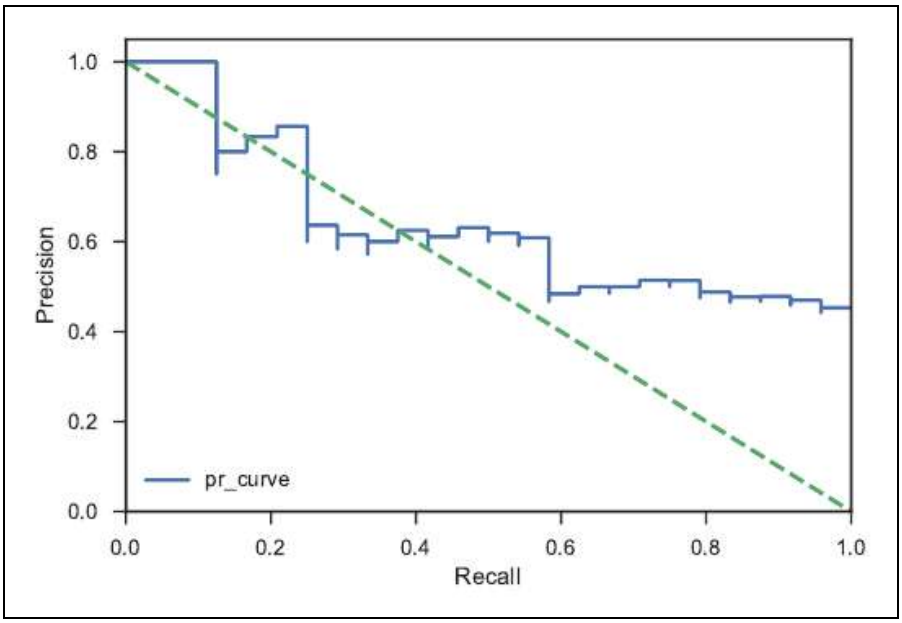


Figure II.2. *An example of the PR curve*

3. Artificial Neural Networks

A field of machine learning called DL is based on the idea of capturing hierarchical structure of data with composite models [36]:

"Deep learning allows computational models that are composed
of multiple processing layers to learn representations of data with
multiple levels of abstraction."

ANN is a parallel distributed connection system made up of simple processing units (commonly referred to as neurons) that has a natural ability for storing experimental knowledge and making it available for use [37].

ANN is a model consisting of multiple computational layers stacked on top of each other. Successive layers, are thought to learn features of increasing abstraction, looking "deep" into the data. At the origins of deep learning, biological neural networks were a big source of inspiration for artificial neural network research. ANNs are a major area of deep learning study and are successfully applied in many areas, such as image recognition [38], speech recognition [39], natural language processing [40] and bioinformatics [41].

This section covers notions which constitute the basis of ANNs, followed by multilayer perceptrones – the foundation stone of deep learning. The rest of the section describes the process of learning with neural networks.

3.1. Basic Concepts

ANNs originated as systems inspired by biological neural networks. They were intended to simulate animal's brain. So, by analogy, they consist of neurons and connections between them.

Artificial neuron was designed as a mathematical model of the biological neuron. It is a function transforming *n* real input "stimuli" to the single "response value" $f : \mathbb{R}^n \rightarrow \mathbb{R}$. The most common definition of the neuron is

$$f(x) = \phi(w^T x + b) \tag{II.8.}$$

Where $\mathbf{w} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}$ are the neuron's **weights** and **biases** respectively, and $\phi : \mathbb{R} \rightarrow \mathbb{R}$ is the activation function, which is described below. Weights and biases are the neuron's parameters, and their values are determined in the process of learning. Bias makes this transformation *affine*, allowing to learn a bigger variation of functions.

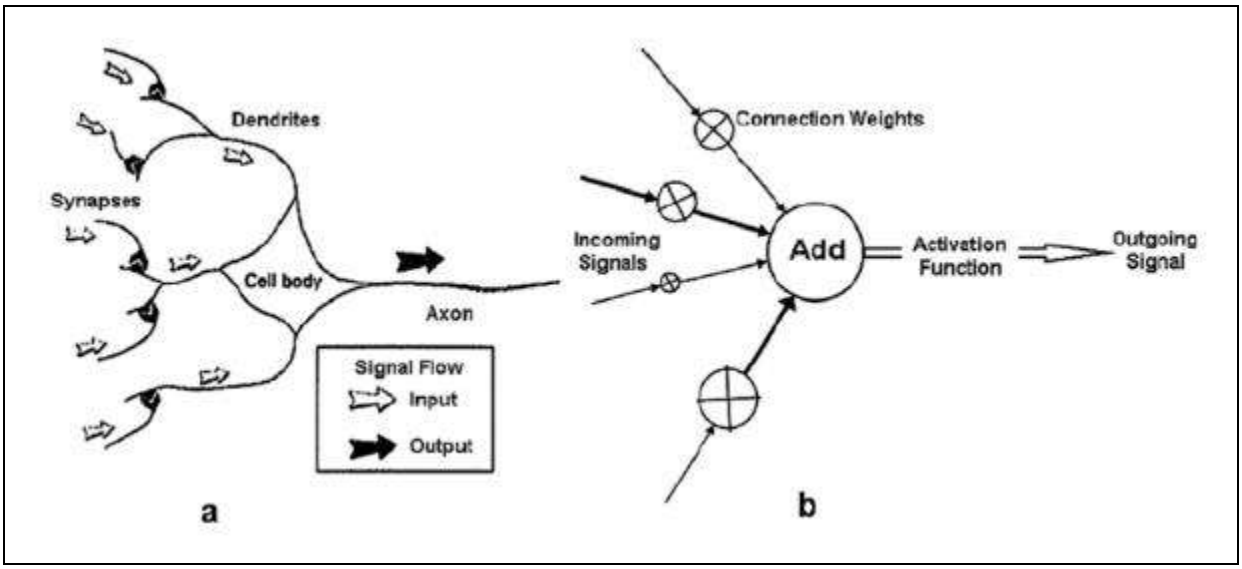


Figure II.3. *A biological neuron (a) compared to an artificial neuron (b).*

Activation function transforms the output of a neuron. It is applied element-wise on the output vector. The purpose of an activation function is to introduce non-linearity to the model in order to learn complex non-linear functions. Without activation functions hypothesis space of a neural network would be constituted only from linear functions. Popular examples of activation function include ReLU (Rectified Linear Unit) ReLU(x) = *max*(0, x) and sigmoid

$$\sigma = \frac{1}{1 + e^{-x}} \tag{II.9.}$$

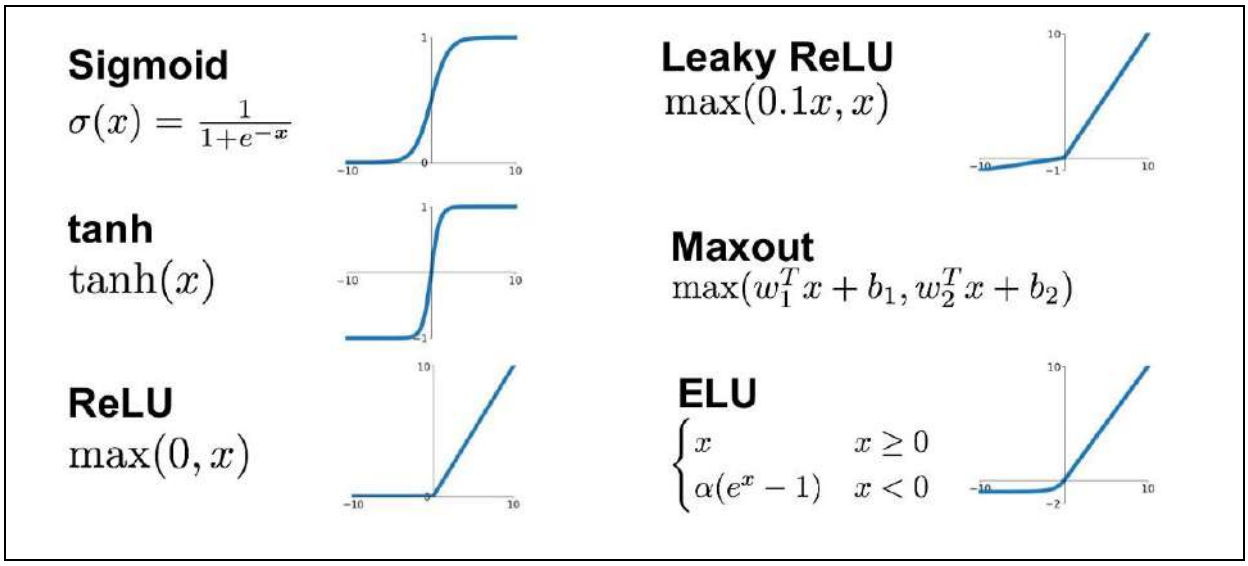


Figure II.4. *Popular activation functions[42]*

A layer, is a collection of neurons which are grouped to process multi-dimensional inputs to multi-dimensional outputs. ANNs are usually described in terms of layers, since it is more convenient due to the bigger scale and abstraction compared to artificial neurons. A notable example of a layer is a fully-connected layer. In a fully-connected layer neurons receive input from each neuron of the previous layer. This allows for convenient simultaneous computation of all neuron outputs of the layer via matrix multiplication:

$$f(x) = \phi(Wx + b) \tag{II.10.}$$

where **W** and **b** are individual neuron's weights and biases collected into a matrix and a vector respectively.

Network topology is a configuration of multiple chain connected layers. Two special layers are always present in the network: input and output ones. An input layer holds values of the input vector and is connected to the next layer only to pass it forward. An output layer is the last layer in the chain and its outputs are the final result of the input processing done by the network. Layers between them are called hidden, because their outputs are only intermediate and are not used outside of the model. The number of hidden layers is called the network's depth. The number of neurons in the layer is referred to as its width.

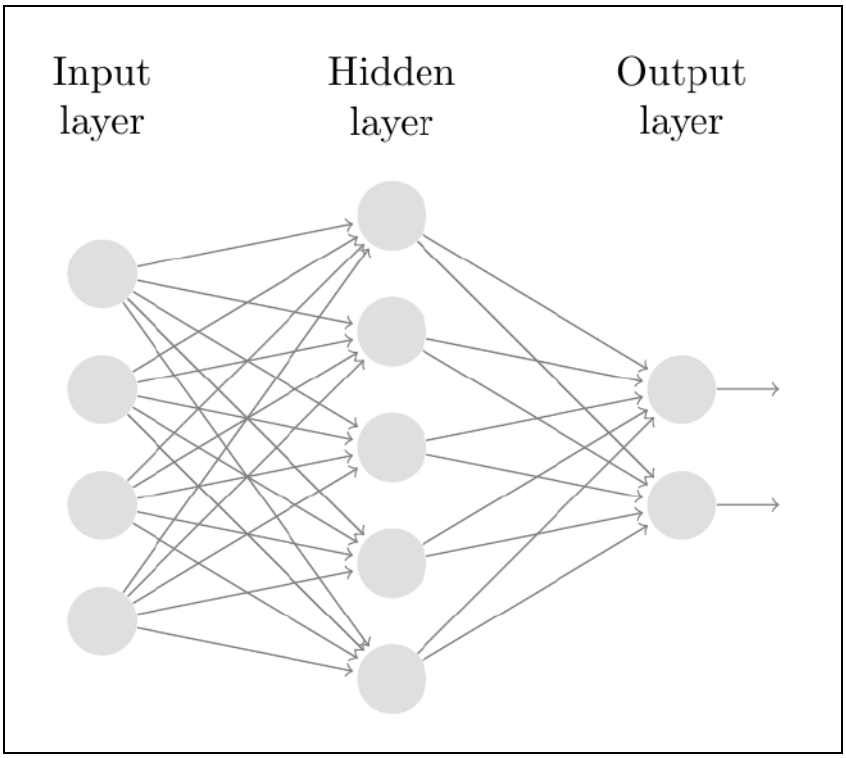


Figure II.5. *A simple artificial neural network example.*

It is worth noting that modern deep learning research and applications deviated from the initial biological and neuroscientific origins in order to develop more task-specific systems that concentrate on achieving better performance using various heuristics. Modern Deep Learning primarily relies on mathematics and computer science [30].

3.2. Feedforward Neural Networks

3.2.1. Single-Layer Perceptron

Single-layer perceptron [43] is the simplest kind of a feedforward neural network. In the basic form, it consists of only one artificial neuron. It can be extended to produce multi-valued output via additional units for each output element, but it still would not have any hidden layers. This type of neural network has very limited learning potential. Due to the absence of hidden layers, outputs are just a linear combination of inputs, so the single-layer perceptron will only be able to approximate linear relationships [44].

Perceptron function is the same as for the artificial neuron in the equation. Perceptrons can be trained by a simple learning algorithm that is usually called the delta rule [45]. It calculates the errors between calculated output and sample output data, and uses this to create an adjustment to the weights.

Algorithm: Delta rule learning

Require: $f : \mathbb{R}^n \rightarrow \mathbb{R}$ ▷*Perceptron function*
Require: $\boldsymbol{\alpha}$ ▷*Learning rate parameter*
Require: $\mathbb{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ ▷*Training dataset, $y_i \in \mathbb{R}_n$ is the desired response to the input vector $x_i \in \mathbb{R}_n$*

for (x_i, y_i) in $\mathbb{D}$ **do**
$\hat{\boldsymbol{y}}_{\boldsymbol{i}} \leftarrow f(x)$
$\boldsymbol{w} \leftarrow \mathrm{w} + \alpha(\mathrm{y_i} - \hat{\mathrm{y}}_\mathrm{i})\acute{\phi}(wx)x_i$
end for

3.2.2. Multilayer Perceptron

Multilayer perceptron (MLP) is a neural network that has at least one hidden layer and all layers are fully-connected. MLP is the basic tool and the basis of many advanced techniques in the DL. It is a cornerstone of deep learning.

Mathematically, MLP is a composite function that alternates affine and non-linear transformations, mirroring the graphical structure of the network :

$$f(x) = g(W_l\phi_{l-1}(W_{l-1}(\ldots\phi_1(W_1x + b_1)\ldots) + b_{l-1}) + b_l) \qquad \text{(II.11.)}$$

where $\mathbf{W}_i$, $\mathbf{b}_i$, ϕ_i are the *i*-th layer's weights, biases, and activation function respectively, and $g(\cdot)$ is the output function. The equation above depicts a multilayer perception with a single hidden layer.

MLP can be used in a broad range of tasks – according to the universal approximation theorem [46], a MLP with at least one hidden layer and non-linearity containing a finite number of units can approximate any continuous function with any desired non-zero error given appropriate parameters. However, the theorem is not constructive and it does not provide a way to find such network configuration and its parameters. In practice optimization algorithm is not guaranteed to find parameters corresponding to the desired function. It means that designing and training a network may not lead to desired approximation quality. Nevertheless it is an important result that shows ANN's rich potential.

In practice MLP has a few problems that balance its attractiveness as a universal approximator. One of the biggest downsides of this model is the high number of trainable parameters due to the fact that all layers are fully connected. Number of parameters can become unfeasible to train for high dimensional data or wide hidden layers. It also limits the depth of the model. Not only deep and wide perceptrones are computationally hard to train, but they tend to overfit if dataset is not large enough because of their high capacity.

3.3. ANN Learning

ANN is a parametric model. In order to get meaningful results for the task we should find appropriate values for its parameters – weights and biases. Just like in any other ML problem, ANN requires choosing a performance measure and an algorithm which will allow the network to improve this measure with respect to the dataset. It can be achieved by means of mathematical optimization.

In this section we introduce two algorithms essential for ANN learning: gradient descent, which adjusts network parameters in a way that improves performance measure, and back-propagation, which computes the gradient of the cost function necessary for gradient descent.

3.3.1. Gradient Descent Methods

The optimization problem of training a neural network can be stated formally:

$$minimize\ L(\theta;\ \mathbb{D}) \qquad w.r.t. \qquad \theta\ \in\ \Theta \tag{II.12.}$$

Where $\mathbb{D}$ denotes the set of training samples, θ denote all parameters of the model under the parameter space Θ and L is the real loss function. Evaluating L naturally involves computing model output on each training sample $x \in \mathbb{D}$. In the case of ANNs it makes performance function non-linear, which leads to complex and often non-convex optimization problems. Such problems can be solved with iterative gradient-based algorithms.

Gradient of a real continuous function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is the column vector of its partial derivatives:

$$\nabla f(x) = \left[\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n}\right]^T \tag{II.13.}$$

Gradient is a special case of the Jacobian matrix J, which is defined for a multi-dimensional real continuous function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \tag{II.14.}$$

- Gradient Descent (GD) is an iterative procedure for greedy optimization of arbitrary functions. Informally, gradient is the vector that points in the direction of the greatest rate of increase of the function. If it is desirable to achieve maximum possible decrease in a value of a function at some point x, the gradient of the

function should be subtracted from this point. This operation is referred to as update rule. Update rule is repeatedly applied for a certain number of steps or until some stopping condition (e.g. relative difference of the previous and successive function values is under a certain threshold) is satisfied. This basic approach from optimization theory applies to ANN training. Training process involves computing gradient many times, so this step should be as fast as possible. In neural networks the back-propagation algorithm is used for gradient computation. It is the subject of the next section. GD update rule:

$$\theta_{k+1} = \theta_k - \alpha \nabla_\theta L(\theta; \mathbb{D}) \quad \text{(II.15.)}$$

where $\alpha \in \mathbb{R}$ is the learning rate, a hyperparameter used to control the magnitude of the update, which influences convergence of the algorithm. In the case of ANNs GD encounters problems which make the optimization particularly difficult: high number of variables (weights and biases combined) and typically high cardinality of a training set, which makes computing step of the gradient descent computationally infeasible, since it requires "feeding" the whole training dataset to the network.

- Batch Stochastic Gradient Descent (SGD) performs gradient update on the random subset of the original dataset, called batch. It addresses the GD issues. Main idea behind SGD is that computing the gradient on random batches gives sufficient approximations of the real gradient. This approach allows much faster computation in exchange for precision in calculations. Because of this the loss function usually fluctuates and does not decrease monotonically. SGD favors exploration over greedy exploitation of the loss function surface because of slightly random suboptimal gradient directions. In the context of SGD one pass through all batches is called an epoch, and processing a single batch is referred to as a step.
- SGD with momentum [47] is intended to simulate the movement of the ball that has mass. It accumulates momentum as the parameter point displaces downhill, accelerating in the direction of a minimum, albeit it does not stop there, oscillating around a minimum when it is reached. It is implemented by subtracting the scaled update vector of its previous step.
- Adaptive gradient (Adagrad) [48] adapts the learning rate of each parameter individually, performing bigger updates for the parameters that change little in time, and vice versa.

where $G_k \in \mathbb{R}^{p \times p}$ is a diagonal matrix with its diagonal entries equal to the sum of the squares of the $\theta(i)$ gradients up to the step k and ϵ is a small number that helps to avoid division by zero.

- Adaptive moment estimation (Adam) [49] is an algorithm that makes use of both adaptive learning rate and momentum techniques. Adam computes decaying averages of past gradients and past squared gradients which are then corrected to avoid biasing towards zero

3.3.2. Back-propagation Algorithm

Any kind of gradient descent algorithm requires computing the gradient numerous times. Evaluating the gradient of composite functions can be computationally expensive, and a loss function can potentially be quite complex as it involves evaluating whole neural network on many data samples. Back-propagation (BP) algorithm provides an efficient way of evaluating the gradient [50].

BP utilizes chain rule in order to compute loss function. The algorithm can be considered as a specialized version of reverse accumulation automatic differentiation [51]. Reverse accumulation traverses the expression of chain rule from outside to the inside, computing all subexpressions along the way.

Neural network can be represented as a computational graph of it, Back-propagation is designed in such way that it visits each node of the network once, avoiding repeated computations that arise while evaluating the gradient of compound function. The amount of computation required for BP scales linearly with the size of the computational graph of the network.

Back-propagation always starts with a forward pass, which computes the error on one training sample. It has to keep all intermediate computation results as they are used to calculate the gradient.

Once the error is evaluated, the backward pass begins. It is called so because it traverses from the output of the network to the inputs. From the point of symbolical computations, the expression of the chain rule is traversed from the outermost subexpression to the inner ones.

3.3.3. Regularization

In addition to L1 and L2 losses two most popular regularization techniques in the area of ANN are dropout and early stopping.

Dropout [52] is a regularization technique specific to neural networks. During the training, network units are randomly removed along with all incoming and outgoing connections. Dropout is controlled with the hyperparameter $p \in [0, 1]$ which denotes the probability that a neuron will be kept in the network. After the training all nodes are inserted back into the network. Dropout reduces overfitting because dropped nodes have been learning only from the part of the data.

Early stopping [28] is a regularization method that can be used in any iterative optimization algorithm. Usually stochastic gradient descent runs for some predefined number of epochs. Once a stopping criterion holds true, the training process stops immediately. Stopping criterion monitors training error and indicates if its value does not change with successive iterations, which means that the algorithm has stuck in a local minimum and further parameter adjusting is unnecessary and leads to overfitting.

4. Convolutional Neural Networks

CNN is a special kind of ANN that uses convolution operation in at least one hidden layer. They are successfully used at different tasks involving processing data with grid-based structure, especially images.

4.1. Convolution Operation

4.1.1. Convolution in Mathematics and Image Processing

Convolution is an operation of two functions producing a third function. In the most general case convolution is defined for continuous functions, but DL adopted its discrete version.

Let $f : \mathbb{Z} \rightarrow \mathbb{R}$ and $g : \mathbb{Z} \rightarrow \mathbb{R}$ be real valued functions and $D_f \subseteq \mathbb{Z}$ be the domain of $f(\cdot)$. Convolution is denoted with asterisk and is defined as follows:

$$(I * K)(t) = \sum_{a \in Di} I(a)K(t - a) \qquad \text{(II.16.)}$$

In deep learning terminology I is called the input, and K is called the kernel. Values of the kernel are called weights as in traditional ANNs.

In ML practice CNNs work with images, which are typically represented as rectangular arrays of real values. In such case, convolution is two-dimensional and discrete. Convolution is also commutative, which means that the input can be shifted instead of the kernel. After these modifications we obtain the following definition:

$$(I * K)(i,j) = \sum_{1}^{m}\sum_{1}^{n} I(i-m, j-n)K(m,n) \quad \text{(II.17.)}$$

$I \in \mathbb{R}^{w \times h}$ is the input image, $K \in \mathbb{R}^{m \times n}$ is the kernel and *i*, *j* are the output image pixel coordinates.

It is also worth mentioning that most modern deep learning libraries use cross-correlation instead of the convolution. The only difference between them is that cross-correlation does not "flip" the kernel:

$$(I * K)(i,j) = \sum_{1}^{m}\sum_{1}^{n} I(i+m, j+n)K(m,n) \quad \text{(II.18.)}$$

This fact simplifies training process and does not influence learning results. If the kernel is flipped, then weights learned by CNN would simply have different indexes.

Convolution has many applications in different areas. Most importantly for the purposes of this text – in statistics, signal and image processing. In those areas it is often used for mixing two functions, filtering the original signal or applying various weighting functions to the input in order to highlight particular patterns or features of the input. CNN also use this operation for the purpose of feature detection. The important part is that networks not just use predefined hand-crafted kernels, but learn them in order to transform the input to the most informative representation. In deep learning terminology the output of convolution is sometimes denoted as the feature map [30].

4.1.2. Convolution in Image Processing

As it was previously mentioned, convolution is widely used in image processing. Using different kernels (also called filters), it is able to detect various image features, such as edges and blobs. Furthermore, it is used for changing image's visual representation, e.g. blurring or sharpening the picture.

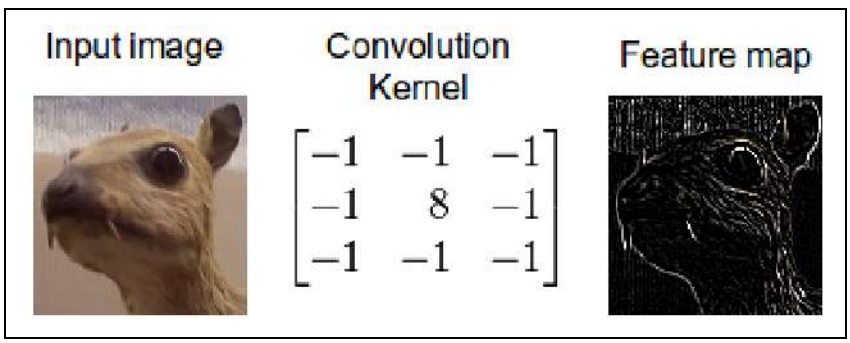


Figure II.6. *Edge detection with convolution [53].*

In this context, usage of convolution in neural networks is a heuristic adopted from the area of digital image processing. Taking advantage of the special form of the data, i.e. images, we can assume that specialized techniques will be more effective than universal multilayer perceptrones.

The main difference between standard image processing algorithms and CNNs is that in the latter case kernel's weights are learned through optimization algorithm. This allows neural networks to learn specialized filters for each task.

4.1.3. Convolution in Neuroscience

Another source of inspiration for CNN design is early neuroscientific research on how mammalian visual system works, especially the study [54] of cat's brain responses to visual stimuli. DL adopted some concepts revolving around the part of the brain called V1, or primary visual cortex.

In a very simplified description of the process of visual input processing, image in the form of light stimulates the area of the eye called retina and eventually this signal reaches V1. V1 is known to have a two-dimensional structure in which each neuron responses only to certain region in retina neuron's local receptive field. Despite that V1 is the most known part of this process, it is supposed that the input signal undergoes through several systems similar to V1 in a repeated fashion. Multiple layer stacking technique in DL mirrors those ideas about the visual system. After several layers of processing, the input reaches cells that individually respond to particular concepts disregarding the way they were seen, e.g. seeing someone on the photograph or in person. In CNN it is done through the output layer, which learns to take values depending on the last feature map and the target output.

4.2. Convolutional Network Architecture

4.2.1. Convolutional Layer

In CNN convolution is used to produce a feature map from the input, which can be the original image or another feature map. The main goal of using convolution in ANNs is to take advantage of special structure of the input and to learn how to transform it to the most informative form. In practice, convolutional layer's behavior is controlled with a set of hyperparameters, which brings flexibility to the design of neural networks and allows to adapt them to various problems:

- Kernel size defines the dimensions of the convolution kernel. It controls the area of the input to which neurons are sensitive. Choice of the appropriate value for this parameter almost always depends on the dataset. One way to do this is setting the

kernel shape of the first layer according to the scale of images to capture important details, e.g. edges. However, no rule of thumb exists for deeper layers and the optimal kernel size is determined experimentally. In the case when the input contains multichannel images or any three-dimensional data kernel itself is often three-dimensional.

- Number of kernels controls the number of dimensions of the layer output, since each kernel would produce distinct feature map. Increasing number of kernels might help to reduce information loss in architectures where feature map size tends to decrease with each layer. It also controls the capacity of the model – with the number of kernels the total number of trainable parameters grows.
- Padding**,** convolution is undefined on the borders of the input, since some part of the kernel cannot be matched with any input value. In order to overcome this problem and apply convolution in corner cases, input can be framed with zeros. The number of the input values for which the convolution is defined directly influences the size of the output, so padding can be used to control it.
- Stride controls the "step" of the convolution filter. Stride value of two would mean that after applying convolution on some pixel, two pixels in all dimensions are skipped. By manipulating stride we can regulate overlapping of different receptive fields and reduction of the output size. Let nin, nout, k, p, s be the total number of the inputs and outputs, total kernel size, padding size and the stride, respectively. Then the following relationship holds true [30]:

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1 \qquad \text{(II.19.)}$$

4.2.2. Pooling

A typical convolutional layer consists of three stages:

1. Applying convolutions to produce intermediate results.
2. Passing intermediate results through non-linear activation function, like in the standard multilayer perceptron. It is sometimes called detection stage.
3. Applying a pooling function.

Pooling function replaces rectangular areas of the input with their summary. It can be viewed as a non-linear downsampling method.

Let $[a_{i,j}] = A \in \mathbb{R}^{n \times m}$ be a real matrix that represents feature map region which is being passed to a pooling function. Most commonly used pooling methods include:

- Max pooling, which replaces the input region with its maximum value:

$$p(A) = \max(A) = \max\left(\{a_{i,j} | i \in 1, \dots, n, j \in 1, \dots, m\}\right) \quad \text{(II.20.)}$$

- Average pooling and weighted average pooling aggregate the input region by taking its average or a weighed sum, which can be based on the distance from the region's center:

$$p(A) = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} a_{i,j} \quad \text{(II.21.)}$$

- L2 pooling computes the L2 norm of the vector constructed by "unfolding" the input region.

$$p(A) = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} a_{i,j}^2} \quad \text{(II.22.)}$$

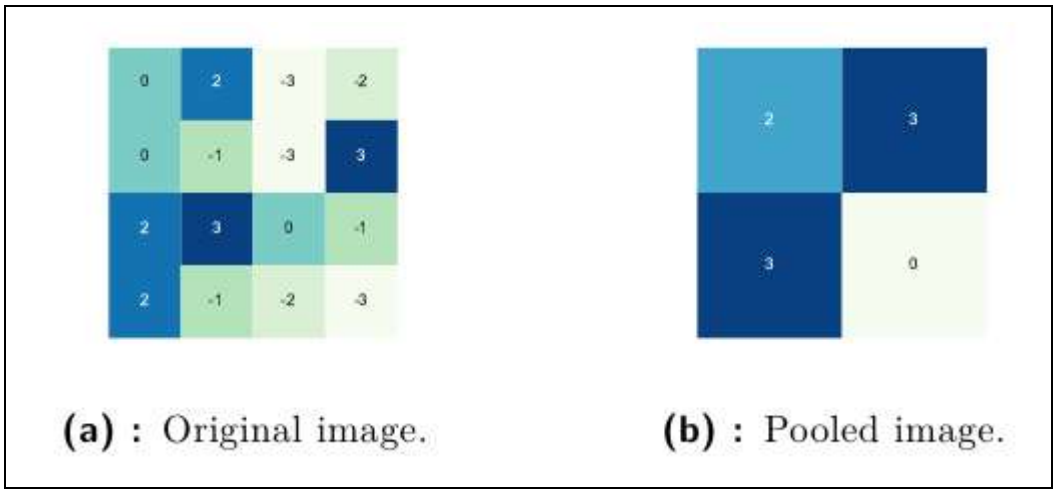


Figure II.7. *Illustration of the 2x2 max pooling function [55]*

Pooling is sometimes considered to be a separate layer, but in most popular architectures it always follows convolution and can be considered as its part. However, it can be situational and there are successful architectures which use pooling following several stacked convolution operations or which do not use pooling at all [56]. It can be adjusted with the padding and stride hyperparameters, analogous to convolution.

Pooling is used to reduce the number of variables in the model, thus preventing overfitting and improving its computational efficiency. Although heavy use of pooling can lead to aggressive information loss and underfitting, so it should be treated lightly. Another purpose of using a pooling function is introducing translational invariance into the model [30]. By considering whole regions instead of separate input values, pooling helps to put a bigger

emphasis on the feature's value regardless of its location. As a result, the network becomes resistant to small perturbations of the input.

4.2.3. Batch Normalization

Batch normalization layer [57] normalizes the output of the layer before it. By centering and scaling the feature maps batch normalization makes the gradient computation more robust because. The main goal of the layer is to discard the change in the distribution of hidden layer outputs, which can happen in the process of training. This simplifies learning and provides faster convergence toward the minimum. The values of $\gamma \in \mathbb{R}$ and $\beta \in \mathbb{R}$ are determined in the process of learning.

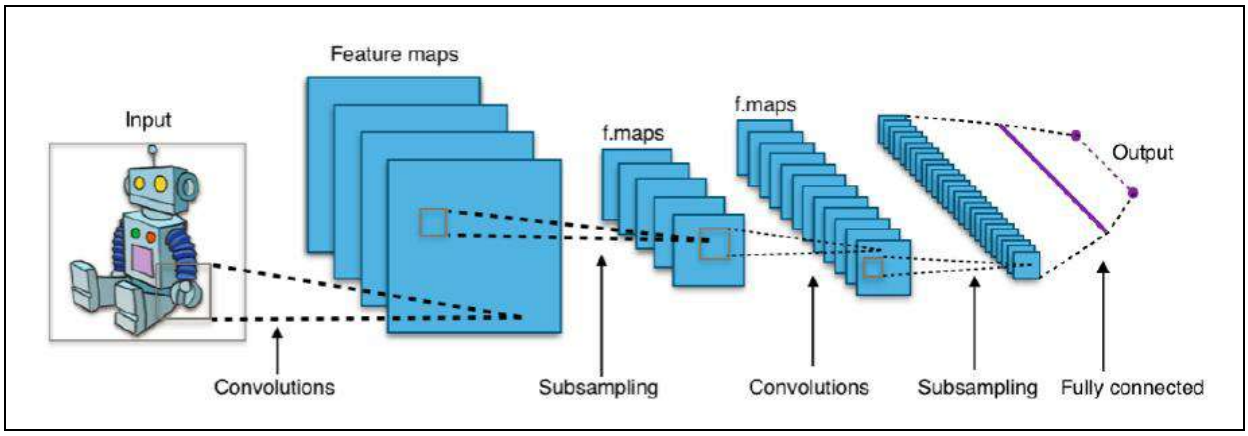


Figure II.8. *CNN illustration [58].*

4.3. Main Concepts Behind CNNs

This section describes most important ideas behind CNNs and tries to explain why they improve a normal ANN system.

4.3.1. Local Receptive fields

In traditional fully-connected layers every output is connected with every input. In CNN, neuron connections are restricted only to the subset of adjacent inputs through kernel. This subset of neurons is sometimes referred to as neuron's local receptive field.

This approach is aimed at the detection of local patterns and small details in the input.

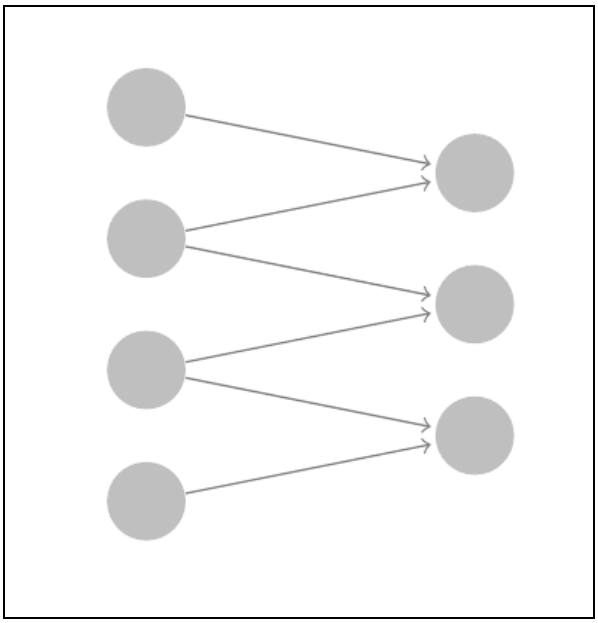


Figure II.9. *Local receptive fields*

Despite the typically small size of the receptive field, neurons located in the deeper layers indirectly interact with the larger area of the input. By the nature of the convolution, if layer's output is defined to be smaller than the input, its feature map will consist of aggregated input regions. Following this principle, feature maps in the deepest layers of the network will assemble "dense" information describing the interactions between different regions of the input.

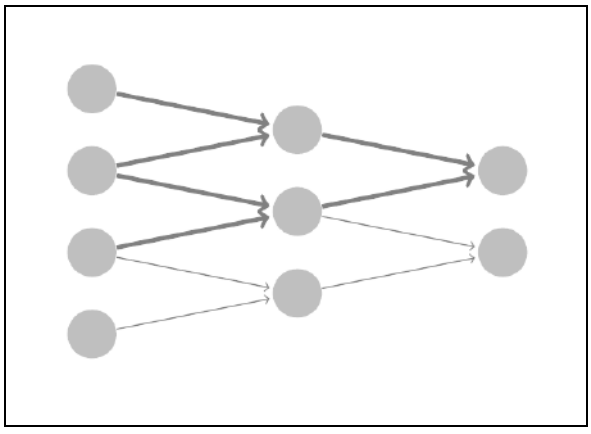


Figure II.10. *Deep indirect interactions*

4.3.2. Parameter Sharing

One of the most substantial downsides of the multilayer perceptron is large increase in the number of trainable parameters with each new layer added to the model, which makes models impractical to train and restricts the width and the depth of the network. In the meantime, convolutional layers have far less parameters that are shared across the whole layer.

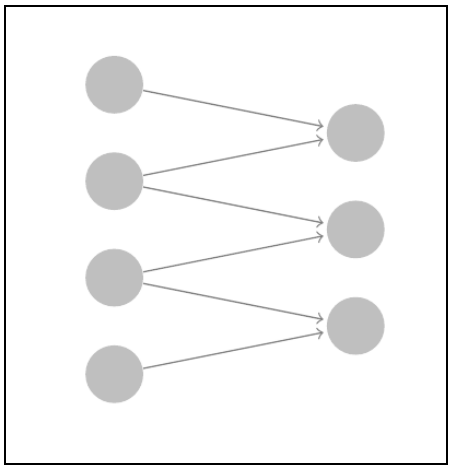


Figure II.11. *Parameter sharing*

Let m, n be the number of neurons in two consecutive layers, resp. Then traditional fully-connected layer would have $n(m + 1)$ trainable parameters (with the addition of biases). But convolutional layer with the kernel size of $w \times h$ ends up with the total of $wh + n$ weights and biases. This number can be sufficiently less than that of the fully-connected layer.

Thus, parameter sharing results in a smaller number of model parameters and positively influences computational and storage costs during the training and prevents overfitting. However, it is worth noting that in some problems it is reasonable to have unshared parameters. For instance, in the problem of the face recognition where the dataset contains centered and normalized images different filters would capture different information, although at a greater computational cost.

4.4. Popular CNN Architectures

- LeNet [59]. The first successful applications of CNNs. LeNet architecture was used to read zip codes, digits, etc.
- AlexNet [60]. The first work that popularized CNNs for Computer Vision was the AlexNet. The AlexNet was submitted to the ImageNet ILSVRC 1 challenge in 2012 and took the first place with the top 5 error of 16% compared to the second place with 26% error. The network featured multiple convolutional layers stacked on top of each other without the intermission of the pooling layers.
- GoogLeNet [61]. The ILSVRC 2014 winner. Its main contribution was the development of an Inception Module. Inception module uses convolutional filter of different sizes and concatenates their outputs, letting the model decide which filter size does the best job at capturing features. Additionally, it uses Average Pooling instead

of Fully Connected layers at the final network layers, eliminating a large amount of parameters which arises in fully-connected layers.

- VGGNet [62]. The runner-up in ILSVRC 2014 was the network that became known as the VGGNet. It showed the importance of the network depth. The network contains 16 layers (without pooling) and only performs 3x3 convolutions and 2x2 pooling. However, VGGNet is quite expensive to evaluate and uses a lot of memory for its 140 million of parameters.
- ResNet [63]. Residual Network was the winner of ILSVRC 2015. It features special residual blocks that improve the optimization of the cost function and a heavy use of batch normalization. The architecture also rejected the pattern of using fully connected layers at the end of the network.
- Squeeze-and-Excitation Network [64] was the ImageNet 2017 winner. It introduced the mechanism of adaptive weighting of intermediate feature map channels in the hidden layers of the network, thus increasing the ability of the network to learn best representations of the input.

As it can be observed from this list, the field of CNNs is rapidly changing. The best improvements of CNN architecture are often based on simple ideas, but are very effective.

5. Conclusion

In this chapter we have presented the important notions that are related to deep learning (definition, Architectures....etc). As well as a general overview on deep learning, with an emphasis on the method chosen in our work which is CNNs. The next chapter, deals with the various materials used in our project; software as well as hardware.

CHAPTER III : MATERIALS USED

1. Introduction

This chapter provides a brief overview of the most important technologies that were used in the development of our project. First, we will go over the hardware we used, including what it is and what are its features, and then we will go over the major software technologies we used to build the website, database, and face recognition software.

2. Hardware used

2.1. Raspberry pi 4

The Raspberry Pi 4 is a low cost small sized computer. It was originally designed for educational purposes that would improve understanding hardware and programming. It runs an operating system based on Linux. Although slower than modern laptops, the same CPU can be found in some smartphones as well. [65]



Figure III.1. *Raspberry Pi 4 board*

Raspberry Pi 4 has the following specifications:[66]

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 2GB, 4GB or 8GB LPDDR4-3200 SDRAM (depending on model)
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports.
- Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
- 2 × micro-HDMI ports (up to 4kp60 supported)
- 2-lane MIPI DSI display port

- 2-lane MIPI CSI camera port
- 4-pole stereo audio and composite video port
- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
- OpenGL ES 3.1, Vulkan 1.0
- Micro-SD card slot for loading operating system and data storage
- 5V DC via USB-C connector (minimum 3A)
- 5V DC via GPIO header (minimum 3A)
- Power over Ethernet (PoE) enabled (requires separate PoE HAT)
- Operating temperature: 0 – 50 degrees C ambient

2.2. Relay

Relay is an electromechanical device that uses an electric current to open or close the contacts of a switch. The single-channel relay module is much more than just a plain relay, it comprises of components that make switching and connection easier and act as indicators to show if the module is powered and if the relay is active or not [67]. It has the following specifications:

- Supply voltage – 3.75V to 6V
- Quiescent current: 2mA
- Current when the relay is active: ~70mA
- Relay maximum contact voltage – 250VAC or 30VDC
- Relay maximum current – 10A Relay Module Specifications

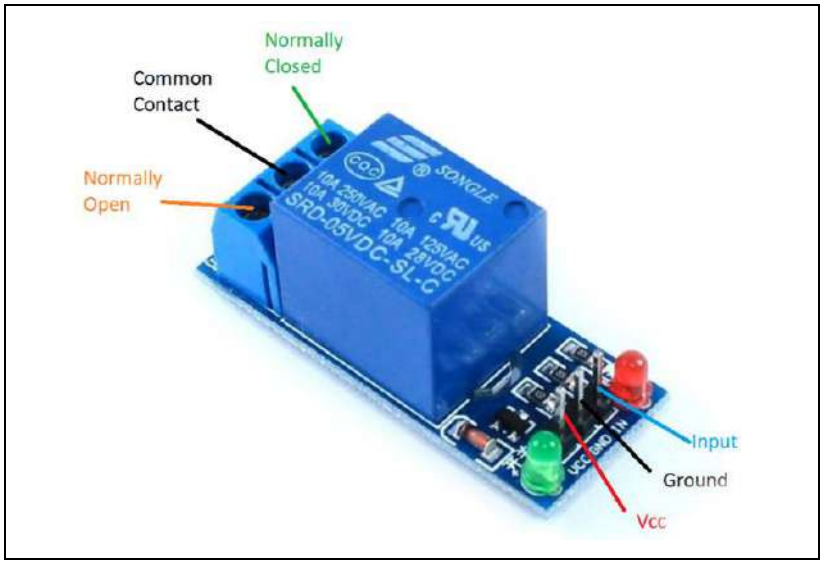


Figure III.2. *Single-Channel Relay Module Pinout*

2.3. Solenoid lock

Solenoids are basically electromagnets: they are made of a big coil of copper wire with an armature (a slug of metal) in the middle. When the coil is energized, the slug is pulled into the center of the coil. This makes the solenoid able to pull from one end.

This solenoid in particular is nice and strong, and has a slug with a slanted cut and a good mounting bracket. It's basically an electric lock, designed for a basic cabinet or safe or door. Normally the lock is active so you can't open the door because the solenoid slug is in the way. It does not use any power in this state. When 9-12VDC is applied, the slug pulls in so it doesn't stick out anymore and the door can be opened.[68]



Figure III.3. *Solenoid lock*

2.4. Raspberry Pi Camera Board v1.3

The Raspberry Pi Camera Board plugs directly into the CSI connector on the Raspberry Pi. It's able to deliver a crystal clear 5MP resolution image, or 1080p HD video recording at 30fps. Custom designed and manufactured by the Raspberry Pi Foundation in the UK, the Raspberry Pi Camera Board features a 5MP (2592x1944 pixels) Omnivision 5647 sensor in a fixed focus module. The module attaches to Raspberry Pi, by way of a 15 Pin Ribbon Cable, to the dedicated 15-pin MIPI Camera Serial Interface (CSI), which was designed especially for interfacing to cameras. The CSI bus is capable of extremely high data rates, and it exclusively carries pixel data to the processor. The board itself is tiny, at around 25mm x 20mm x 9mm, and weighs just over 3g, making it perfect for mobile or other applications where size and weight are important. The camera is supported in the latest version of Raspbian.

The Raspberry Pi Camera Board Features:[69]

- Fully Compatible with Both the Model A and Model B Raspberry Pi
- 5MP Omnivision 5647 Camera Module
- Still Picture Resolution: 2592 x 1944
- Video: Supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 Recording
- 15-pin MIPI Camera Serial Interface - Plugs Directly into the Raspberry Pi Board
- Size: 20 x 25 x 9mm
- Weight 3g

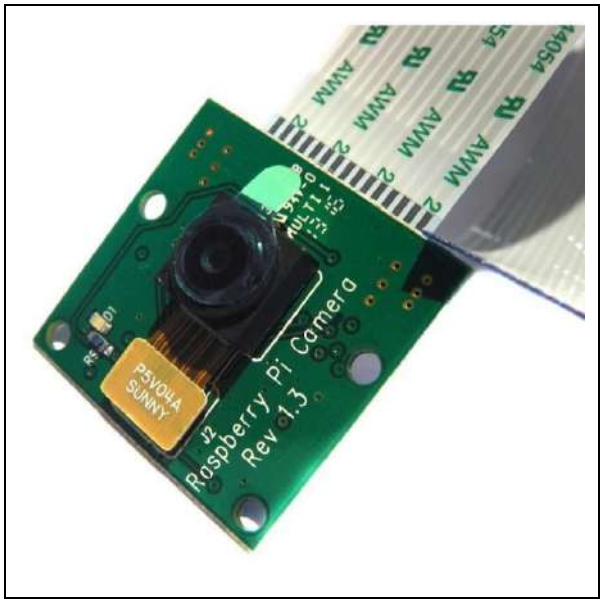


Figure III.4. *Camera RPi v1.3 Module*

3. Software used

3.1. Python

Python is a cross-platform open-source interpreted programming language. Python is a general purpose programming language created by Guido van Rossum in 1991. It has grown very popular among data scientist and researchers in general for its low barrier of entry and easy syntax. The most important selling feature of Python is its package creating community.

A package is a self-contained code library dealing with a specified task, e.g. working with tables or machine learning algorithms. As the number one choice for data scientists, there are many packages solving everyday tasks in machine learning research while abstracting complicated implementation away. We are going to present a few of those which we used while implementing the tasks of this thesis.



Figure III.5. *Python Programming Language Logo*

3.2. TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow is built hierarchically. Its lowest layers are CPU, GPU or TPU kernels making it versatile to work on any possible system. The core functions are written in C++ wrapped in a Python interface. It features many libraries for building custom machine learning models as well as testing them and evaluating them. TensorFlow mostly focuses on the implementation of neural networks.

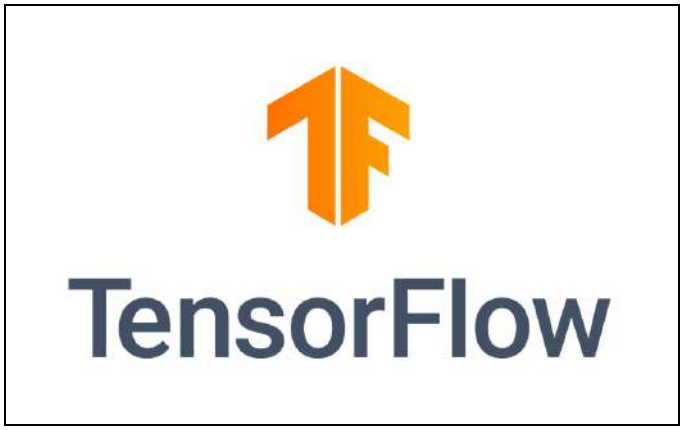


Figure III.6. *TensorFlow Logo*

3.3. Keras

Keras is an open-source software library written in Python and using TensorFlow, Microsoft Cognitive Toolkit, MXNet, Theano as a backend tensor manipulation library.

Keras is again developed for the purpose of deep learning. It's powerful feature is the content of extensible objects for defining different types of layers, models, loss functions and other tools widely used in the field of ANNs



Figure III.7. *Keras Logo*

3.4. OpenCV

OpenCV is one of the most used frameworks for computer vision and image processing. It is an open-source released under a BSD license and is free to use for academic or commercial applications. The library has more than 2500 optimized algorithms, which can be used in many applications such as face recognition, object identification and tracking, scenery recognition and many others. OpenCV's user community has more than 47 thousand users and the estimated download count of this library exceeds 7 milion (in the year 2015).

The OpenCV library supports Mac OS, Android, Windows and Linux in C++, C Python, Java and MATLAB interfaces. The library is written natively in C++ and has a templated interface that works seamlessly with STL containers. OpenCV version 4.1.0 was used for capturing the camera stream as well as processing the frames. [70].

3.5. Sqlite

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file [71].

3.6. Php

PHP is a server side scripting language. that is used to develop Static websites or Dynamic websites or Web applications. PHP stands for Hypertext Pre-processor, that earlier stood for Personal Home Pages. PHP scripts can only be interpreted on a server that has PHP installed. The client computers accessing the PHP scripts require a web browser only.[72]

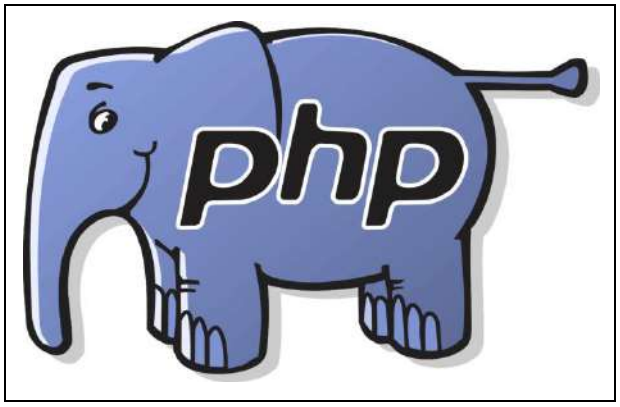


Figure III.8. *PHP Logo*

3.7. Tkinter

Tkinter is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library.

This Python framework provides an interface to the Tk toolkit and works as a thin object-oriented layer on top of Tk. The Tk toolkit is a cross-platform collection of 'graphical control elements', aka widgets, for building application interfaces.[73]

3.8. Jupyter and Google Colaboratory

"The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text."[74] It is used in scientific environments exactly for above mentioned features. Jupyter notebooks (formerly known as IPython) support Markdown for explanatory text, LATEX for mathematical equations and a large amount of popular programming languages for interactive applications, e.g.:

- Python
- Julia
- R
- Scala

Google Colaboratory (Colab) is a notebook environment with similar features to Jupyter. Additionally, Colab is completely cloud based and offers powerful hardware support for faster computation (see Table III.1.). That is very useful in the ML field since the most of its methods are computationally demanding.

Table III.1. *Offered specs by Google Colaboratory*

CPU	GPU	TPU
Intel Xeon Processor with two cores @ 2.30 GHz and 13 GB RAM	Up to Tesla K80 with 12 GB of GDDR5 VRAM, Intel Xeon Processor with two cores @ 2.20 GHz and 13 GB RAM	Cloud TPU with 180 teraflops of computation, Intel Xeon Processor with two cores @ 2.30GHz and 13 GB RAM

CHAPTER IV : REALISATION AND TESTS

1. Introduction

After having presented the different used tools, in this chapter, we will present in detail the technical and practical implementation of the proposed solution.

Before starting the project's design and implementation, we must first create a detailed description of our system, also known as the specification book; we must then conduct some experiments and preliminary testing to achieve good results in practice. For a better understanding of our project's general functioning, we will present the system's flow diagram and the specification book. Then, using CNN, we try different approaches to face recognition to find the most effective method; once we have that model, we will proceed to the implementation of our project.

2. Specification book

In order to carry out our smart authentication system, we have drawn outlines that are considered to be requirements of our project, making the purpose clear and precise.

2.1. Purpose

The purpose of this project is that of security, creating a safe environment for employees where one has to visually identify in order to authenticate to said space, thus eliminating all kinds of intruders, from thieves to potential attackers for they will be risking their identity if they attempt to authenticate themselves

2.2. Product features

- The system should be able to be integrated into a variety of spaces (example: safes, cabins, doors,etc.).
- The face detection must be robust and fast since the product can be deployed virtually in any environment.
- The system must have a very high accuracy rate, because it may be used in high clearance spaces.
- It must be able to compile reports of all access attempts as well as make it accessible over the network.
- The system must be extensible and easily expandable (E.g., the administrator should be able to add new users to the database, simply by adding their face to the database).

2.3. Interface requirements

- The users should be able to identify themselves using a graphical interface on a touch screen mounted near the door, it should have a camera mounted relatively close to the touch screen to maximize the quality of the biometric data (face aligned correctly).
- The door should have light indicators showing the state of the door (a red light is on at all times indicating that the door is locked, if a user is granted access a green light will momentarily light up, indicating for the user that the door is unlocked).
- The door must have an electromechanical lock so our system can take control of it and change its state depending on who's authenticating.

3. Global System flow diagram

The synoptic diagram corresponding to the overall operation of our system is shown in Figure IV.1. It is composed of three main parts: authentication, processing and cloud part. A user approaches the touch screen mounted on a physical access point in order to authenticate himself (1a). If it is engaged, the user's face is then captured through the camera (1b).The captured image is then forwarded to the processing unit (2), and passed to a face detection block where the user's face is extracted (3). The pre-processing block (4) resizes the extracted face to the right dimensions for our face recognition model. The processed face is then passed to a face descriptor block (5), in this step the developed CNN model transforms the face to a vector expressing facial characteristics. The feature matching block (6) attempts to calculate the similarity between the user's face against all the enrolled users in its database, and generates a report (7a) containing the authentication status, time, and face. The similarity result (7b) is then passed to the command unit where a decision is made; if the similarity rate is above a certain threshold, access is granted, otherwise the user is rejected. Depending on the results in "7b", the electronic lock (8a) is either engaged momentarily for the user to get in(successful attempt), otherwise it is kept locked. The light indicators (8b) follow the same pattern the electronic lock state, the red light is at all time on, if a successful attempt is made, a green light will momentarily light up. The cloud layer (website frontend) (11) allows the administrator to monitor all access attempts as well as add new users by uploading their face/name to the enrolled users database (10).All access attempts will be extracted on demand (when visiting the website) from the reports database.

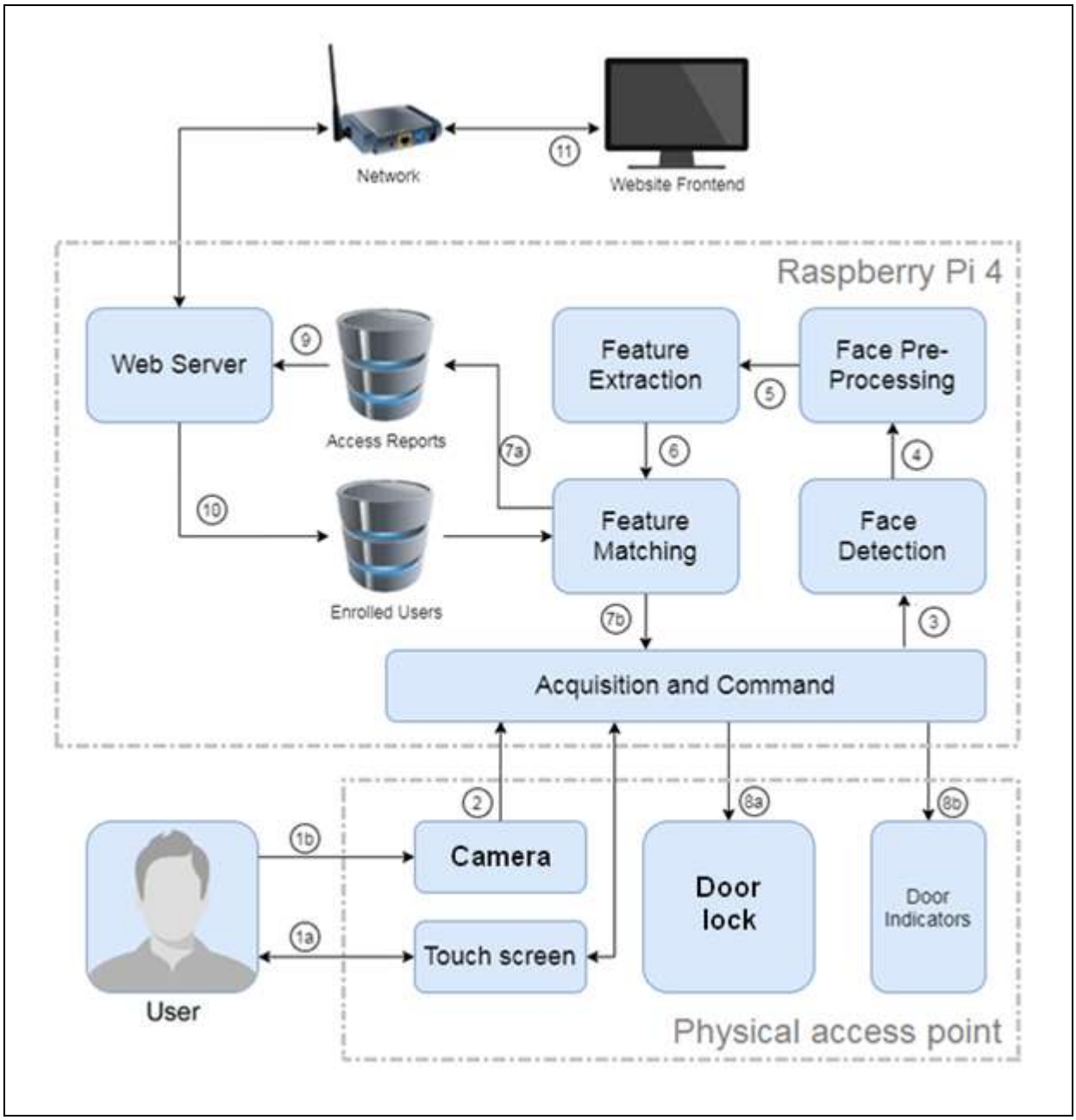


Figure IV.1. *System flow diagram*

4. Face recognition model

During our experiments on building a valid face recognition model we tried different approaches, one of which showed promising results but we will present more than one approach to discuss the problems we faced in the first attempt.

4.1. First attempt: CNN Face Classifier

The premise of this approach is very simple, a convolutional neural network model was built where we give it the user's face as input (in grayscale) and each user is assigned a class on the output layer.

4.1.1 Dataset

First we have to collect face samples of each user to create the dataset, since training CNNs is computationally expensive we decided to limit the users to a small number, for development purposes we decided to take face samples from existing databases, namely LFW (Labeled Faces In the Wild) database, we chose eleven diverse subjects and took around forty face samples per subject then we separated each subject into a folder named after them.

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 100 by 100 pixels grayscale images, as well as label each face with its owner on a separate vector (to be considered as output classes), to further facilitate dealing with images we flattened them to a 1-D vector, next we randomized the dataset to eliminate class discrimination during training, finally we saved the input images vector as well as the output classes vector in a numpy .npz file which is a way to save several arrays into a single file in uncompressed format.

It is necessary to note that we should use the same seed that we used to randomize our images during the training phase since our model have to go under many changes such as kernel size of the convolutional neural networks. Figure IV.2. shows a sample of images in our final dataset.

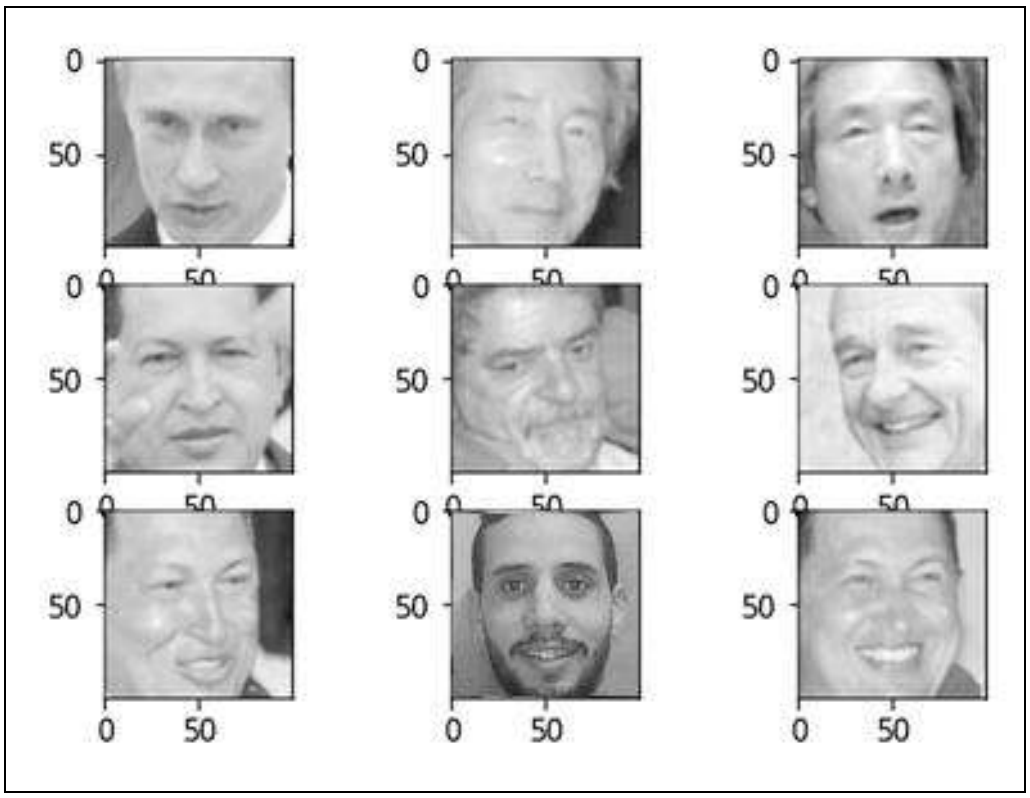


Figure IV.2. *Dataset sample images after pre-processing*

4.1.2 Training

The model described in this thesis is developed in Keras on top of Tensorflow. The decision to use Keras was made because it supports convolutional and recurrent networks as well as its ease of use since it has a higher level of abstraction meaning no boiler-plate need to be added which gives us the ability to pass directly to prototyping our model.

When defining the network layer configuration it is important to note that there does not exist an exact formula for optimal layer configuration. Instead, the best approach is to trial and error, where a few layer configurations are explored and benchmarked to see how well they work. However, in this case we will be showing only the configuration that yielded the best results and compare it with a random previous configuration attempt.

Training the deep convolutional network can take a variable amount of time, depending on the dataset size and available processing power. Three computation options are available to train CNNs, and choosing the most optimal one is crucial depending on the amount of available time to solve a particular task:

1. CPU based computation.
2. GPU based computation.
3. Cloud based GPU computation.

The CPU based computation option is the most simple and readily available. However since a CPU compute tasks in serial configuration, training the network takes the longest time using this method. The use of a Graphical Processing Unit cuts down the training time significantly for its Capability for parallel computation, and can be done with Keras without further programming. For this thesis we will be utilizing Google's cloud based GPU Since training CNNs is computationally expensive and Google offers the service "Google Colab" for free, Google's cloud-based GPU was used in our project.

The labeled dataset was divided into two, where 70% of the data is used to train the network, and 30% 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 configuration that yielded the best results is configuration 1 (Figure IV.3).

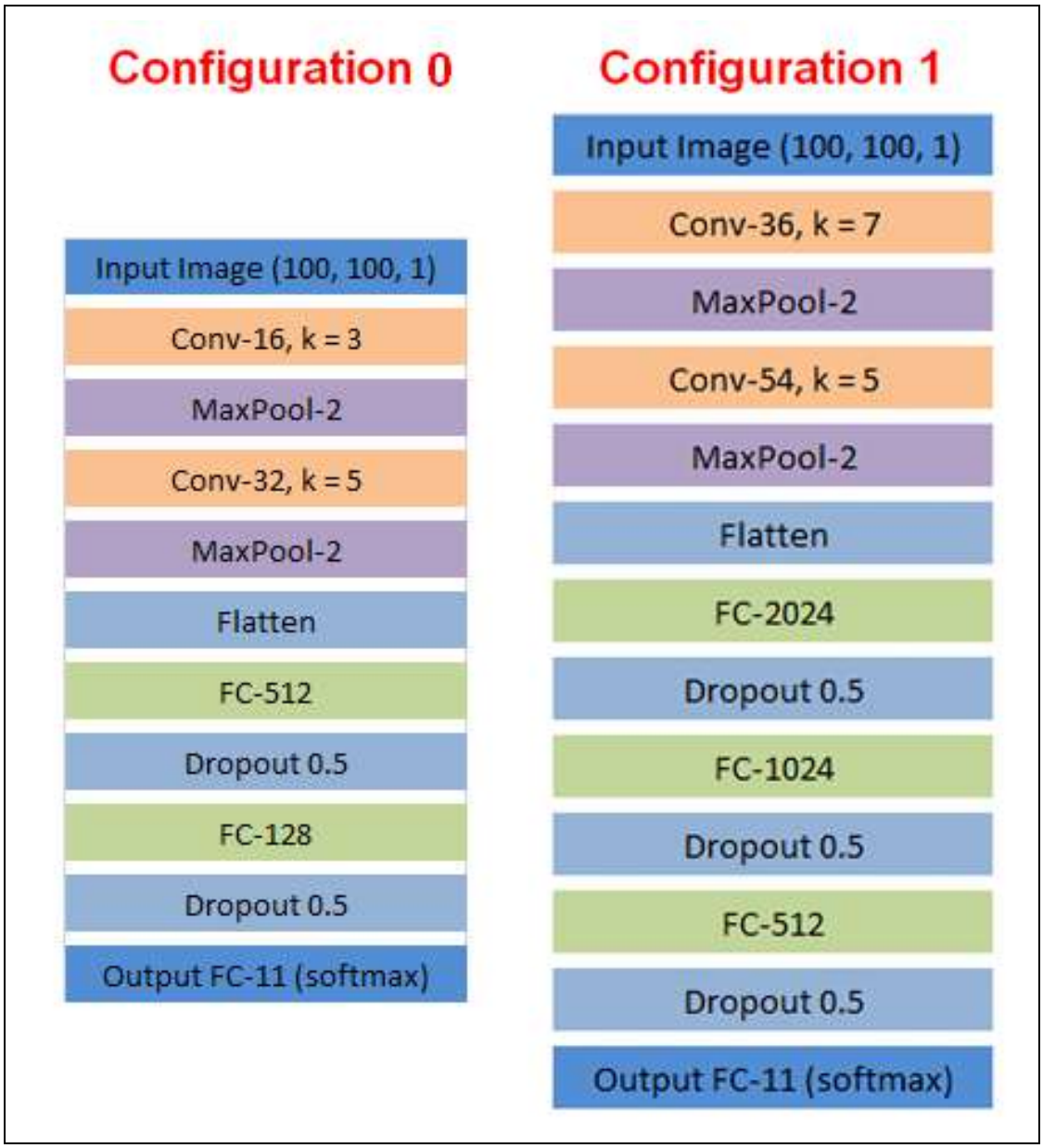


Figure IV.3. *CNN Configuration with the best accuracy rate*

4.1.3 Results

We will be discussing the results from the successful configuration (configuration 1) compared with one of the previous attempts (configuration 0) for reference (Figure IV.3).

From the two plots below (figure IV.4. configuration 0) , we can see that the validation accuracy became flat after a couple of epochs and rarely increased after. In the beginning, the validation accuracy was linearly increasing with loss, but then it did not increase much to the point of stopping.

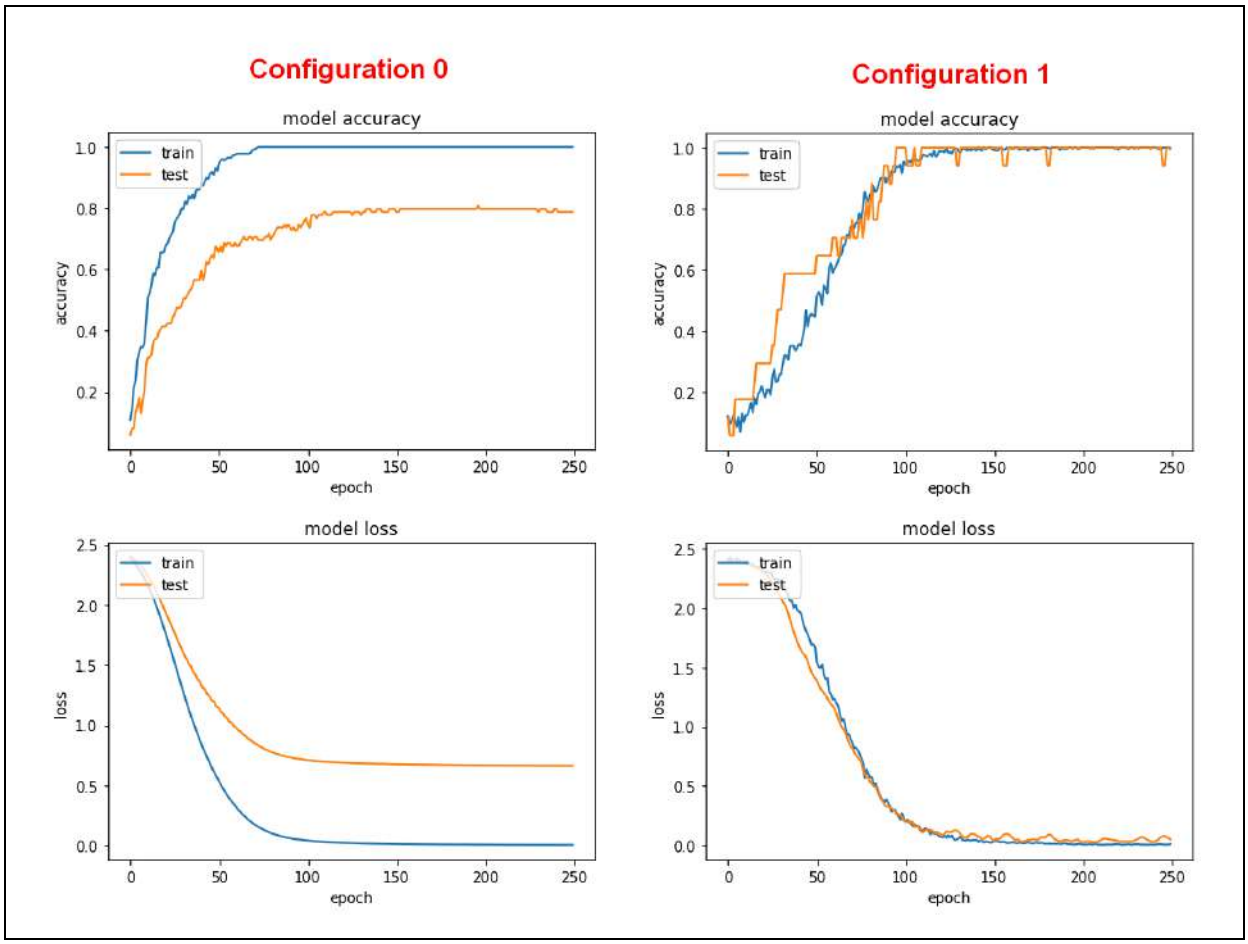


Figure IV.4. *Training loss and accuracy results*

The validation loss shows that this is the sign of overfitting, similar to validation accuracy it linearly decreased but some epochs later, it flattened. This means that the model tried to memorize the data and succeeded.

To overcome the problem of overfitting to some extent. Dropout randomly turns off a fraction of neurons during the training process, reducing the dependency on the training set by some amount. How many fractions of neurons we want to turn off is decided by a hyperparameter, which can be tuned accordingly. This way, turning off some neurons will not allow the network to memorize the training data since not all the neurons will be active at the same time and the inactive neurons will not be able to learn anything.

From Figure IV.4. (configuration 1) we see that adding dropout in our model worked, we can see that the validation loss and validation accuracy both are in sync with the training loss and training accuracy. It shows that our model is not overfitting: the validation loss is decreasing and not increasing, and there is almost no gap between training and validation accuracy.

Exposing configuration 1 to test data that wasn't part of the training dataset yielded an accuracy rate of 95.45% which is impressive considering the problem we are trying to solve.

Classification accuracy alone can be misleading since our data has more than 2 classes, we may have an accuracy rate of 95.45%, but we don't know if that is because all classes are being predicted equally well or whether one or two classes are being neglected by the model. Calculating a confusion matrix (Figure IV.5.) can give us a better idea of what our classification model is getting right and what types of errors it is making.

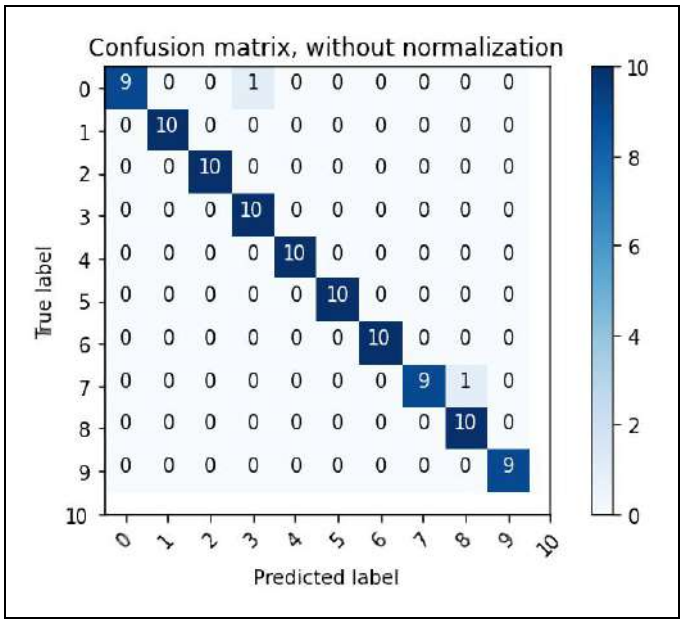


Figure IV.5. *Confusion matrix for configuration 1*

The diagonal elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier. The higher the diagonal values of the confusion matrix the better, indicating many correct predictions. Which is the case in our model as shown in Figure IV.5.

The off-diagonal elements usually hint that there is a class classification imbalance, although our model did misclassify some samples (see Figure IV.5.), it is very minimal thus we can say that our model is not only accurate but also precise and balanced, which can be further confirmed by looking at the F1 score shown in the table below (Table IV.1.)

The F1 score can be interpreted as a weighted average of the precision and recall values, where an F1 score reaches its best value at 1 and worst value at 0. From our results (Table IV.1.) we see that the classifier is underperforming for class 0 regarding both precision and recall. For class 4, the classifier is lacking precision but on overall the only significantly low F1 score belongs to class 0, which imply that our model may misclassify it more often than

others but on average the F1 score of other classes indicate that the model is indeed precise and not just memorizing faces.

Table IV.1. *F1-Score for configuration 1*

	precision	recall	f1-score
Class 0	0.89	0.80	0.84
Class 1	1.00	0.90	0.95
Class 2	0.91	1.00	0.95
Class 3	1.00	1.00	1.00
Class 4	0.83	1.00	0.91
Class 5	1.00	1.00	1.00
Class 6	1.00	1.00	1.00
Class 7	1.00	1.00	1.00
Class 8	1.00	0.90	0.95
Class 9	0.91	1.00	0.95
Class 10	1.00	0.90	0.95
Accuracy			0.95
Macro avg	0.96	0.95	0.95
Weighted avg	0.96	0.95	0.95

4.1.4 Additional tests

To confirm the precision of our model we put it through a series of tests on newly acquired data.

- First test: we gave the model a set of new faces that belong to classes it already knows. The results are shown in figure IV.6.These results are decisive that our model can classify faces from outside the dataset given that they are indeed represented by a class in the output layer.

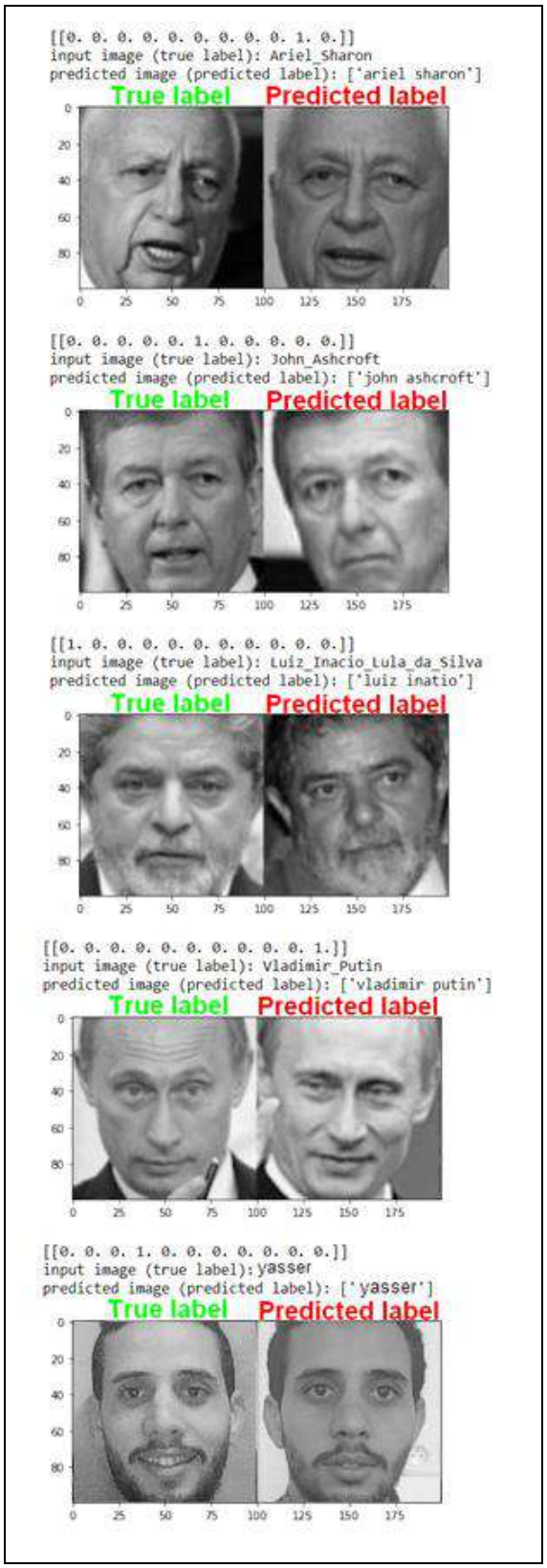


Figure IV.6. *Test results on known subjects*

- Second test: the model was subjugated to unknown subjects (people that are not represented by a class on the output layer) to see its reaction. The results are shown in figure IV.7. The results show that the model has a false positive rate of 100% on the

unknown subjects due to the face morphology being similar between races. Thus, the model will attempt to force any face to its closest neighbor, for example it is apparent that “Heizo takenaka” and “Junichiro Koizumi” posses similar facial characteristics (both come from east Asian background).The model identified “Heizo takenaka” as “Junichiro Koizumi” despite the fact that “Heizo takenaka” doesn’t exist in the output classes of our model. For these reasons, and to make our system precise and reliable we tried a second attempt.

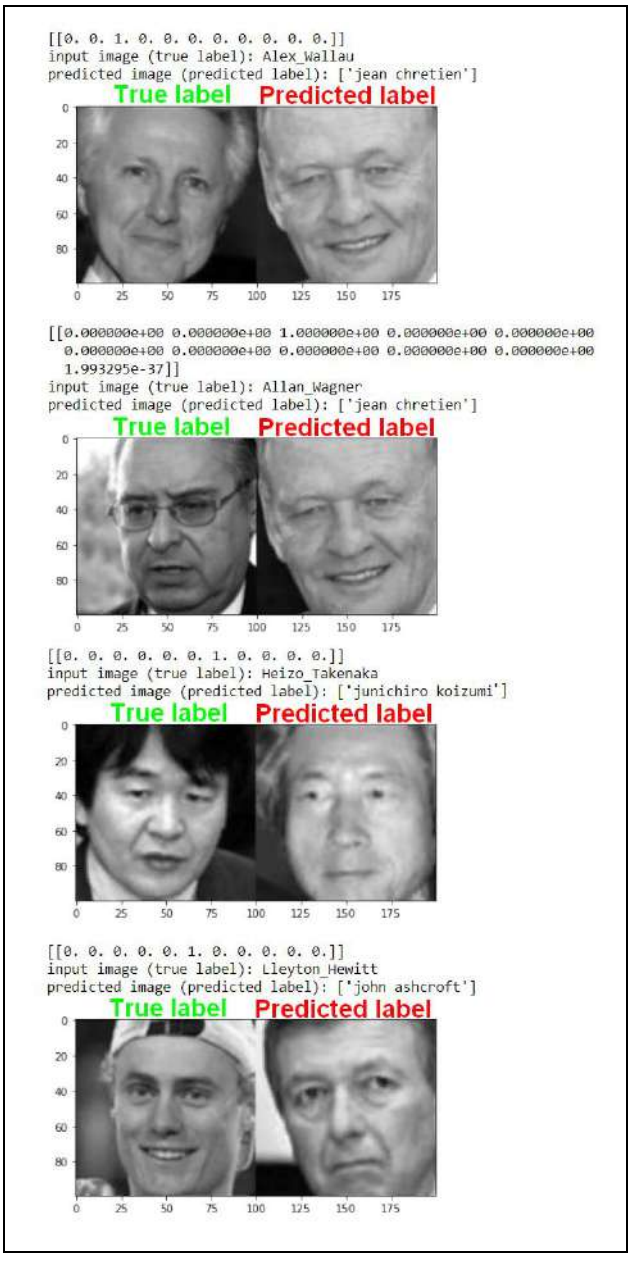


Figure IV.7. *Test results on unknown subjects*

4.2. Second attempt: CNN Face Descriptor

For our second attempt we decided to pursue a different path, a path that's taken by tech-giants. After reading up on the literature on this subject a pattern emerged, all major methods employ the CNN as if its describing the face instead of classifying it, for that we decided to experiment a bit using readily available models.

4.2.1 Hypothesis

From the previous test (Figure IV.7.), we hypothesize that face morphology in humans is not a unique identifier between people, although it varies from face to face, the general morphology of different racial groups and phenotypes is distinct from one another, so it is practically impossible to create a face classifier for a small set of people because it will more likely focus on learning the general morphology of our subjects instead of learning the exact features, since the network wouldn't know how other humans look it will clump them to their nearest phenotype instead of trying to guess which face it would be (i.e. matching Heizo takenaka as Junichiro koizumi (Figure IV.7.) for the sole purpose that they come from a common background).

To solve this problem we hypothesize that the last hidden layer of a CNN network trained on classifying faces actually represents the features of that face and not the face itself, so the more subjects there is and the more diverse our subjects would be, the more accurate the CNN will actually describe that face, by getting the feature vector from the last hidden layer we allow for matching faces by means of calculating the distance between them, the closer the two feature vectors are the more likely those two faces belong to the same person.

4.2.2 Model used

Since 2014, the emergence of Deep Learning techniques has reshaped the research landscape, launched by the breakthroughs of DeepFace and DeepID. Since then, deep learning technique, characterized by the hierarchical architecture to stitch together pixels into invariant face representation, has dramatically improved the state-of-the-art performance and fostered successful real-world applications.

The reason to choose a pre-trained model is the size of dataset used for training, the more faces we use the better the face descriptor will be, it's also infeasible to train the model from scratch due to time and processing power constraints as well as the resources to build a dataset containing millions of faces is not available,

Table IV.2 provide a comprehensive review of the recent developments on deep face recognition. Noting that Labeled Faces in the Wild (LFW) a database of face photographs designed for studying the problem of unconstrained face recognition. The data set contains more than 13,000 images of faces collected from the web. Each face has been labeled with the name of the person pictured. 1680 of the people pictured have two or more distinct photos in the data set. The only constraint on these faces is that they were detected by the Viola-Jones face detector.[77]

Table IV.2. *The accuracy of different methods evaluated on the LFW dataset [75]*

Method	Public. Time	Loss	Architecture	Number of Networks	Training Set	Accuracy±Std(%)
DeepFace [20]	2014	softmax	Alexnet	3	Facebook (4.4M,4K)	97.35±0.25
DeepID2 [21]	2014	contrastive loss	Alexnet	25	CelebFaces+ (0.2M,10K)	99.15±0.13
DeepID3 [36]	2015	contrastive loss	VGGNet-10	50	CelebFaces+ (0.2M,10K)	99.53±0.10
FaceNet [38]	2015	triplet loss	GoogleNet-24	1	Google (500M,10M)	99.63±0.09
Baidu [58]	2015	triplet loss	CNN-9	10	Baidu (1.2M,18K)	99.77
VGGface [37]	2015	triplet loss	VGGNet-16	1	VGGface (2.6M,2.6K)	98.95
light-CNN [85]	2015	softmax	light CNN	1	MS-Celeb-1M (8.4M,100K)	98.8
Center Loss [101]	2016	center loss	Lenet+-7	1	CASIA-WebFace, CACD2000, Celebrity+ (0.7M,17K)	99.28
L-softmax [104]	2016	L-softmax	VGGNet-18	1	CASIA-WebFace (0.49M,10K)	98.71
Range Loss [82]	2016	range loss	VGGNet-16	1	MS-Celeb-1M, CASIA-WebFace (5M,100K)	99.52
L2-softmax [109]	2017	L2-softmax	ResNet-101	1	MS-Celeb-1M (3.7M,58K)	99.78
Normface [110]	2017	contrastive loss	ResNet-28	1	CASIA-WebFace (0.49M,10K)	99.19
CoCo loss [112]	2017	CoCo loss	-	1	MS-Celeb-1M (3M,80K)	99.86
vMF loss [115]	2017	vMF loss	ResNet-27	1	MS-Celeb-1M (4.6M,60K)	99.58
Marginal Loss [116]	2017	marginal loss	ResNet-27	1	MS-Celeb-1M (4M,80K)	99.48
SphereFace [84]	2017	A-softmax	ResNet-64	1	CASIA-WebFace (0.49M,10K)	99.42
CCL [113]	2018	center invariant loss	ResNet-27	1	CASIA-WebFace (0.49M,10K)	99.12
AMS loss [105]	2018	AMS loss	ResNet-20	1	CASIA-WebFace (0.49M,10K)	99.12
Cosface [107]	2018	cosface	ResNet-64	1	CASIA-WebFace (0.49M,10K)	99.33
Arcface [106]	2018	arcface	ResNet-100	1	MS-Celeb-1M (3.8M,85K)	99.83
Ring loss [117]	2018	Ring loss	ResNet-64	1	MS-Celeb-1M (3.5M,31K)	99.50

In our project we chose to work with “VGGface”, the choice was partially arbitrary since most methods presented above have almost the same accuracy, in addition the “VGGface” is almost racially unbiased as shown in the table below (Table IV.3.).

VGGface is a deep face recognition architecture that was developed by the Visual Geometry Group of Oxford to compete in imagenet challenge in 2014 and was from the top 5 winning architectures.

Table IV.3. *Racial bias in existing commercial recognition APIs and face recognition algorithms [75]*

Model	LFW	RFW			
		Caucasian	Indian	Asian	African
Microsoft	98.22	87.60	82.83	79.67	75.83
Face++	97.03	93.90	88.55	92.47	87.50
Baidu	98.67	89.13	86.53	90.27	77.97
Amazon	98.50	90.45	87.20	84.87	86.27
mean	98.11	90.27	86.28	86.82	81.89
Center-loss [101]	98.75	87.18	81.92	79.32	78.00
Sphereface [84]	99.27	90.80	87.02	82.95	82.28
Arcface [106]	99.40	92.15	88.00	83.98	84.93
VGGface2 [39]	99.30	89.90	86.13	84.93	83.38
mean	99.18	90.01	85.77	82.80	82.15

Originally the purpose from VGGface was face recognition, from either a single photograph or from a set of faces tracked in a video. The structure of the VGG-Face model is demonstrated in Figure IV.8. The networks are “very deep”, in the sense that they comprise a long sequence of convolutional layers. Such CNNs have achieved state-of-the-art performance in some of the tasks of the ImageNet ILSVRC 2014 challenge, as well as in many other tasks.

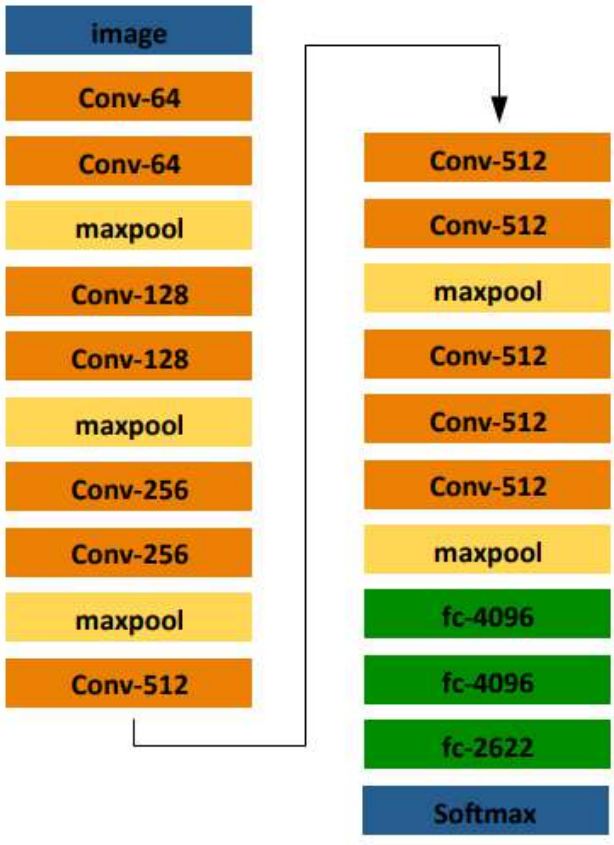


Figure IV.8. *Visualization of VGGface architecture.[76]*

It should be noted that the Visual Geometry Group of Oxford built their dataset from scratch, containing 2.6 million faces belonging to 2622 identities scraped from various sources online, with steps that can be summarized in table IV.4.

Table IV.4. *VGGface Dataset curation process[76]*

No.	Aim	Mode	# Persons	# images / person	Total # images	Anno. effort
1	Candidate list generation	Auto	5000	200	1,000,000	-
2	Candidate list filtering	Manual	2622	-	-	4 days
3	Rank image sets	Auto	2622	1000	2,622,000	-
4	Near duplicate removal	Auto	2622	623	1,635,159	-
5	Manual filtering	Manual	2622	375	982,803	10 days

In order to allow for a direct comparison to previous work, while the CNNs are trained on the new dataset of table IV.4 evaluation is performed on existing benchmark datasets (which contain different identities from the dataset used for training).

The first dataset is the Labeled Faces in the Wild (LFW) dataset [77]. It contains 13,233 images with 5,749 identities, and is the standard benchmark for automatic face verification. they followed the standard evaluation protocol defined for the "unrestricted setting" using data external to LFW for training, which in their case is the new face dataset. In addition to the verification accuracy Acc., they use the Equal Error Rate (EER) as an evaluation metric, defined as the error rate at the ROC operating point where the false positive and false negative rates are equal. The advantage on Acc. is that that it is independent of the distance threshold τ.

The second dataset is the YouTube Faces (YTF) dataset. It contains 3,425 videos of 1,595 people collected from YouTube, with an average of 2 videos per identity, and is a standard benchmark for face verification in video. Again, they followed the standard evaluation protocol defined for the "unrestricted setting", and report EER.

The table IV.5, and figure IV.9 below represent a comparison between famous state-of-the-art methods compared to VGGface.

VGGface achieves comparable results to the state of the art using much less training data (than DeepFace and FaceNet) and using a simpler network architecture (than DeepID-2,3).

Table IV.5. *LFW unrestricted setting results[76]*

No.	Method	# Training Images	# Networks	Accuracy
1	Fisher Vector Faces	-	-	93.10
2	DeepFace (Facebook)	4 M	3	97.35
3	DeepFace Fusion (Facebook)	500 M	5	98.37
4	DeepID-2,3	Full	200	99.47
5	FaceNet (Google)	200 M	1	98.87
6	FaceNet+ Alignment (Google)	200 M	1	99.63
7	Ours (VGG Face)	2.6 M	1	98.78

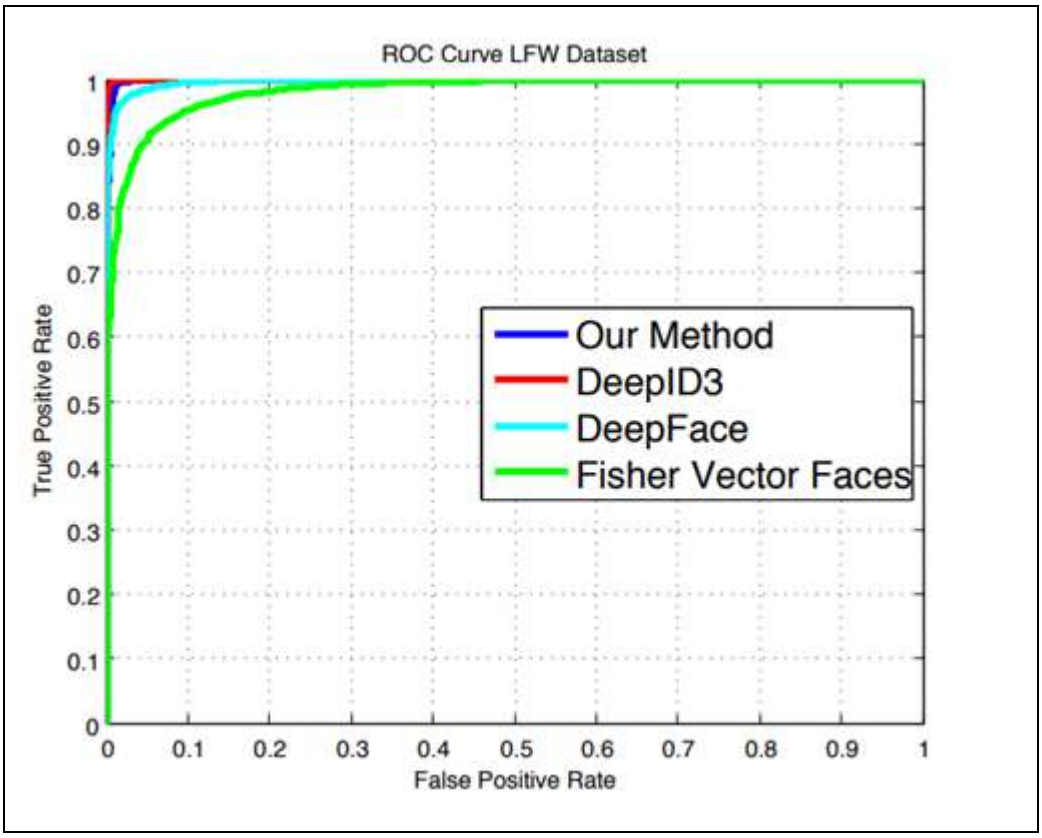


Figure IV.9. *LFW ROC curve comparison[76]*

Table IV.5. Shows the performance on the YTF dataset. VGGface achieves the state of the art performance using a triplet loss embedding method.

4.2.3 Similarity distance threshold fine-tuning

We've represented input images as vectors. And the output from VGGface descriptor is a long vector representing facial characteristics. We will decide both pictures are same person or not based on comparing these vector representations. Now, we need to find the distance between these vectors. There are two common ways to calculate the distance between two vectors: cosine distance and Euclidean distance.

Cosine similarity looks at the angle between two vectors, Euclidian similarity at the distance between two points. There is an exponential increase in volume associated with adding extra dimensions to a mathematical space leading to accuracy loss associated with comparing higher dimension vectors. Below are the equations to calculate cosine similarity and Euclidean similarity.

$$CosineSimilarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i} \times \sqrt{\sum_{i=1}^{n} B_i}} \quad \text{(IV.1.)}$$

$$EuclideanDistance(A,B) = \sqrt{\sum_{i=1}^{n} (B_i - A_i)^2} \quad \text{(IV.2.)}$$

For comparing distances we can use VGGface's last layer which yields a feature vector of 2622-d relating to the 2622 identities it had been trained on, alternatively we can use the face descriptor available in the layer before last which is a 4096-d vector describing the face in a more generalized way.

Before we start fine-tuning the distance threshold we must first compile a list of tests, a set of pairs that correspond to the same identity, and a set of pairs that correspond to different identities, after we run all tests we plot the histograms of both negative pairs and positive pairs to further analyze how good the model is separating the two and finally finetune the distance threshold to its optimal value.

The test set we used was provided by LFW research team, they offer both positive and negative sets, each contain 1100 pairs of faces.

4.2.3.1 Fine-tuning using Euclidean distance

In these tests we will try using both the 2622-d descriptor and the 4096-d descriptor.

While comparing faces we pushed all results to an array to later make a histogram out of it. (Figure IV.10.)

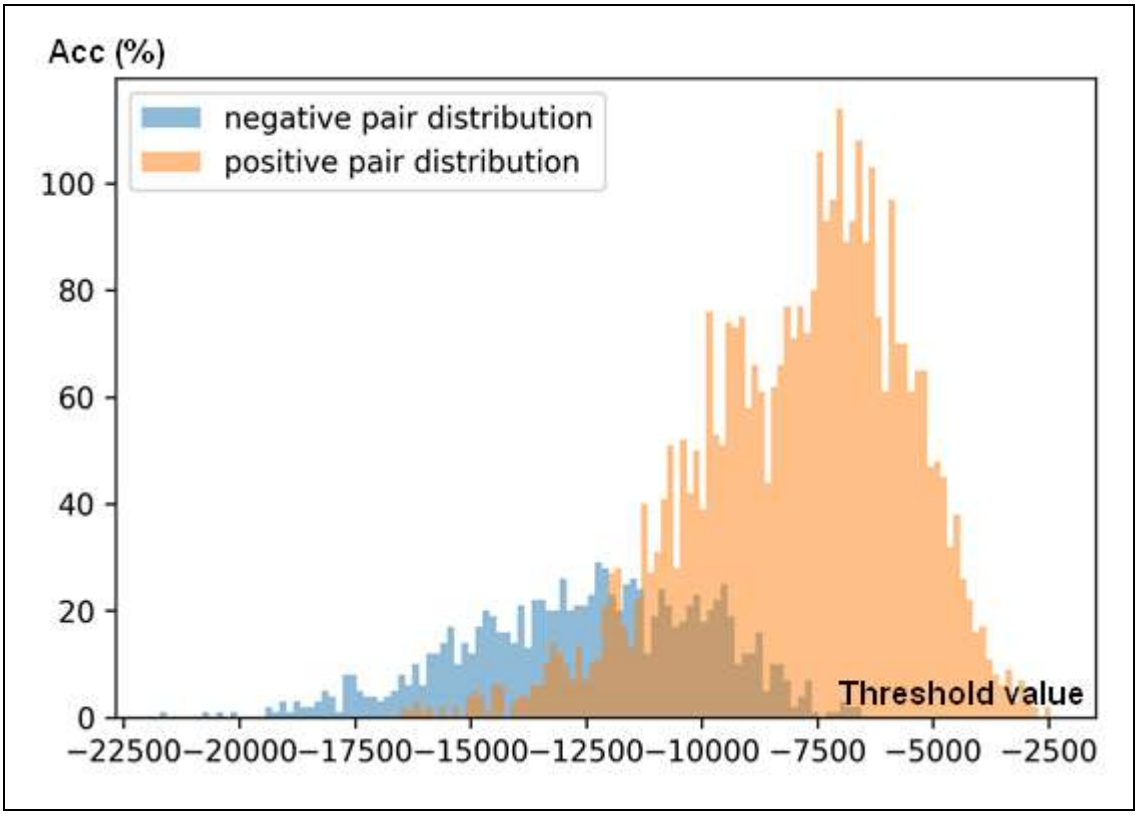


Figure IV.10. *Histogram of threshold distribution for positive/negative pairs using Euclidean distance and the 2622-d feature vector*

From the histogram above we notice that the distribution of negative and positive pairs intersect in a large area denoting a high false positive rate.

To choose the optimal threshold (value to separate between matching and non-matching identities) we need to minimize the false positive rate of both negative and positive pairs, to do this we simply pinpoint the highest accuracy rate between the two sets of test as seen in figure IV.11.

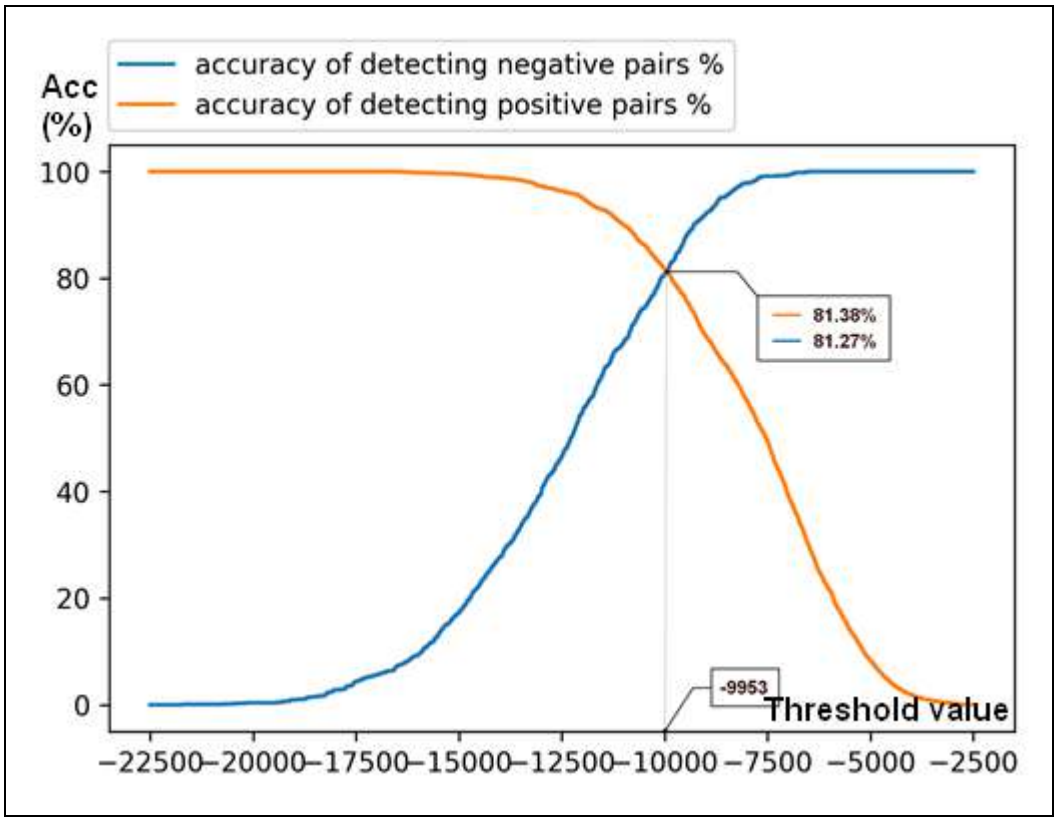


Figure IV.11. *Threshold accuracy in detecting negative and positive pairs, for Euclidean distance and 2622-d feature vector*

The results above tell us that the optimal threshold value is at -9953 which corresponds roughly to 81.32% accuracy (81.38% accurate in finding matching faces, 81.27% accurate in finding non-matching faces).

Next we will re-apply the same tests this time discarding the last layer from the model, meaning we will use the 4096-d descriptor instead.

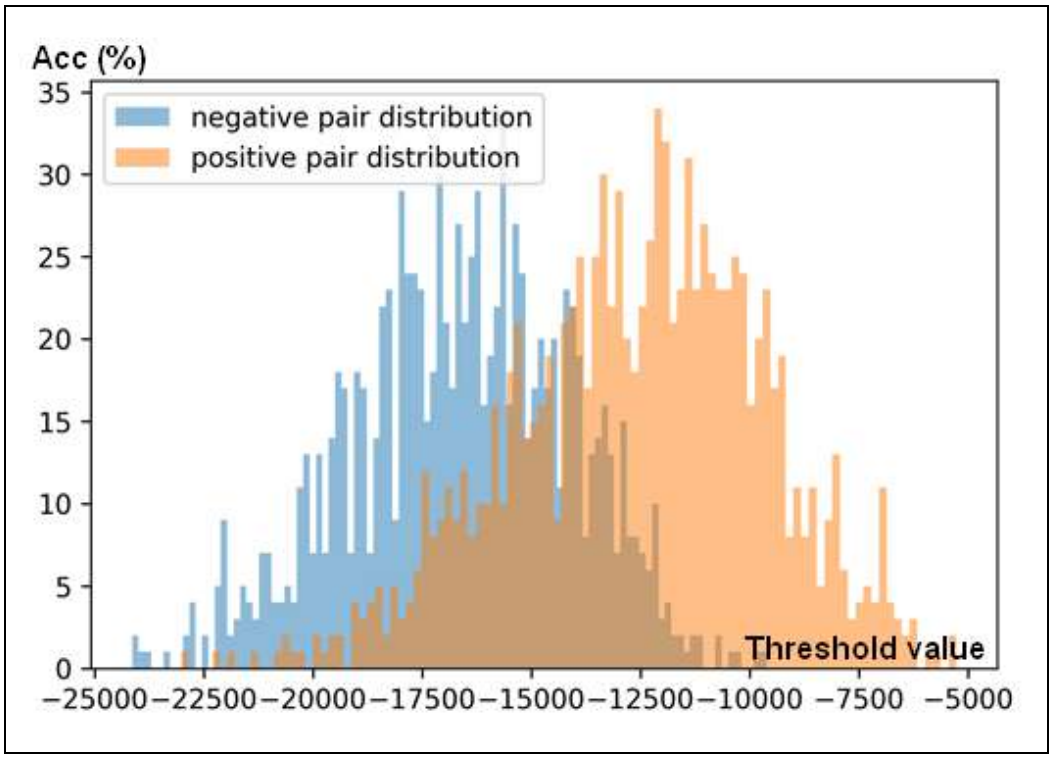


Figure IV.12. *Histogram of threshold distribution for positive/negative pairs using Euclidean distance and the 4096-d feature vector*

The histogram results from figure IV.12 show similar results to the first try using the 2622-d, that is the intersection area is too big.

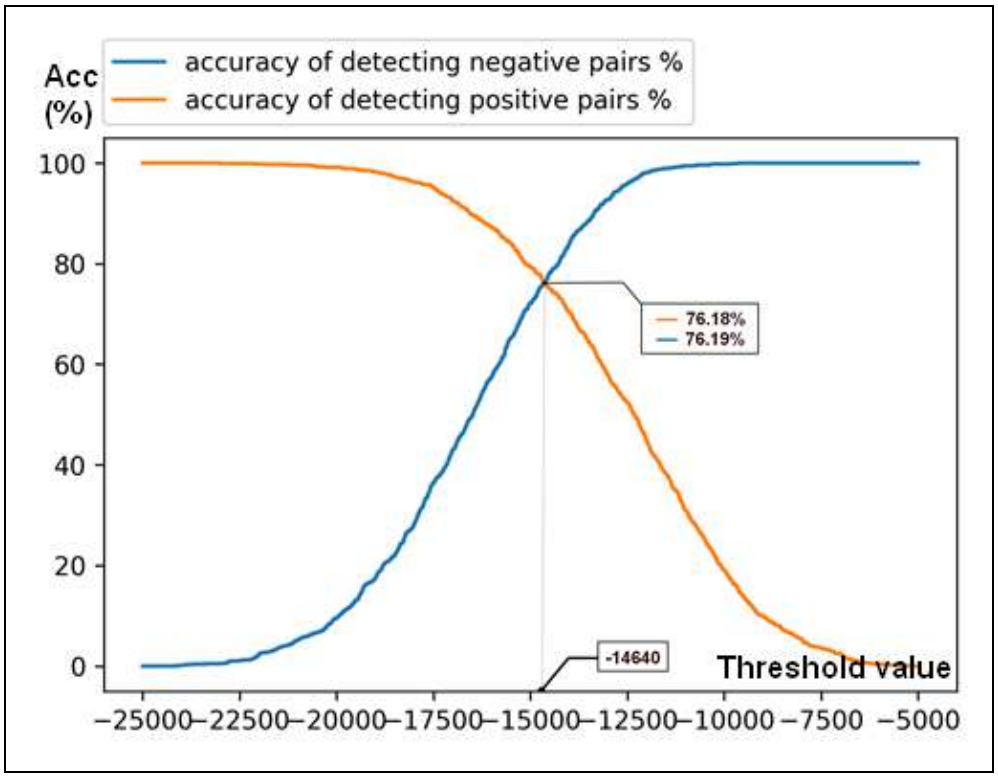


Figure IV.13. *Threshold accuracy in detecting negative and positive pairs, for Euclidean distance and 4096-d feature vector*

Figure IV.13 shows that the optimal threshold value is at -14640 which corresponds roughly to 76.18% accuracy (76.18% accurate in finding matching faces, 76.19% accurate in finding non-matching faces).

We see a big decline in accuracy when switching from the 2622-d descriptor to 4096-d, the most likely reason is that euclidean distance struggles to find similarities between vectors with higher dimensions since it's based on distance between points (as mentioned earlier, the higher the dimension the more volume it has, thus it loses the ability to catch small changes between vectors).

4.2.3.2 Fine-tuning using Cosine distance

The same tests will be re-applied in this experiment, first we will try to measure the optimal accuracy we can get from the last layer descriptor (2622-d), then we try to use the descriptor in the layer before it to see if any improvements happen.

First we calculate the distances needed to make the histograms using the 2622-d descriptor (Figure IV.14.).

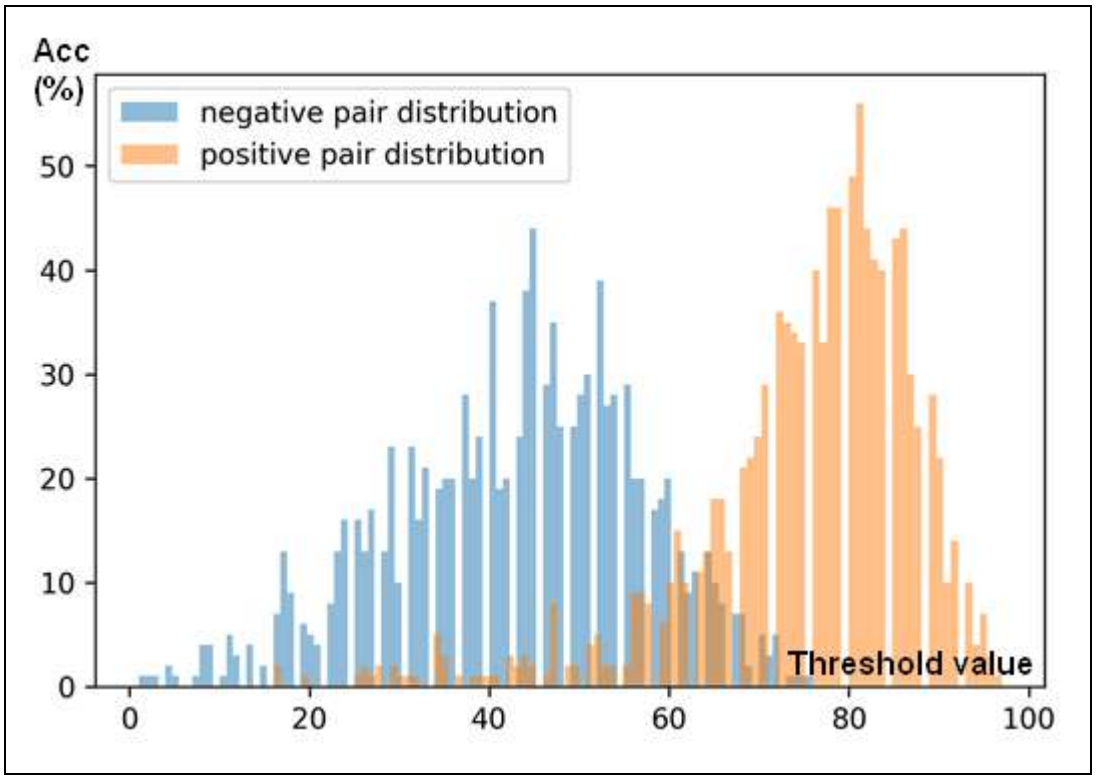


Figure IV.14. *Histogram of threshold distribution for positive/negative pairs using Cosine distance and the 2622-d feature vector*

The first thing we notice from the histogram above (Figure IV.14.), is that the negative and positive set distributions are separated much better compared to the method earlier used (Figure IV.10.), which is an indication that the accuracy will most likely increase.

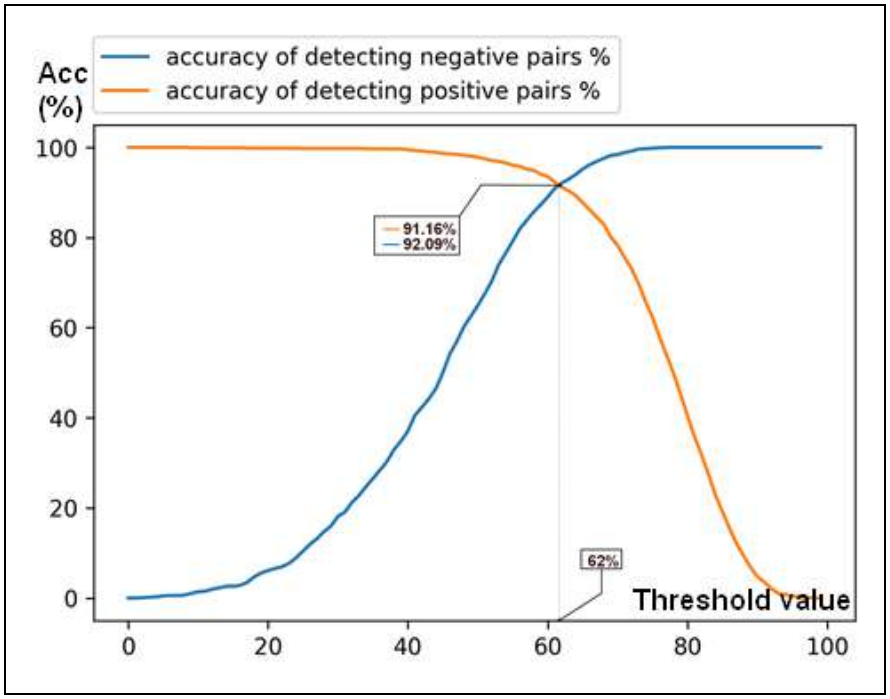


Figure IV.15. *Threshold accuracy in detecting negative and positive pairs, for Cosine distance and 2622-d feature vector*

The figure (Figure IV.15.) shows that the optimal threshold value is at 62% which averages to 91.625% accuracy rate split between 91.16% accuracy in finding matching faces, and 92.09% accuracy in finding non-matching faces.

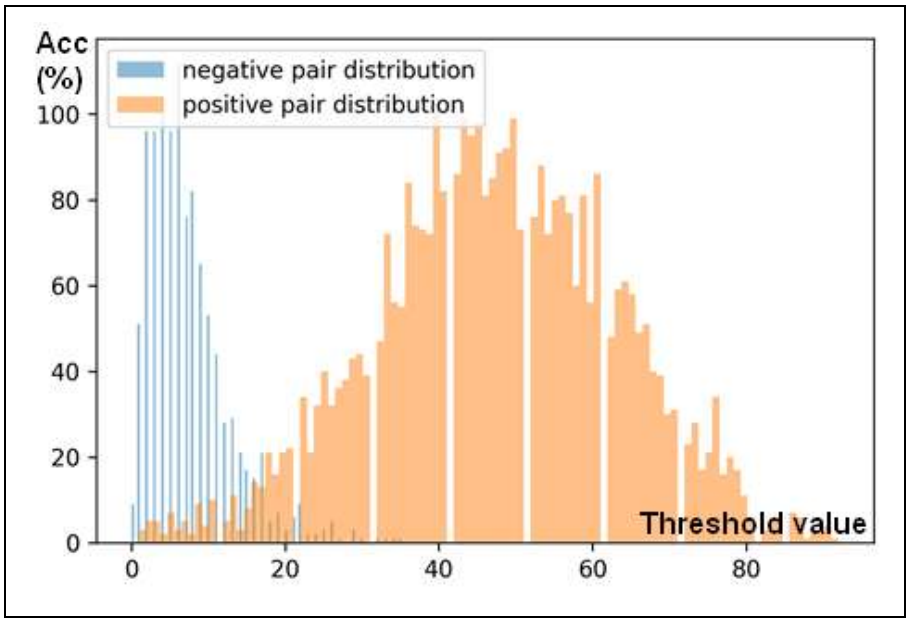


Figure IV.16. *Histogram of threshold distribution for positive/negative pairs using Cosine distance and the 4096-d feature vector*

Next we retry using the 4096-d face descriptor and see if any changes occur.

The distribution histogram (Figure IV.16.) shows that the model has a strong ability to clump non-matching faces together away from the positive distribution, further separating the two which may indicate even more accuracy.

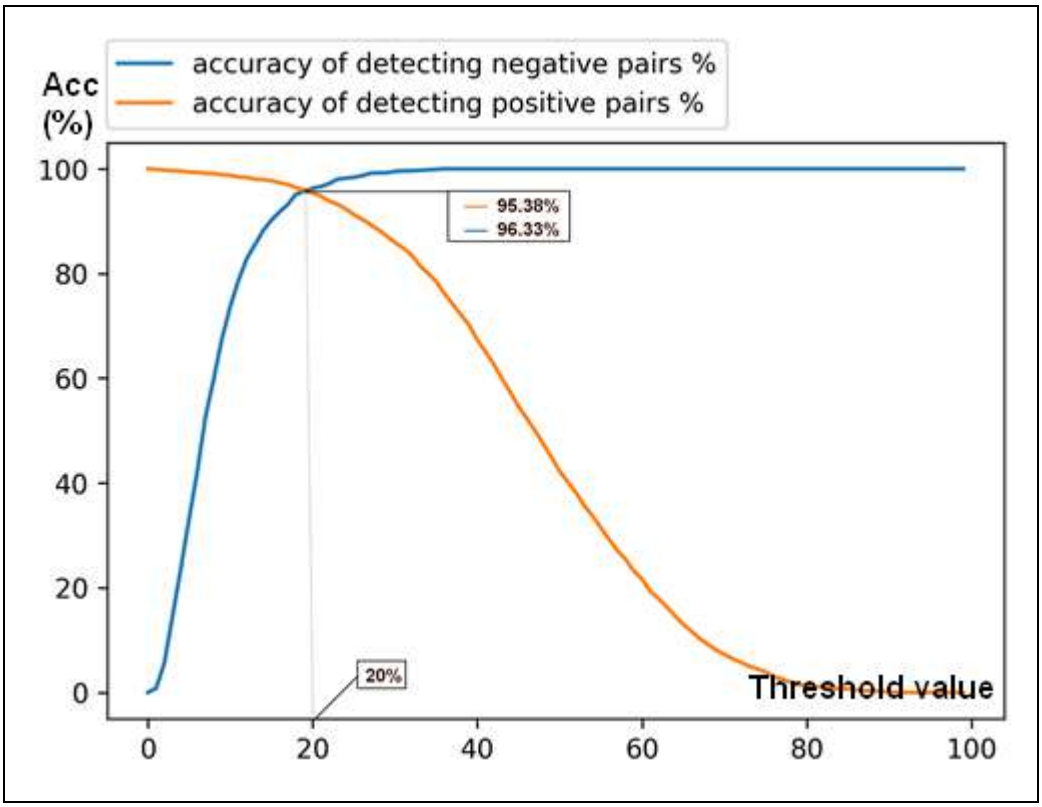


Figure IV.17. *Threshold accuracy in detecting negative and positive pairs, for Cosine distance and 4096-d feature vector*

Figure IV.17. Shows that the optimal threshold value is at 20, giving us an accuracy rate of 95.855%, where the accuracy rate of detecting matching faces reached 95.38% and the accuracy rate of detecting non-matching faces (negative pairs) reached 96.33%.

The reason the accuracy rate improved when we used the feature vector with more dimensions is the fact that cosine similarity only looks at angles and not distance, meaning that all those extra dimensions are actually holding more valuable information.

4.3 Discussion

In this section we have tried multiple methods to try to pinpoint which strategy is best when recognizing faces, our first attempt showed great potential from the validation accuracy and F1 score, unfortunately it ended up unusable the moment we introduced identities alien to our database. With a false positive rate of 100% the network is completely unusable for authentication purposes, it may however be used as consumer app to see your closest match

from a pool of famous people or something similar, since it does actually try to map you to the closest face.

On the other hand our second attempt was a complet success, it showed spectacular results when differentiating between faces on a state-of-the-art level, from the table below we see a summary of configurations used as well as the test results concerning the accuracy and the optimal threshold value (the value used in the condition that determines if a pair of faces match). The winning candidate is clear, we use the 4096-d vector from VGGface with the combination of Cosine similarity function which gave us approximately 96% accuracy.

Table IV.6 is a summary of the test results we did to fine-tune the distance threshold.

Table IV.6. *Comparison between the results of different distance algorithms*

		Euclidean Distance		Cosine Distance	
		Accuracy rate	Optimal threshold	Accuracy rate	Optimal threshold
Last Layer Description vector 2622-d	Negative Pairs	81.27%	-9953	92.09%	62
	Positive Pairs	81.38%		91.16%	
Before Last Layer Description vector 4096-d	Negative Pairs	76.19%	-14640	96.33%	20
	Positive Pairs	76.18%		95.38%	

5. Implementation

We are set to start implementing our authentication system after pinpointing the best method to recognize faces, First we built the basic face detection and recognition functionality (using openCV haar-cascade filters to detect faces, then using the VGGface model to extract the features vector), then we encapsulated this functionality in a Graphical User Interface application using the Tkinter library from python, finally we linked our GUI application with an SQLite database to store all access attempts, in parallel to this we've built a website to allow the administrator of our application to manage access reports as well as add users to the enrolled database.

5.1 Graphical User Interface

Designing the graphical interface is very simple and intuitive, the Tkinter library provides a higher level abstraction layer to create elements with no boiler-plate code, meaning everything was declared in immediate mode.

For our application we need 5 main areas, shown in Figure IV.18. we have (1) to show the video stream coming from the webcam, (2) is a placeholder to show authentication status when a user attempts to authenticate, (3) is a button that allows the user to authenticate, (4) is a button that allows the user to re-calculate features vectors from the enrolled users database incase of adding a new user to the database, finally (5) a button that allow us to exit the program.

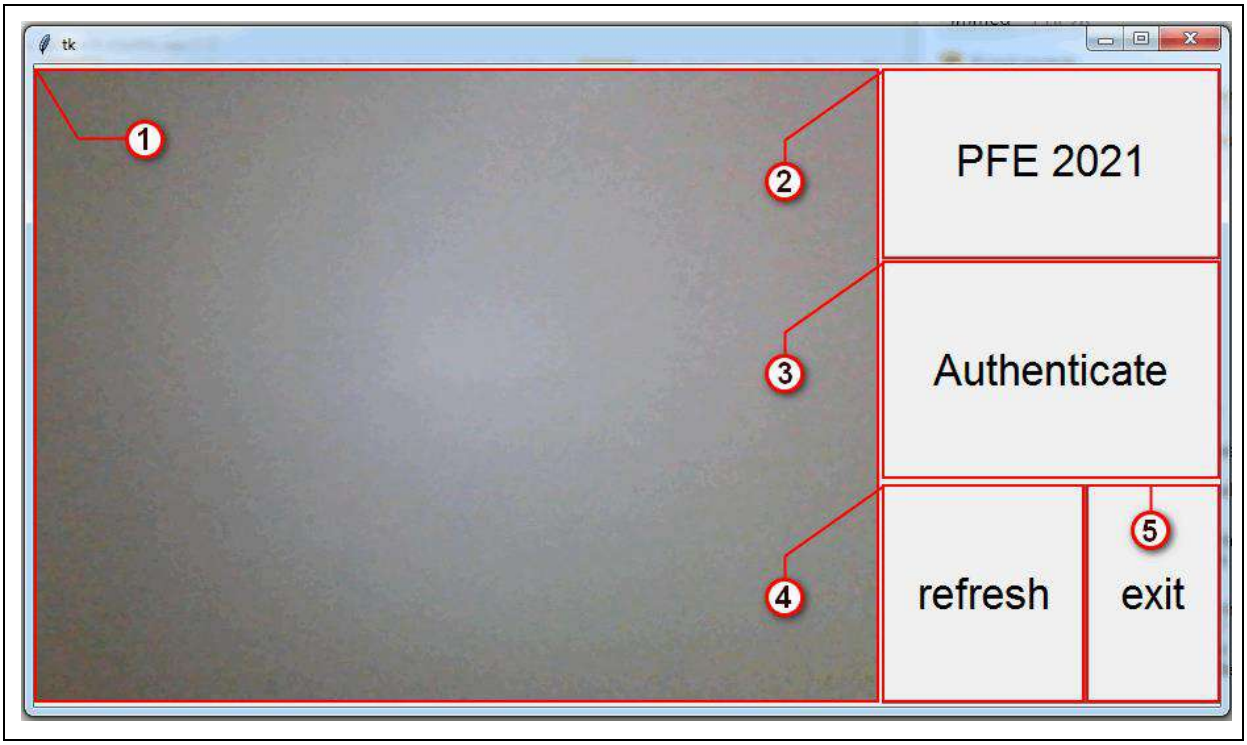


Figure IV.18. *Graphical user interface of our authentication system*

The diagram below (Figure IV.19.) corresponds to the general flow of action/information in our graphical interface program.

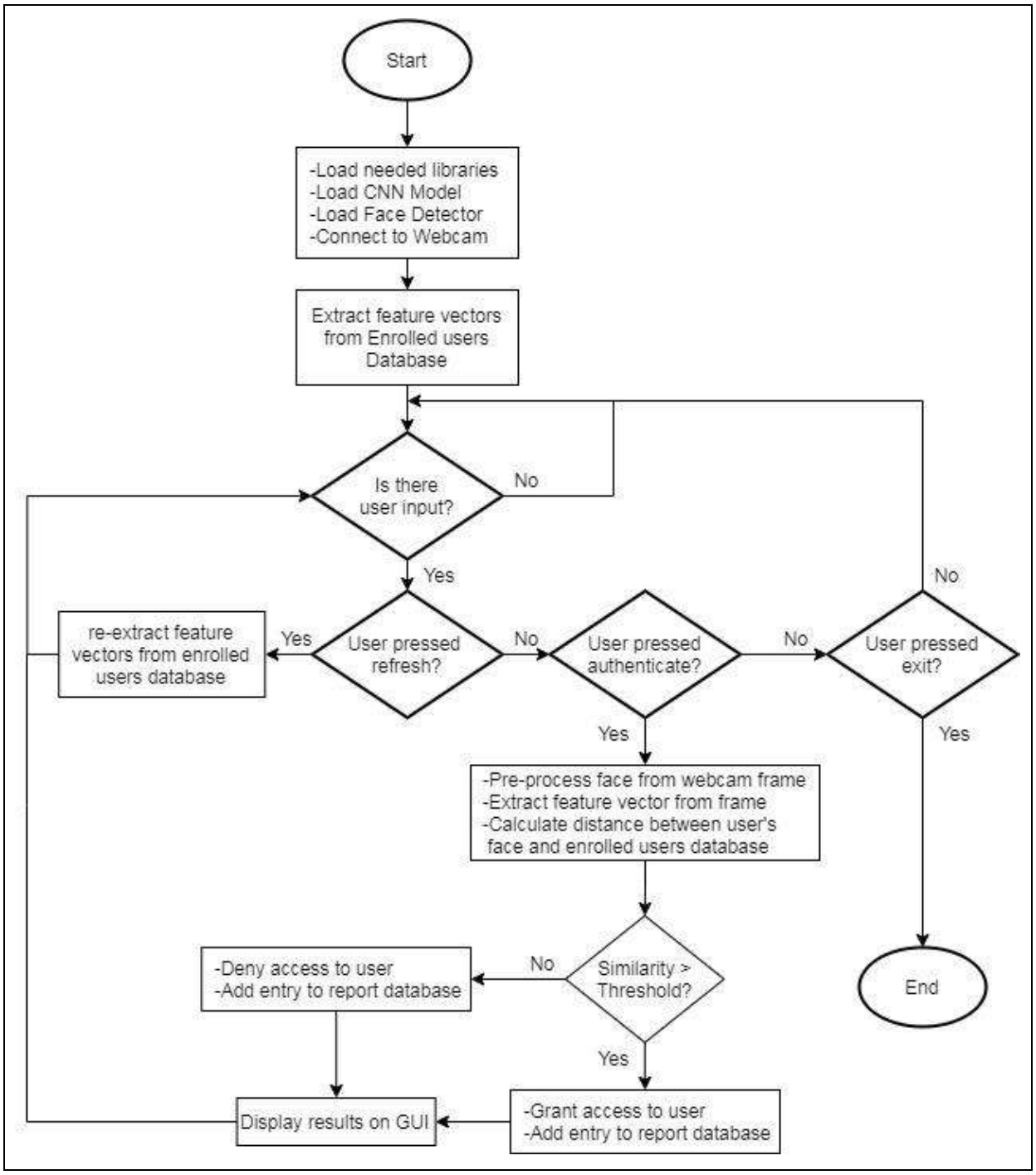


Figure IV.19. *Flowchart of the main functionality of the GUI application*

The flow in our program fairly straightforward, first the program initiates a blank VGGface model and load the pre-trained weights to it, next it loads the haar-cascade filter provided by OpenCV (used to detect faces) then creates a handle to the webcam (to start receiving the videostream), after everything is initialized the app will import all faces from the enrolled users database, extract features vectors from each face and store everything in a dictionary.

Next the program enters the GUI loop which scans for changes or actions in each tick, there are three actions available for the user to interact with:

- Refresh button is pressed: this button will try to import all faces from the enrolled users database and re-calculate their features vectors, this button is useful in case the admin of our system decides to add a new user to the database, the only way our GUI program know about them is by re-calculating features vectors of our enrolled users.
- Exit button is pressed: this button will simply send a termination signal to our program's process, so it exits from the GUI loop then exits the program itself.
- Authenticate button is pressed: this button will execute the main face detection and recognition function by providing a frame from the webcam, this function will run the necessary processes to figure out if the user's face match any identity from the enrolled users database. If the result is positive, the door is unlocked, otherwise the door is kept locked, meanwhile the result will be shown on our GUI, it will also be stored in the access reports database.

5.2 Face detection and recognition

Now that we've created a graphical user interface shell for the functionality we require, the only thing left to do is integrate the face detection and recognition function; Figure IV.20. depicts the major inner workings of this function.

The face detection and recognition function will accomplish the following points:

- Detect faces from webcam
- Filter out low resolution faces
- Calculate features vector of user's face using VGGface
- Compare user's features vector with enrolled users features vector and pinpoint to which it belongs (given the similarity is above the threshold)
- Compile an access attempt report and push it to the access reports database
- Grant or deny access permission to user
- Draw bounding box over the user's face with its name in it.
- Return authentication status, and modified image to be displayed.

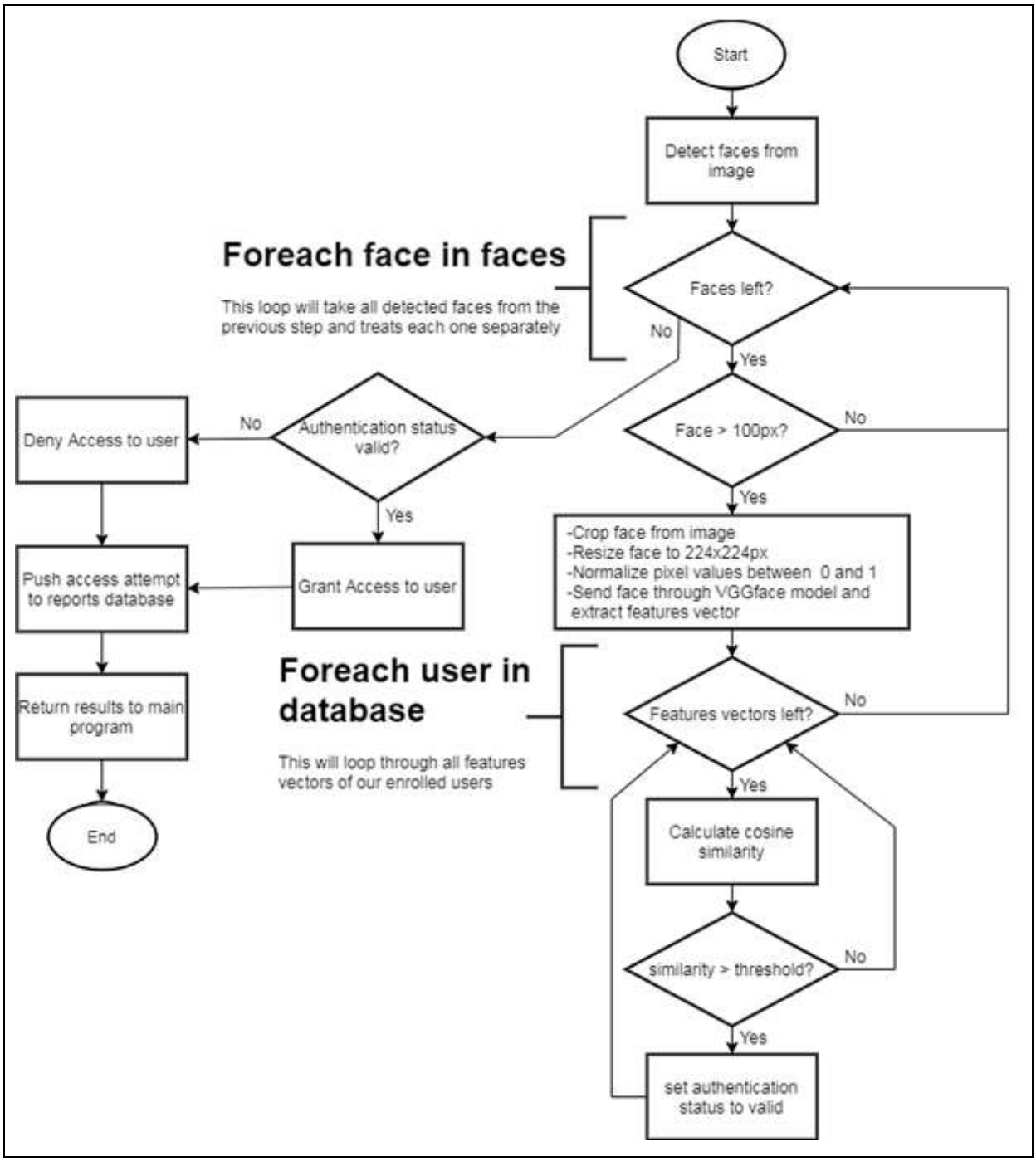


Figure IV.20. *Detailed flowchart of the inner workings of the recognition function*

5.3 Website

After finishing the main program that will be used to authenticate people, we still need a way to add people to the database, as well as a place to check all access attempts, for that we proposed a simple website to be used by the administrator of the system, it will be password protected, it will have a table view of all access attempts, and it will allow the admin of our system to add users to the enrolled users database with ease.

5.3.1 Login page

Since the website will be visible to anyone connected to the local network, we need to introduce a login page (Figure IV.21.) to restrict who can view and add users to the system only to the administrator.

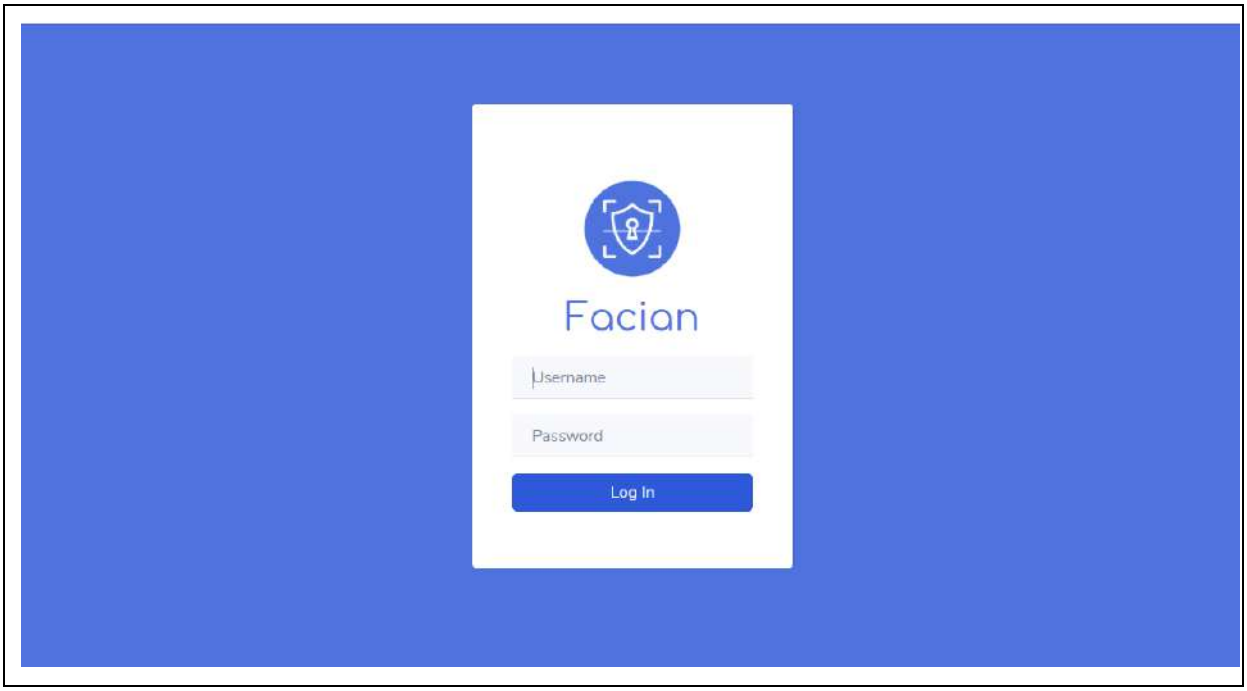


Figure IV.21. *Login page of the website*

5.3.2 Access reports page

The roll of this page (Figure IV.22.) is presenting a detailed log in tabular form of all

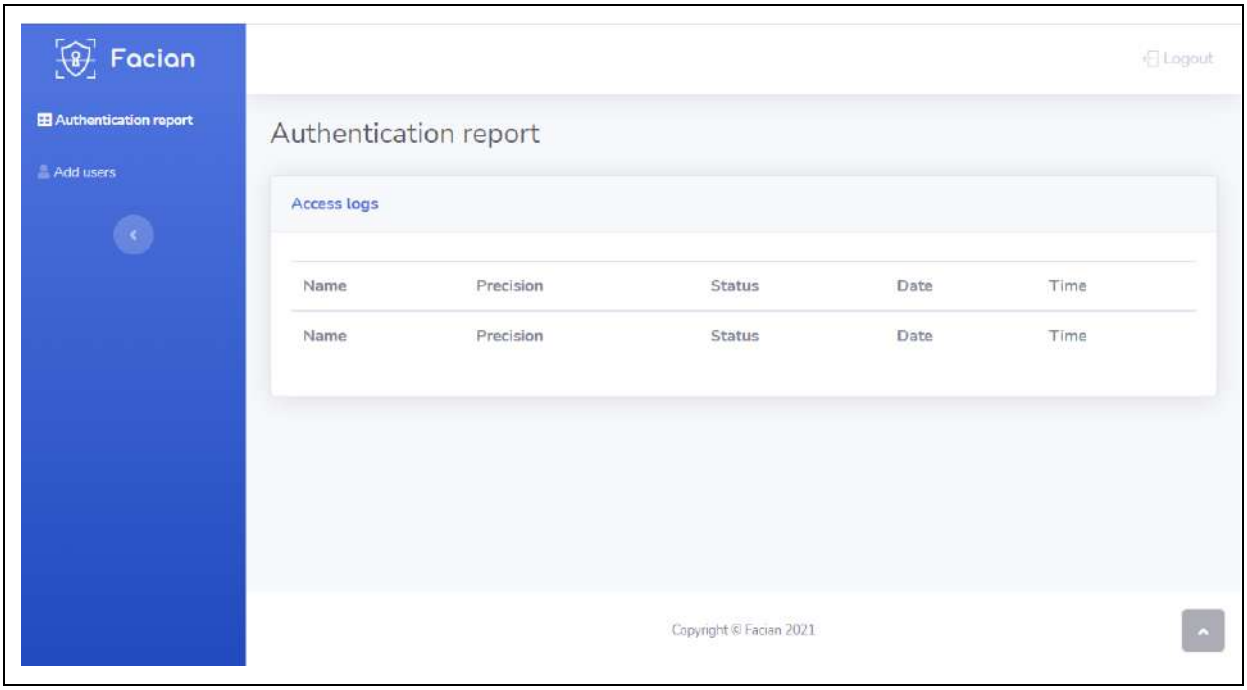


Figure IV.22. *Authentication report page of the website*

access attempts on our authentication system. To get such info we utilize the reports database mentioned in Figure IV.20. which will have the picture of whoever is authenticating as well as their name (if available), the accuracy of prediction, authentication status, and the time and date at which they attempted to enter.

5.3.3 Add users page

To quickly add users to the enrolled users database we developed the page shown in Figure IV.23.

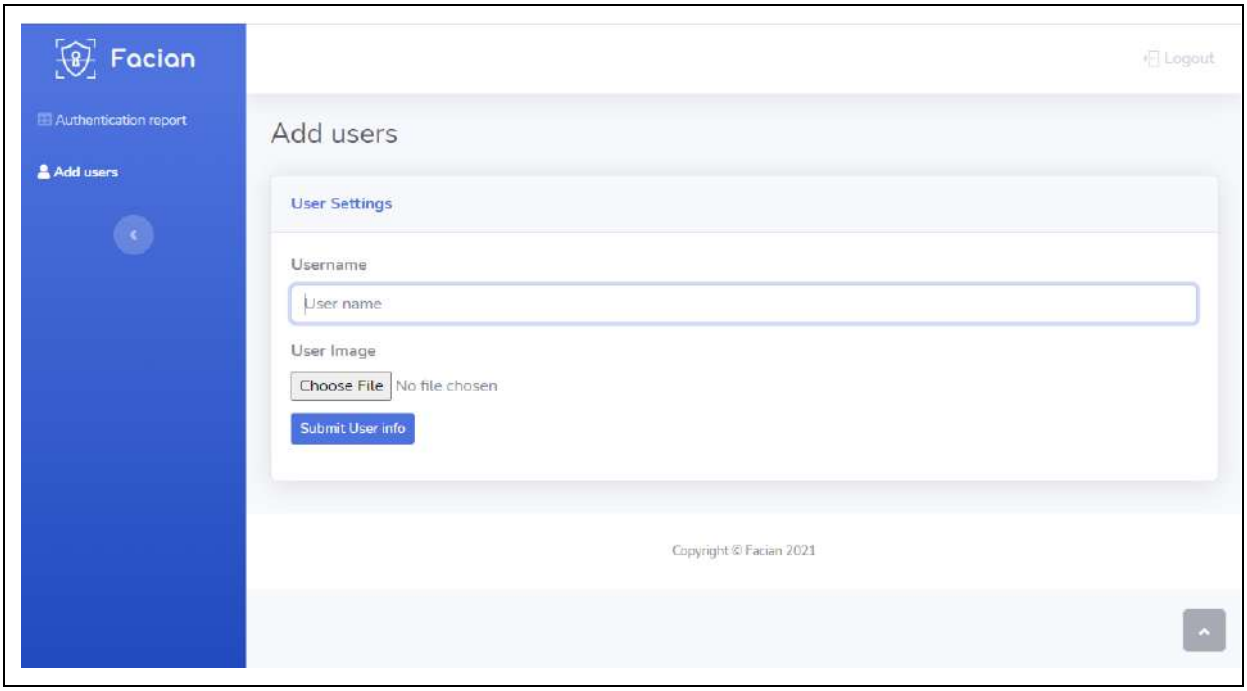


Figure IV.23. *User management page of the website*

This way the administrator can easily add users, simply by stating their name and attaching a picture showing their face.

In case of submiting a new user, their picture is sent to a python script that will detect and crop only their face, it will then store the user's cropped face coupled with their name on the enrolled users database then return the operation result to be displayed on the website. In case of any error (such as failing to find a face in the picture or failure to add a new entry to the database) the website will then prompt the admin that a problem occurred and the operation is cancelled.

5.4 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 ran our graphical interface as well as the webserver on a raspberry pi 4, and attached it to a generic raspberry pi cam for the live feed.

Finaly, to control the electric lock a relay was used (triggered by the raspberry pi to allow energy flow from the power supply (12V) to the electric lock).

5.4.1 Electrical wiring

Figure IV.24 shows the complete wiring diagram of our system:

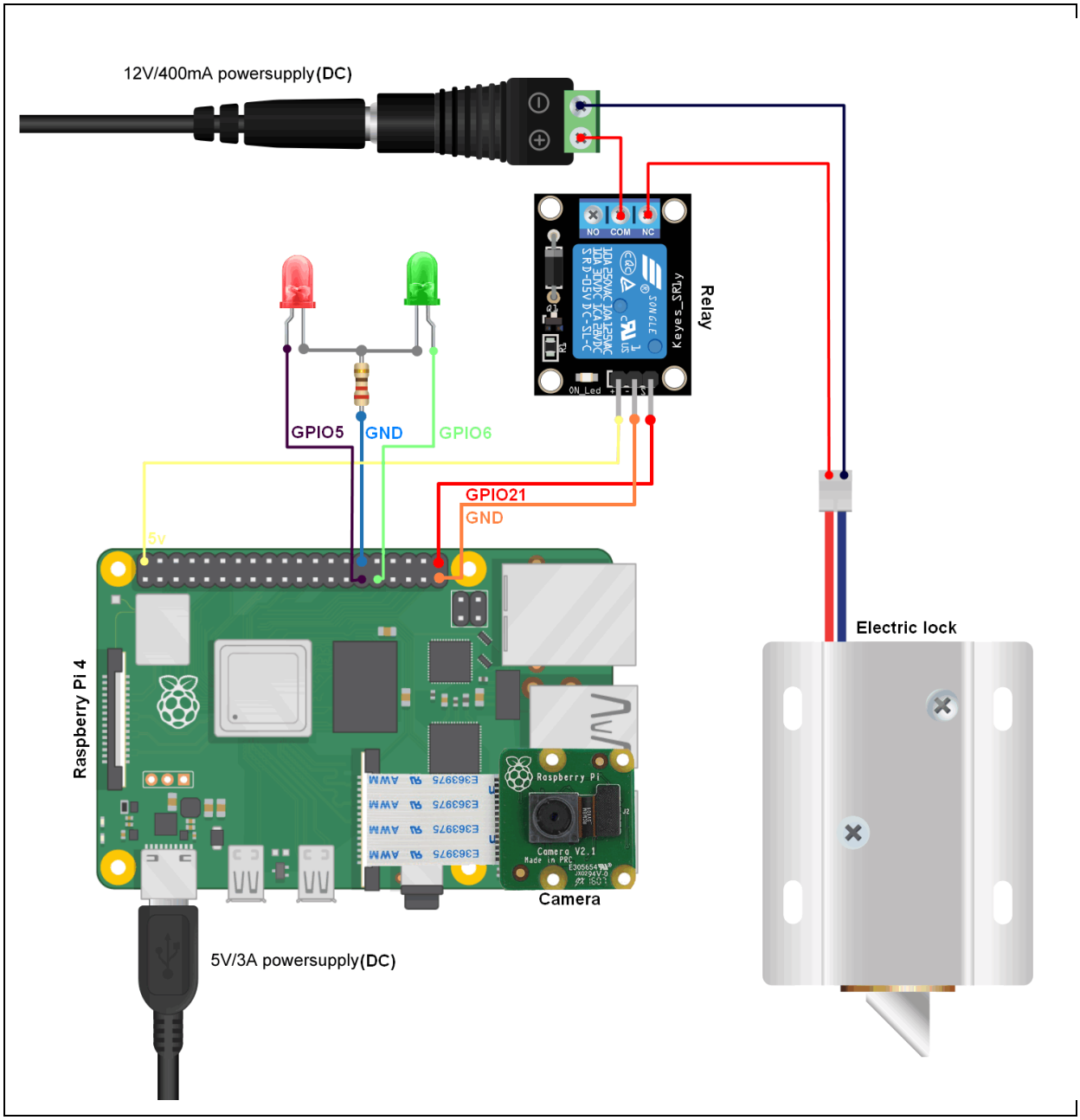


Figure IV.24. *Electrical wiring diagram of our system*

The GPIO pins of the Raspberry Pi can give an output of 3.3V, but the electric lock requires 12V to operate. For this reason, we used an external power source and a relay to actuate the lock.

For the electric lock part we first connected the VCC and GND of the relay module to 5V and GND of Raspberry Pi. Then connect the signal pin of the relay module to the GPIO 21 of Raspberry Pi. On the other side of the relay module, we connected the negative of the DC power supply to the negative of the electric lock. Then we connected the positive of the power supply to the common of the relay module, finally we connect the normally closed of the relay module to the positive of the electric lock.

For the door indicators we connected the anode of the green led to GPIO 6, and the anode of the red led to GPIO 5, we then connected both of their cathodes to GND through a common resistor.

Finally we used the default 15-pin connector to connect the camera module.

5.4.2 Final product assembly

To accomplish this, we used the materials discussed earlier in Chapter 3 and then followed the instructions stated in the sub-subsection above (5.4.1). (Figure IV.25.) depicts the entire assembly of our product.

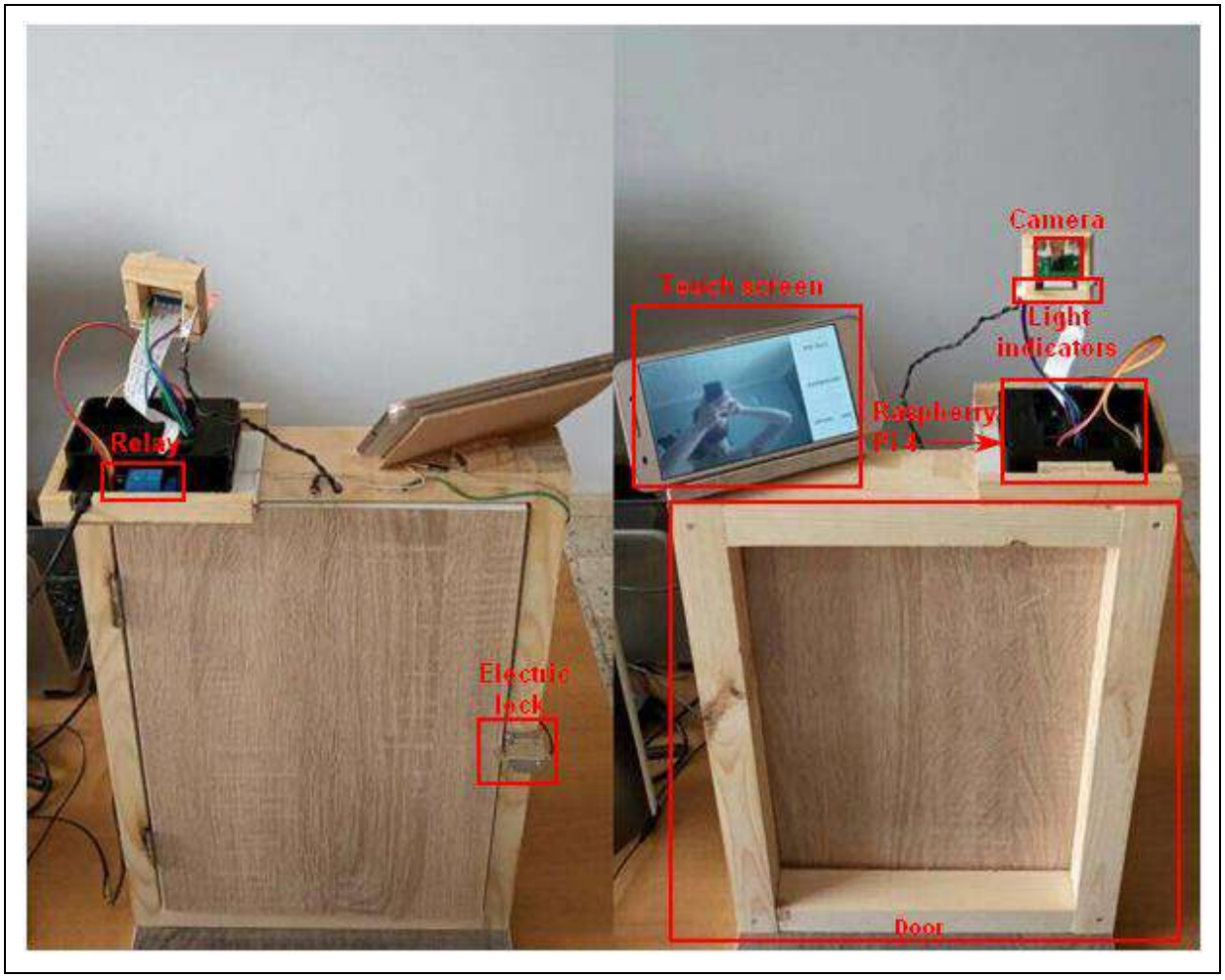


Figure IV.25. *Assembled authentication system*

6. Results

Now that the product is finalized we can showcase it, first we tried running it before adding the user's face to the enrolled database, then added a the user to the database and test it again, finally we checked the results on our website:

- First case: User is not present in the enrolled users database, the system will reject him (unknown) as shown in Figure IV.26.



Figure IV.26. *Authentication test before adding the user*

- We add the user to the enrolled users database through the system's website (Figure IV.27).

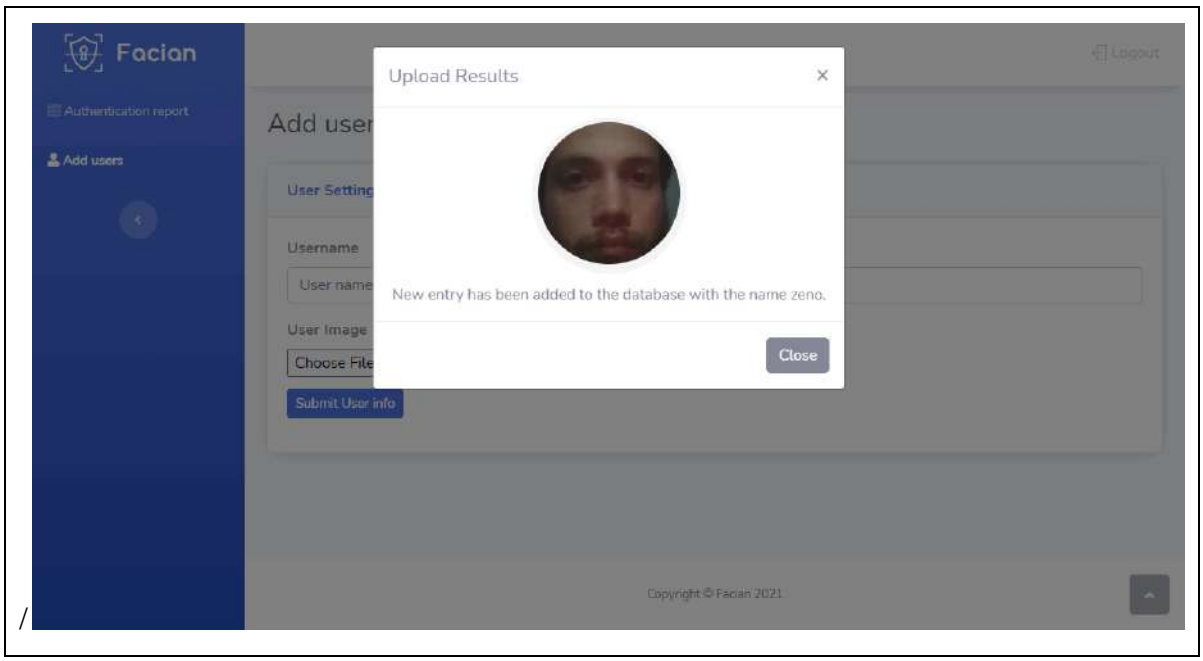


Figure IV.27. *User successfully added to enrolled users database*

- Second case: User is now present in the enrolled users database, the system will grant him access permissions as shown in figure IV.28. and unlock the door.

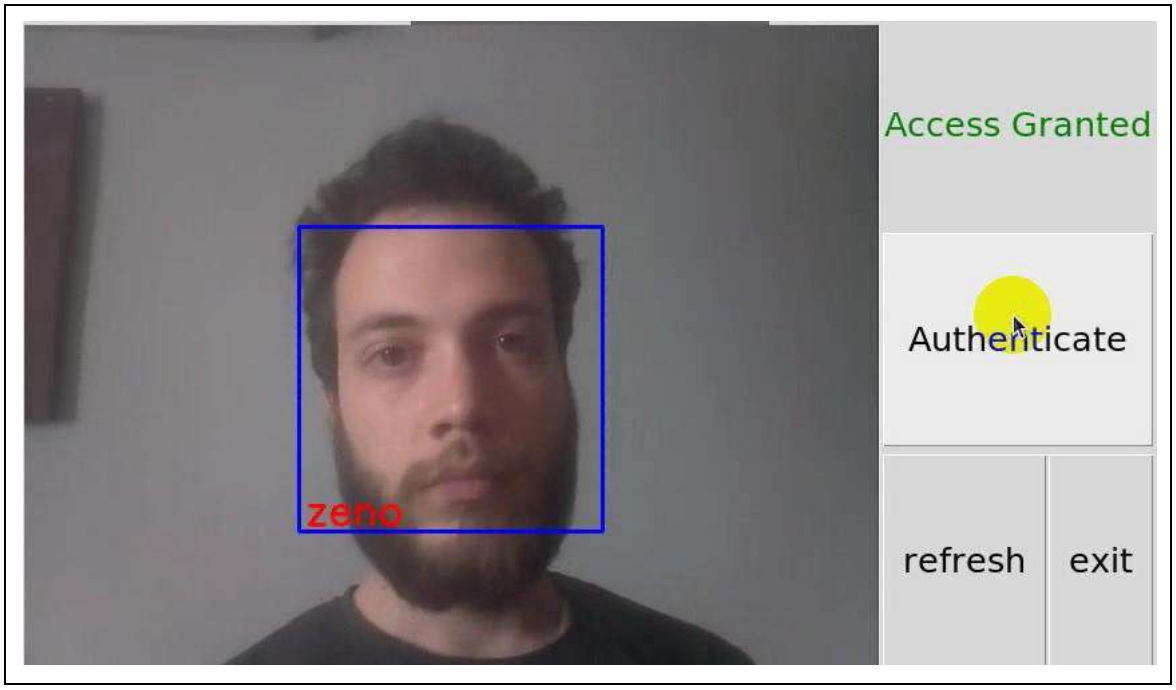


Figure IV.28. *Access Granted to user that is present in enrolled users database*

- Third case: Same as the second case with the addition of facial accessories (to see if the system recognise occluded faces), the system will grant him access permissions as shown in figure IV.29. and unlock the door.

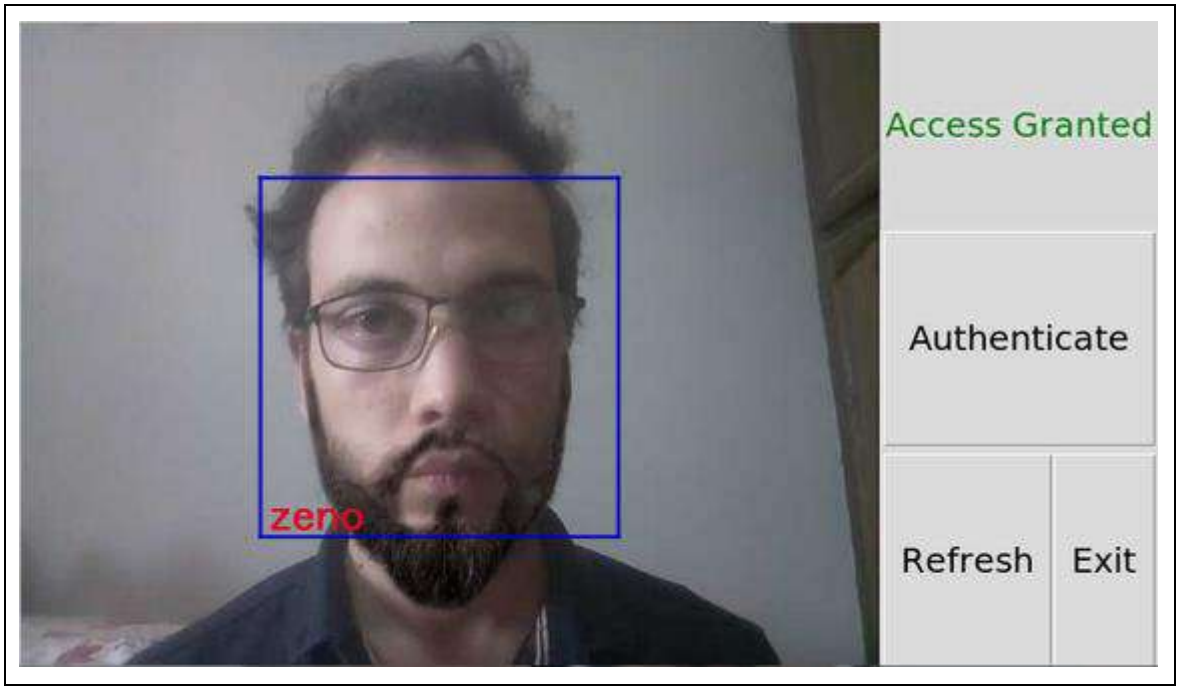


Figure IV.29. *Access Granted to user with artificial beard and glasses that is present in enrolled users database*

If we check the Authentication report page on our website (Figure IV.30.) we notice that the system has logged that someone unknown tried to authenticate twice but failed, after we added that person and assigned a name to him, then we tried to authenticate again, we see from the logs that the system did indeed recognize the user and granted him access:

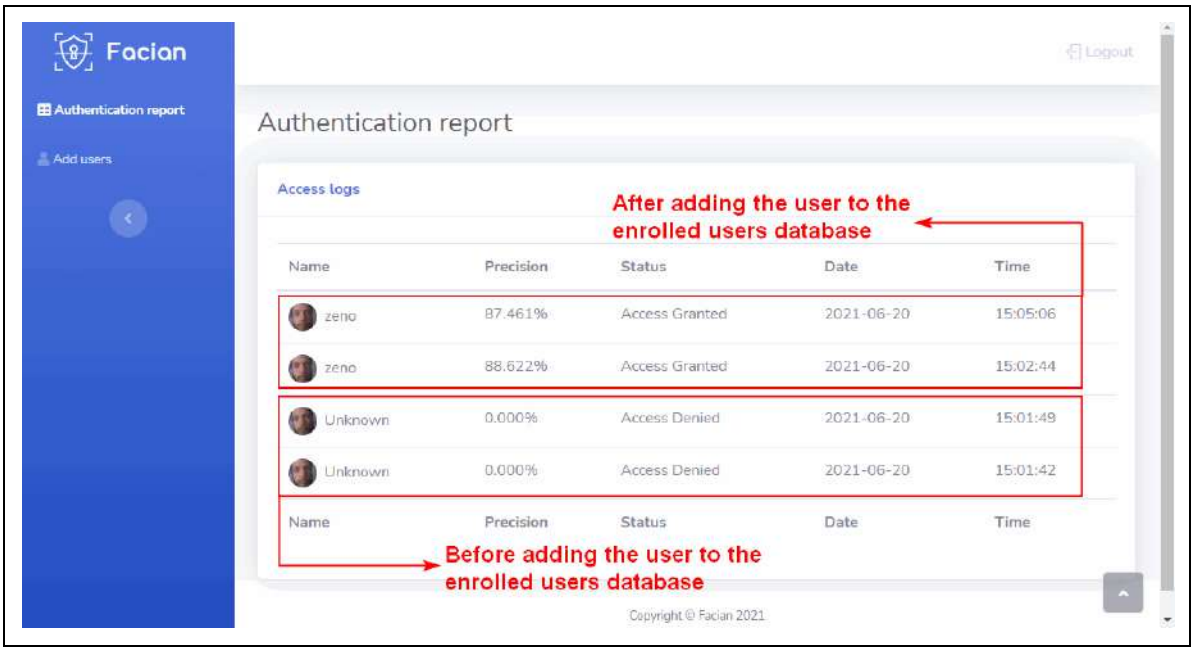


Figure IV.30. *Authentication report populated by access attempts*

7. Conclusion

We presented different approaches to face recognition using convolution neural networks in this chapter; in the first approach, we used CNNs to categorize faces; in the second, we tried using a larger model to describe faces rather than categorizing them into identities; and we also presented the results obtained for each approach. We then integrated the solution into our product, allowing users to authenticate themselves using a graphical interface deployed near the access point, as well as providing a website where users can easily log in and add new users. Overall, the system demonstrated promising results and high accuracy in face recognition.

GENERALE CONCLUSION

In this thesis, we made a full solution for access control based on face recognition specifically by using convolutional neural networks. We compared different approaches to face recognition. First being a CNN with N classes (N being the number enrolled users) that was trained on a set of faces that belong to the identities allowed to enter; but after testing it, we found a major flaw, although the validation accuracy was very high (around 95%) the network failed to filter out faces that didn't belong to those N classes making the approach a complete disaster for our purpose but not necessarily for other purposes (such as making fun consumer apps with it). In the second approach, we used a VGGface network to extract face feature vector then calculate the closest distance between it and the feature vectors of our enrolled users, if the degree of similarity was above a certain threshold, then it means the faces we are comparing belong to the same identity, thus solving the recognition problem. After multiple tests, we fine-tuned the threshold to attain an accuracy rate of ~95%.

After pinpointing the best method to do face recognition we moved to integrate it with a graphical interface that was made with Tkinter (in python), which allowed our users to interact with a touch screen to request access, the system used a raspberry pi 4 as a base processing unit which in turn was connected to a webcam and the electric door lock, if a user authenticates successfully the system will unlock the door for them otherwise it will keep it closed, in either way it will record all access attempts and store them in a database. To add new users to our system or check the recorded access logs we made a website that is only accessible by the admin of the system.

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, PHP, SQLite, Tkinter, OpenCV, Keras, and so on.

For future work we would like to develop this product even further by making it modular and extensible, by making all doors in a facility report to a central unit by utilizing IPTVs and IoT modules to wirelessly control all access points from a central point, we would like to also add multiple clearance levels to our users to better control who and where our users can and can’t go. Finally, we would also add second-factor authentication (via RFID) to enhance the security level even further.

BIBLIOGRAPHY

[1] " What is Biometrics – Definition, History, Future of Biometrics Technology". [Online]. At: https://www.earlyintime.com/2018/10/18/what-is-biometrics-definition-history-future-of-biometrics-technology. [Visited on 29 mai 2021].

[2] " NEW PAYMENT METHODS: Is Canada Ready?" Sophie Roussin . [Online]. At: https://uniondesconsommateurs.ca/wp-content/uploads/2020/12/35-Modes-paiement-f-e2.pdf. [Visited on 29 mai 2021].

[3] " Improvement of the Fingerprint Recognition Process | Zenodo" *Farah Dhib Tatar* . [Online]. At: https://zenodo.org/record/1289252#.YLGCzZOxXSF. [Visited on 29 mai 2021].

[4] John. D, Woodward. Jr, Christopher Horn, Julius Gatune, and Aryn Thomas,«Biometrics A Look at Facial Recognition», RAND Corporation , 2003

[5] " Nine distances selected for analysis: 1 face distance (Left), 3 eye... | Download Scientific Diagram" *Journal of Autism and Developmental Disorders* . [Online]. At: https://www.researchgate.net/figure/Nine-distances-selected-for-analysis-1-face-distance-Left-3-eye-distances-2-nose_fig2_328810887. [Visited on 29 mai 2021].

[6] "Reconnaissance faciale" *Reconnaissance faciale | CNIL* . [Online]. At: https://www.cnil.fr/fr/definition/reconnaissance-faciale. [Visited on 27 mai 2021].

[7] Cabal (Christian), 2003. "Rapport sur les méthodes scientifiques d'identification des personnes à partir de données biométriques et les techniques de mise en œuvre". national assembly on 16 juin 2003, N° 938.

[8] "Visage" ≡ *Biometrie - Biometrics - Visage* . [Online]. At: https://www.biometrie-online.net/technologies/visage. [Visited on 27 mai 2021].

[9] "Histoire de la reconnaissance faciale" *Reconnaissance Faciale – TPE* . [Online]. At: https://labiometrie.wordpress.com/2017/02/12/reconnaissance-faciale/. [Visited on 27 mai 2021].

[10] F. Perronnin and J.-L. Dugelay. "Introduction à la biométrie –Authentification des individus par traitement audio-vidéo". Traitement du signal, Vol. 19, No. 4, 2002.

[11] " ENQUÊTE " *Comment la reconnaissance faciale s'installe en France | Les Echos* . [Online]. At: https://www.lesechos.fr/tech-medias/intelligence-artificielle/comment-la-reconnaissance-faciale-sinstalle-en-france-1140171. [Visited on 27 mai 2021].

[12] "La Sécurité" *Reconnaissance faciale : les 7 tendances de 2021 | Thales* . [Online]. At: https://www.thalesgroup.com/fr/europe/france/dis/gouvernement/biometrie/reconnaissance-faciale. [Visited on 27 mai 2021].

[13] P. Jonathon Phillips and Hyeonjoon Moon and Syed A. Rizvi and Patrick J. Rauss, "The FERET Evaluation Methodology for Face-Recognition Algorithms", IEEE Transactions on Pattern Analysis and Machine Intelligence, volume 22, number 10, pages 1090-1104, 2000

[14] M. Turk, A. Pentland, Eigenfaces for recognition. J. of Cognitive Neuroscience 3, 72–86,1991.

[15] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in Proc. IEEE Conf. Computer Vision Pattern Recognition, Jun. 1991, pp.586–591.

[16] "Abstract" *Linear discriminant analysis for face recognition* . [Online]. At: https://ieeexplore.ieee.org/document/5256630. [Visited on 27 mai 2021].

[17] "Face Detection with Haar Cascade". [Online]. At: https://towardsdatascience.com/face-detection-with-haar-cascade-727f68dafd08. [Visited on 27 mai 2021].

[18] "Local Binary Patterns" *Local Binary Patterns* . [Online]. At: http://www.scholarpedia.org/article/Local_Binary_Patterns. [Visited on 27 mai 2021].

[19] "Les réseaux de neurones" *Reconnaissance faciale | CNIL* . [Online]. At: http://tpe-biometrie.e-monsite.com/pages/reconnaissance-faciale.html. [Visited on 27 mai 2021].

[20] W. Hizem, « Capteur Intelligent pour la Reconnaissance de Visages », Doctoral thesis, Option : Informatics/Electronics, National Institut of Telecommunications at University Pierre et Marie Curie, Paris, 2009

[21] A. Chaari, « Reconnaissance de Visages par Réseaux d'Ondelette de Gabor », Doctoral Thesis, Discipline : Automatic, Informatics, University of Sfax, Tunisia, 08 Décembre 2009.

[22] M. Fedias, « Combinaisons de Données d'Espaces Couleurs et de Méthodes de Vérification d'Identité pour l'Authentification de Visages », Doctoral Thesis, Option: automatic, University Mohamed Khider, Biskra, 21 Février 2013

[23] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. An introduction to statistical learning, volume 112. Springer, 2013.

[24] A. L. Samuel. Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, 44(1.2):206–226, Jan 2000

[25] Mitchell T.M. Machine Learning. McGraw-Hill International Editions.
McGraw-Hill, 1997.

[26] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction, volume 1. MIT press Cambridge, 1998.

[27] S.J. Russell, S.J. Russell, P. Norvig, and E. Davis. Artificial Intelligence: A Modern Approach. Prentice Hall series in artificial intelligence. Prentice Hall, 2010.

[28] C.M. Bishop. Pattern Recognition and Machine Learning. Information Science and Statistics. Springer, 2006.

[29] C.M. Bishop. Pattern Recognition and Machine Learning, chapter Linear Models for Regression. Information Science and Statistics. Springer, 2006.

[30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org

[31] Jan A Snyman. Practical mathematical optimization. 2005.

[32] David M Allen. Mean square error of prediction as a criterion for selecting variables. Technometrics, 13(3):469–475, 1971.

[33] John Shore and Rodney Johnson. Properties of cross-entropy minimization. IEEE Transactions on Information Theory, 27(4):472–482, 1981.

[34] Bin Dai, Shilin Ding, Grace Wahba, et al. Multivariate bernoulli distribution. Bernoulli, 19(4):1465–1483, 2013.

[35] Sylvain Arlot, Alain Celisse, et al. A survey of cross-validation procedures for model selection. Statistics surveys, 4:40–79, 2010.

[36] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. Nature, 521:436–444, 2015.

[37] S. Haykin and S.S. Haykin. Neural Networks and Learning Machines. Number v. 10 in Neural networks and learning machines. Prentice Hall, 2009.

[38] Michael Egmont-Petersen, Dick Ridder, and Heinz Handels. Image processing with neural networks - a review. pattern recogn 35:2279c2301. 35:2279–2301, 10 2002.

[39] A. Graves, A. r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pages 6645–6649, May 2013.

[40] Yoav Goldberg. A primer on neural network models for natural language processing. Journal of Artificial Intelligence Research, 57:345–420, 2016.

[41] Masood Zamani and Stefan C Kremer. Neural networks in bioinformatics. In Handbook on Neural Information Processing, pages 505–525. Springer, 2013.

[42] " Рэлу супраць софтмакс | 2020" *адказ 1:* . [Online]. At: https://be.classicfoxvalley.com/key/relu-vs-softmax/. [Visited on 29 mai 2021].

[43] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain [j]. 65:386 – 408, 12 1958.

[44] Marvin L. Minsky and Seymour A. Papert. Perceptrons: Expanded Edition. MIT Press, Cambridge, MA, USA, 1988.

[45] Bernard Widrow and Marcian E. Hoff. Adaptive switching circuits. In 1960 IRE WESCON Convention Record, Part 4, pages 96–104, New York, 1960. IRE.

[46] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. Neural Networks, 4(2):251 – 257, 1991.

[47] Ning Qian. On the momentum term in gradient descent learning algorithms. Neural Networks, 12(1):145 – 151, 1999.

[48] Yoram Singer John Duchi, Elad Hazan. Journal of Machine Learning Research, pages 2121–2159, Jul 2011.

[49] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[50] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In Advances in neural information processing systems, pages 396–404, 1990.

[51] David A Fournier, Hans J Skaug, Johnoel Ancheta, James Ianelli, Arni Magnusson, Mark N Maunder, Anders Nielsen, and John Sibert. Ad model builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models. Optimization Methods and Software, 27(2):233–249, 2012.

[52] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929–1958, 2014.

[53] "Convolutional Autoencoders" *Convolutional Autoencoders – P. Galeone's blog* . [Online]. At: https://pgaleone.eu/neural-networks/2016/11/24/convolutional-autoencoders/. [Visited on 29 mai 2021].

[54] David H Hubel and Torsten N Wiesel. Receptive fields of single neurons in the cat's striate cortex. The Journal of physiology, 148(3):574–591, 1959.

[55] " Deep Learning Cage Match: Max Pooling vs Convolutions | LaptrinhX". [Online]. At: https://laptrinhx.com/deep-learning-cage-match-max-pooling-vs-convolutions-165907296/. [Visited on 29 mai 2021].

[56] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806, 2014.

[57] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

[58] " Day 63(DL) — Convolutional Neural Nets for Images". [Online]. At: https://medium.com/nerd-for-tech/day-63-dl-convolutional-neural-nets-for-images-3f42fd81c9e8. [Visited on 29 mai 2021].

[59] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, November 1998.

[60] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012.

[61] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions.

[62] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014.

[63] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[64] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. arXiv preprint arXiv:1709.01507, 2017.

[65] " What is a Raspberry Pi?" *What is a Raspberry Pi? | Opensource.com* . [Online]. At: https://opensource.com/resources/raspberry-pi. [Visited on 29 mai 2021].

[66] "Specifications" *Raspberry Pi 4 Model B specifications – Raspberry Pi*. [Online]. At: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/. [Visited on 29 mai 2021].

[67] " 5V Single-Channel Relay Module" *5V Single-Channel Relay Module - Pin Diagram, Specifications, Applications, Working* . [Online]. At: https://components101.com/switches/5v-single-channel-relay-module-pinout-features-applications-working-datasheet. [Visited on 29 mai 2021].

[68] "Solenoid Lock-style 12VDC" *Solenoids : Solenoid Lock-style 12VDC Philippines | Circ ...* . [Online]. At: https://circuit.rocks/solenoid-lock-style-12v-dc. [Visited on 29 mai 2021].

[69] " Pi Camera Module – 5MP" *Pi Camera Module Pinout, Features & How to Use* . [Online]. At: https://components101.com/misc/pi-camera-module. [Visited on 29 mai 2021].

[70] *Home - OpenCV*. [Online]. At: http://opencv.org. [Visited on 29 mai 2021].

[71] "About SQLite" *About SQLite*. [Online]. At: https://www.sqlite.org/about.html. [Visited on 29 mai 2021].

[72] "What is PHP?" *What is PHP? Write your first PHP Program* . [Online]. At: https://www.guru99.com/what-is-php-first-php-program.html. [Visited on 29 mai 2021].

[73] " What is Tkinter used for and how to install this Python Framework?" *What is Tkinter used for and how to install this Python Framework?* . [Online]. At: https://www.activestate.com/resources/quick-reads/what-is-tkinter-used-for-and-how-to-install-it/. [Visited on 29 mai 2021].

[74] "Project Jupyter | Home" *The Jupyter Notebook* . [Online]. At: https://jupyter.org/. [Visited on 27 june 2021].

[75] Mei Wang, Weihong Deng. Deep Face Recognition: A Survey. arXiv preprint arXiv: 1804.06655, 2020.

[76] "Deep Face Recognion" Omkar M. Parkhi,Andrea Vedaldi, Andrew Zisserman . [Online]. At: https://www.robots.ox.ac.uk/~vgg/publications/2015/Parkhi15/poster.pdf. [Visited on 20 juin 2021].

[77] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. University of Massachusetts, Amherst, Technical Report 07-49, October, 2007.

ANNEX A

Installing used libraries

OpenCV 4.1.0

To install OpenCV we must first install the required dependencies for that package by executing the following commands in the terminal:

```
$ sudo apt-get install build-essential cmake pkg-config

$ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev

$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev

$ sudo apt-get install libxvidcore-dev libx264-dev

$ sudo apt-get install libgtk2.0-dev libgtk-3-dev

$ sudo apt-get install libatlas-base-dev gfortran
```

After successfully installing the above mentioned packages we simply install the OpenCV package using pip with the following command:

```
$ sudo pip3 install opencv-contrib-python==4.1.0.25
```

Tensorflow 2.2.0

To install Tensorflow we followed this guide [https://qengineering.eu/install-tensorflow-2.2.0-on-raspberry-pi-4.html]. Which states that we need to install some prerequisite packages, download a compiled version of Tensorflow, then install it using pip.

```
# install the dependencies (if not already onboard)

$ sudo apt-get install gfortran

$ sudo apt-get install libhdf5-dev libc-ares-dev libeigen3-dev

$ sudo apt-get install libatlas-base-dev libopenblas-dev libblas-dev

$ sudo apt-get install openmpi-bin libopenmpi-dev

$ sudo apt-get install liblapack-dev cython

$ sudo pip3 install keras_applications==1.0.8 --no-deps

$ sudo pip3 install keras_preprocessing==1.1.0 --no-deps

$ sudo pip3 install -U --user six wheel mock
```

```
$ sudo -H pip3 install pybind11

$ sudo -H pip3 install h5py==2.10.0

# upgrade setuptools 40.8.0 -> 52.0.0

$ sudo -H pip3 install --upgrade setuptools

# install gdown to download from Google drive

$ pip install gdown

# copy binary

$ sudo cp ~/.local/bin/gdown /usr/local/bin/gdown

# download the wheel

$ gdown https://drive.google.com/uc?id=11mujzVaFqa7R1_lB7q0kVPW22Ol51MPg

# install TensorFlow

$ sudo -H pip3 install tensorflow-2.2.0-cp37-cp37m-linux_armv7l.whl

# and complete the installation by rebooting

$ sudo reboot
```

Keras 2.4.3

Installing Keras is far simpler since it uses Tensorflow as a backend meaning that all dependencies are already satisfied, the only thing left is executing this command in the terminal.

```
$ sudo pip3 install keras==2.4.3
```

Jupyter notebook

Jupyter is already available in pip, so we use the following command to install it

```
$ sudo pip3 install notebook
```

Other libraries

We have also used other libraries in our work which can be installed in a similar fashion to Jupyter's notebook; the libraries used are as follows: numpy, pillow, scikit-learn, matplotlib

```
$ sudo pip3 install package-name
```

ANNEX B

Creating a database in SQLite3

To create an empty database with the required access reports table first we define the functions responsible for creating a connection to the database and to make a table in it:

```
import sqlite3
def create_connection(db_file):
    conn = sqlite3.connect(db_file)
    return conn

def create_table(conn, create_table_sql):
    c = conn.cursor()
    c.execute(create_table_sql)
```

Next we need to define the path to where we want to store the database, then define the structure of the table we want to make, finally create a connection to the database and create our table in it:

```
# path to store database
database = r"./accessLogs.db"
# define table structure
sql_create_projects_table = """ CREATE TABLE IF NOT EXISTS report(
                                    datetime text,
                                    name text,
                                    probability text,
                                    face text
                                ); """
# create a database connection
conn = create_connection(database)
# create projects table
create_table(conn, sql_create_projects_table)
```

After executing this script we find that our database is created.

