JULY 2018

M.Sc. in Electrical and Electronics Engineering

SUFIYAN ABDUL-KARIM N-YO

UNIVERSITY OF GAZİANTEP

GRADUATE SCHOOL OF

NATURAL & APPLIED SCIENCES

DEVELOPMENT OF A REAL-TIME FACE RECOGNITION SYSTEM USING DEEP LEARNING TECHNIQUES

M. Sc. THESIS

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

BY

SUFIYAN ABDUL-KARIM N-YO

JULY 2018

Development of a Real-Time Face Recognition System Using Deep Learning Techniques

M. Sc. Thesis

In

Electrical and Electronics Engineering

University of Gaziantep

Supervisor

Prof. Dr. Ergun ERÇELEBİ

by

Sufiyan Abdul-Karim N-YO

July 2018

REPUBLIC OF TURKEY
UNIVERSITY OF GAZİANTEP
GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES
ELECTRICAL AND ELECTRONICS ENGINEERING

Name of the thesis: Development of a Real-Time Face Recognition System Using Deep Learning Techniques

Name of the student: Sufiyan Abdul-Karim N-YO

Exam date: 23.07.2018

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Ahmet Necmeddin YAZICI
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Ergun ERÇELEBİ
Head of Department

This is to certify that we have read this thesis and that in our consensus opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Ergun ERÇELEBİ
Supervisor

Examining Committee Members	Signature
Prof. Dr. Ergun ERÇELEBİ	.....
Prof. Dr. İlyas EKER	.....
Doç. Dr. Ahmet METE VURAL	.....

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Sufiyan Abdul-Karim N-YO

ABSTRACT

DEVELOPMENT OF A REAL-TIME FACE RECOGNITION SYSTEM USING DEEP LEARNING TECHNIQUES

N-YO, Sufiyan Abdul-Karim
M. Sc. In Electrical and Electronics Engineering
Supervisor: Prof. Dr. Ergun ERÇELEBİ
July 2018
58 Pages

Face recognition system is a non-invasive type of a biometric system used to verify or identify a person from a digital image. Face recognition systems are mainly used for security purposes, however, they can be used in many other areas such as entertainment and healthcare. A lot of work has been done in the literature about the accuracy and speed of face recognition systems over the years. In this thesis project, a real-time face recognition system is design and implemented on 2 different inexpensive platforms . The system used a combination of different machine learning and deep learning algorithms and techniques. There were four main stages for realizing this project. For the face detection, the Histogram of Oriented Gradients (HOG) was used because it is faster in detecting faces on a digital image. After detecting the face a modified version of face landmark estimation algorithm was used to generate 5 face landmarks used to center the face before passing it on to a pre-trained face_recognition model which generates 128 embedding for the face. Finally, the system used a Support Vector Machine (SVM) classifier to identify whose face is on the image. The system reached a performance of 96.88% accuracy and approximately 3 frames per second when tested with a database of 40 images of 8 different individuals. This thesis project shows that real-time face recognition systems based on recent deep learning techniques can be implemented on a limited computer hardware.

Key Words: Face recognition, deep learning, machine learning, computer vision

ÖZET

DERİN ÖĞRENME TEKNİKLERİNİ KULLANARAK GERÇEK ZAMANLI YÜZ TANIMA SİSTEMİNİN GELİŞTİRİLMESİ

N-YO, Sufiyan Abdul-Karim
Yüksek Lisans Tezi, Elektrik Elektronik Mühendisliği
Tez Yöneticisi: Prof. Dr. Ergun ERÇELEBİ
Temmuz 2018
58 Sayfa

Yüz tanıma sistemi, dijital görüntüdeki bir kişiyi doğrulamak veya tanımlamak için kullanılan biyometrik bir sistemin invaziv olmayan bir türüdür. Yüz tanıma sistemleri ağırlıklı olarak güvenlik amaçlı kullanılmaktadır, ancak eğlence ve sağlık gibi birçok alanda kullanılabilirler. Literatürde yüz tanıma sistemlerinin doğruluğu ve hızı hakkında birçok çalışma yapılmıştır. Bu tezde, gerçek zamanlı yüz tanıma sistemi, 2 farklı ucuz platformda tasarlandı ve uygulandı. Sistem farklı makine öğrenime ve derin öğrenme algoritma ve tekniklerinin bir kombinasyonunu kullanmıştır. Yüz tanıma sistemini dört ana aşamada gerçekleştirilmiştir. Yüz algılama için, Dijital Görüntüdeki yüzlerin algılanmasında daha hızlı olduğu için Yönelimli Eğim Histogramı (HOG) kullanıldı. Yüzü tespit ettikten sonra, yüz için 128 gömme üreten yüz tanıma modelinin eğitimi öncesine geçmeden önce yüzü ortaya getirmek için kullanılan 5 yüz yer işareti üreten yüz yer işareti tahmin algoritması kullanılmıştır. Son olarak, imgede kimin yüzü olduğunu tanımlamak için bir Destek Vektör Makinesi (SVM) sınıflandırıcı kullanılmıştır. Sistem 8 farklı kişinin 40 görüntüsünden oluşan bir veritabanı ile test edildiğinde % 96.88'lik doğruluk performansı elde etti. Bu tez, son derin öğrenme tekniklerine dayanan gerçek zamanlı yüz tanıma sistemlerinin sınırlı bir bilgisayar donanımı üzerinde uygulanabileceğini göstermektedir.

Anahtar Kelimeler: Yüz tanıma, derin öğrenme, makine öğrenimi, bilgisayar vizyonu

Dedicated to my wife and my family.

ACKNOWLEDGEMENTS

I wish to express my deepest gratitude to my head of department and supervisor Prof. Dr. Ergun ERÇELEBİ for his guidance, advice, encouragement, kindness, and insight throughout the research.

I will also like to thank all the staff and professors at the department of electrical and electronics engineering in Gaziantep University.

I equally acknowledge and thank the Turkish Government Scholarship office for their financial support. Finally, many thanks go to my wife and family for helping me to accomplish this research work.

TABLE OF CONTENTS

Page

ABSTRACT ..... **v**

ÖZET ..... **vi**

ACKNOWLEDGEMENTS ..... **viii**

TABLE OF CONTENTS ..... **ix**

LIST OF FIGURES ..... **xii**

LIST OF TABLES ..... **xiv**

ABBREVIATIONS ..... **xv**

CHAPTER 1 ..... **1**

INTRODUCTION **1**

1.1 Introduction 1

1.2 Statement of the Problem 3

1.3 Purpose of Thesis 4

1.4 Contribution of the Thesis 4

1.5 General Outline of this Thesis Project 5

CHAPTER 2 ..... **6**

LITERATURE REVIEW **6**

2.1 Introduction 6

2.2 Related Work in Literature 6

2.2.1 Traditional Methods 6

2.2.1.1 Feature-based approach 7

2.2.1.2 Holistic-based approach 9

2.2.2 Modern Methods 10

CHAPTER 3 ..... **14**

THEORETICAL BACKGROUND OF DEEP LEARNING (DL) 14
3.1 Artificial Neural Networks (ANNs) 15
3.1.1 Structure of ANN 16
3.1.2 How ANN works 17
3.1.3 ANN training 21
3.2 Deep Learning (DL) 22
3.2.1 Deep Neural Networks (DNNs) 24
3.2.2 Convolutional Neural Networks (CNNs) 25
CHAPTER 4 27
DESIGN AND METHODOLOGY 27
4.1 Face detection 28
4.2 Face posing and projecting 31
4.3 Face recognition 32
4.4 Face classification 35
CHAPTER 5 36
EXPERIMENTS, RESULTS & DISCUSSION 36
5.1 System hardware 36
5.1.1 Dell Laptop with Intel core i7 processor 37
5.1.2 Embedded system (Raspberry Pi 3 model B) 38
5.1.3 Raspberry Pi camera V2.1 38
5.2 System Software 39
5.3 Datasets 40
5.4 Applications of our system 43
5.4.1 Identifying faces on a dataset of images 43
5.4.2 Identifying faces on a live video from our webcam 45
5.5 Results and Discussion 48
5.5.1 Results for test on Intel core i7 laptop and Raspberry Pi 3 48
5.5.1.1 System accuracy 48
5.5.1.1.1 Comparison of results with the state-of-the-art 49

5.5.1.2 System speed 49
5.5.2 Comparison of the results of the Intel core i7 and Raspberry Pi 3 50
5.5.2.1 Which platform can support real-time face recognition system? 51
5.6 Limitation of our system 51
CHAPTER 6 53
CONCLUSION AND FUTURE WORKS 53
6.1 Conclusion 53
6.2 Future work 53
REFERENCES 55

LIST OF FIGURES

Page

Figure 1.1. General architecture of a face recognition system 3

Figure 3.1. General structure of the perceptron 16

Figure 3.2. Simple Structure of a Multiple Layer ANN 17

Figure 3.3. Graphical representation of the sigmoid function 18

Figure 3.4. Graphical representation of the hyperbolic tangent function 19

Figure 3.5. Graphical representation of the ReLU function 20

Figure 3.6. An example of how deep neural network learn features from each layer until it is able to recognize more complex abstractions 24

Figure 3.7. Structure of a CNN. The images reduces in size until it reaches the fully connected layer 25

Figure 4.1. General structure of our system 28

Figure 4.2. Representation of the major features of an image by the HOG method...30

Figure 4.3. The faces on the center and left are detected by the HOG method from the image on the right 30

Figure 4.4. Demonstration of affin transformation of a face 32

Figure 4.5. Iluustration of 128 measurements generated from an image 34

Figure 4.6. Demostration of FaceNet's triplet-loss training process 35

Figure 5.1. Dell Latitude E6530 Laptop 37

Figure 5.2. Raspberry Pi 3 B 38

Figure 5.3. Raspberry Pi camera model V2.1 39

Figure 5.4. Images of the training dataset for this project 41

Figure 5.5. Images of dataset for testing our system 42

Figure 5.6. Results of face recognition on still images 44

Figure 5.7. Recognition of more than 1 face on an image 45

Figure 5.8. Face recognition from a live video 46

Figure 5.9. Face recognition of 2 faces from a live video 47

Figure 5.10. Face recognition of 2 people in front of a live camera 48

LIST OF TABLES

Page

Table 5.1. Comparison of the accuracies of different face recognition techniques...49

Table 5.2. Time Analysis of the System on a Typical Intel Core i7 Laptop.....50

Table 5.3. Time Analysis of the System on Raspberry Pi 3B 50

ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Network
GPU	Graphical Processing Unit
HOG	Histogram of Oriented Gradient
LBP	Local Binary Patterns
LFW	Labelled Faces (in the) Wild
ML	Machine Learning
MLP	Multilayer Perceptron
PC	Personal Computer
PCA	Principal Component Analysis
RBF	Radial Basis Function
RNN	Recurrent Neural Network
SIFT	Scale-Invariant Feature Transform
SVM	Support Vector Machine
VGG	Visual Geometry Group

CHAPTER 1

INTRODUCTION

1.1 Introduction

Recent advancement of machine learning (ML) algorithms, data analytics, and the internet are generating more and more data on daily basis. This, coupled with the increase in processor power and speed has created enormous opportunities for researchers to develop robust face recognition systems. Face recognition is a biometric technique that researchers have been working on since the 1970's. Face recognition is about teaching computers to see and identify faces in a similar way that the human visual system functions.

It has numerous applications in industries such as security, healthcare, entertainment and even retail. Recently, some educational institutions use face recognition systems to detect whether students are paying attention during lessons. Face recognition systems are equally deployed in borders of countries to detect and recognize faces of known criminals trying to enter a country. Furthermore, face recognition systems are used in aging homes of the elderly to monitor the health conditions of the aged without intruding them. There are many face recognition methods proposed in the literature, we have generally grouped them into traditional methods and modern methods based on deep learning (DL) and ML techniques. A detailed literature review is presented in chapter 2.

The traditional methods are generally based on geometric approach which looks to recognize faces based on special features on a face as compared to the faces in the dataset, and then the photometric approach which relies on converting an image into values then mapping it with a face template values in order to find a face match or difference. Some examples of such traditional methods are Fisher Vector algorithm

[1], and Eigenfaces which is used in the Principal Component Analysis (PCA) method [2]. These methods either use handcrafted local image descriptors such as Local Binary Patterns (LBP) [3] or Scale-Invariant Feature Transform (SIFT) [4] to extract features of the face from the image or through holistic features. These traditional methods have achieved good results over the years. However, the linear nature of these methods cause them to fail under extreme conditions such as poor lighting, faces with glasses, or faces with different pose as compared to the ones in the dataset. This happens because the face recognition problem is generally a nonlinear problem, hence, more robust methods are developed to solve the problem.

Modern face recognition methods are based on DL techniques. Due to the presence of huge amount of data and the more powerful CPUs and GPUs, researchers have been able to develop more robust face recognition systems with very high accuracy that can work in many different environments and conditions. The general working principle of these DL based methods is that when data in the form of digital images are fed into a DL algorithm the algorithm learns some features of the data over time. After the DL algorithm has learned enough of the patterns or features from the data, it will be capable of learning the pattern from a new digital image which it has not seen before and make a comparison with the old patterns learned from the images on the stored dataset in other to identify what is on the image.

The general structure of a face recognition system is such that the digital image is first acquired by a camera or any other image acquisition device, after which the image is preprocessed and passed to the next stage to extract the facial features. Then a classifier is used to classify the facial features taking into consideration the faces stored in the database. If a match is found then the system identifies the face otherwise it is an unknown face. Figure 1.1 shows the general structure of a face recognition system. The first part is the face detection phase where there are several different methods that can be used for the detection. The second main part is the face recognition phase where the actual recognition is done, recognition can be achieved by many different recognition methods as mentioned above. The method used for the face detection in

our system will be presented in detail in chapter 4 and some of the methods for face recognition and the selected method will be discussed in Chapter 2.

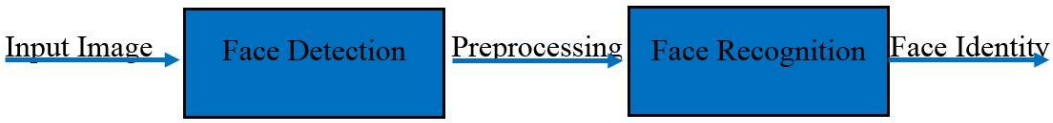

Figure 1.1. General architecture of a face recognition system

1.2 Statement of the Problem

The human visual system can detect and recognize faces within milliseconds with ease. However, computers are not capable of high-level generalizations such as recognizing what is on an image. A computer has to be trained step by step to do these kind of high level generalizations. Face recognition has always been a difficult problem for Computer Vision (CV) and Image analysis researchers. A review of the traditional methods shows that under certain conditions such as bright lighting, different facial conditions or different pose of the face the methods perform poorly. The recent use of DL and ML techniques has significantly increased the accuracy of face recognition under various conditions. However, face recognition is still a difficult problem for researchers because the amount of data and computational resources needed to build these systems to perform properly is very large and expensive. There is active ongoing research regarding which DL method is the best for face recognition. Regardless of the method selected, accuracy and speed of the system are the main concerns when building face recognition systems. The most robust face recognition systems are currently built in industrial research laboratories because they have large datasets and powerful computational resources at their disposal. Another problem is that due to how big and how interconnected these processors are, the training of the deep convolutional neural networks is stationary, in other words the training takes place in setup laboratories and takes a long time to complete the whole process. In addition, due to

the inadequate availability of large-scale data to the academic research community and face recognition enthusiast, it is much difficult to build these systems from scratch.

This work is built with the deep methods which are based on Convolutional Neural Networks (CNNs). CNNs are neural networks with multiple layers; they are very good at finding patterns in visual data [5]. If properly trained with visual data they can successfully find objects in images and videos with very high accuracy. There are many DL based methods available, the current state-of-the-art methods are FaceNet [6] by Google and DeepFace [7] and its variations by Facebook. These networks require very large scale datasets and powerful stationary processors to train them as stated above. Their datasets are also not available for usage by individual researchers.

1.3 Purpose of Thesis

The main purpose of this thesis is to design and develop a real-time face recognition system using DL techniques on both an in-expensive embedded system (Raspberry Pi 3) and a typical 8-Core Intel Core i7 processor. As we are in the age of the Internet of Things (IoT) it has become important for us to develop mobile real-time face recognition systems that we can easily incorporate into the IoT infrastructure. This is why developing real-time face recognition mobile and easily accessible platforms such as embedded systems and an everyday standard laptop are important. The accuracy, execution time and the ability of the CNN to learn the face pattern from a single image in mobile scenarios are the main aspects that we focused on in this thesis. After building the system on both hardware. We evaluated the performance (accuracy and speed) of each and presented our result on which one is best for real-time recognition. We demonstrated that fast and accurate face recognition based on DL can also be achieved on limited computational resources. Our system actually tells whose face it is on the image or video not just detecting the faces.

1.4 Contribution of the Thesis

Our development can be used in many areas such as multimedia, personal identification - driver licenses, passports at border controls, video and home surveillance. The developed system can also be integrated with the components of IoT

for person identification and security purposes without spending a huge budget on expensive computational platforms which are not portable. We used an inexpensive Raspberry Pi 3 model B and a typical Dell laptop with an Intel processor to achieve performance close to that of an expensive GPU hardware. To increase accuracy, the best deep CNN techniques and methods were used. The development of the real-face recognition system is an integrated system consisting of low-cost hardware, software, and suitable deep CNN methods.

1.5 General Outline of this Thesis Project

This thesis project is divided into six different chapters:

Chapter 1: This chapter presented the introduction to the thesis, the statement of the thesis problem and the purpose of the thesis.

Chapter 2: In this chapter, a thorough review of the literature is presented.

Chapter 3: This chapter presented the theoretical background of deep learning.

Chapter 4: In this chapter, we explained the design and methodology used in the thesis project.

Chapter 5: This chapter describes how the experiments were carried out on two different platforms and the results were also analyzed and compared.

Chapter 6: We concluded and suggested improvements to our system in this chapter.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter presents the literature review on face recognition. There has been active research on face recognition for about 5 decades now by researchers from various backgrounds including CV, neuroscience, artificial intelligence (AI) and signal and image processing etc. A thorough literature survey on the research relevant to this thesis has been presented in the sections below. The review has guided us on the best method to use for our project.

2.2 Related Work in Literature

We grouped our review into traditional methods and modern methods. With the traditional methods being the classical face recognition methods while the modern methods being the AI and DL based methods. We first present the traditional methods in section 2.2.1 and then the modern methods in section 2.2.2.

2.2.1 Traditional Methods

Research in face recognition first started in the 1970's [8]. The idea was to teach computers how to see and recognize a face on an image just like how the human sees faces. However, after many years of active research, it turned out that the face recognition problem was more difficult than initially thought of.

In general, the idea is that when an image with one or multiple faces is passed into the face recognition system as an input the system first isolates all the faces by detecting them. The faces then go through a pre-processing stage by rotating, scaling, translating

etc. before each face is represented in a low-dimensional form. Finally, the low-dimensional representations are classified. There are many challenges associated with

the face recognition systems because images of a person's face can be taken in several angles that make them look different. The face recognition system has to be resilient to face changes on an image such as age, pose, facial expressions and facial deformations due to an accident and at the same time able to distinguish between faces of two different people.

In 2009, Jafri and Arabnia [9] published the most comprehensive literature survey on face recognition techniques and methodologies up until that year. Over the years there have existed many good techniques and methods developed by researchers. In general, up till the late 2000's, we categorized face recognition methods into holistic matching methods and feature-based methods. Most of the earlier research on face recognition was based on the feature-based approach.

2.2.1.1 Feature-based approach

The feature-based approach is a biologically inspired approach which relies on intensity data without assuming any knowledge of the structure of the face. It works by identifying and measuring peculiar facial characteristics from the face on the input image [9], examples of such facial characteristics are the nose, eyes, mouth, etc. Then the face in the image is simplified and represented in a vector of geometric features by computing the geometric relationships among these facial feature points. These geometric vectors are then fed into standard statistical pattern recognition methods to match faces in the database and the input faces. One example of such early works on face recognition was developed by Kanade [8] in 1973. He used simple image processing techniques to extract a vector comprising of 16 facial parameters. These parameters were ratios of angles, areas, and distances. He also employed a simple Euclidean distance measure technique for the face recognition and achieved a performance of about 75% accuracy on a dataset of 40 images (20 different individuals, 2 images each).

From this point on Brunelli and Poggio in 1993 developed upon Kanade's work by increasing the size of the geometric features of the vector to 35 and a larger dataset of 188 images (4 images per person). They reported a 90% accuracy which was a significant improvement at that time. However, when they used a different technique

called template-matching approach on the same dataset they had a perfect score of 100% accuracy on the performance [10].

As the early research work continues more sophisticated feature-based methods were developed. Some of these sophisticated techniques include Hough transform [11], Graf's filtering and morphological operations [12] and Reisfeld's symmetry operator [13]. All these methods used some form of a deformable template. These template approach produced better performance results as compared to the methods before them. However, since these models cannot perfectly fit the structures of the faces in the image they needed to introduce some form of a threshold for tolerance. The problem with the tolerance is that a big tolerance value makes the accuracy higher but with also a high possibility of false negatives, in other words, it will wrongly recognize 2 different people as the same. On the other hand, if the tolerance value is too small the accuracy will be affected greatly. Hence, these methods are very insensitive to small changes to the tolerance value chosen [9].

In 1996, Cox et al. later on developed a face recognition system still based on the feature- based approach which used a relatively large dataset with respect to the earlier systems. They used a dataset of 658 images (1 image per person) and manually extracted 35 facial features to form the geometric vector and achieved an accuracy of 95%. The issue with this system is the manual feature extraction, it is assumed that if the feature extraction stage was automated the accuracy would have been reduced and the system less precise [14].

There are many other feature-based methods such as the elastic bunch graph matching method developed by Wiskott et al. [15] in 1997 which achieved even better results. However, the fundamental problems remain the same for the feature-based methods. The facial features have to be manually extracted, hence, the implementer has to make a decision on which features are more important for detection. It is really difficult to automate the feature extraction stage without significantly compromising the performance of the system.

2.2.1.2 Holistic-based approach

The holistic approach was developed to try to identify faces in the entire image rather than focusing on the local features of the face as mentioned above. The holistic approach is generally classified into statistical and artificial intelligence approaches.

The statistical approach represents an image into a 2D array of intensity values and then compare these intensity values directly with those of the images in the dataset. This method works very well under controlled conditions [16]. Nevertheless, it is computationally expensive when doing things in a high dimension. Additionally, when tested in an unregulated environment such as different illumination, face orientation, background clutter, and noise the performance suffered [17].

In 1987, Sirovich and Kirby [20] used the PCA [18, 19] method to develop the first technique to reduce the dimensions of the statistical method to an economical level. They showed that the PCA can efficiently represent any face along the eigen pictures coordinate space and the face can easily be reconstructed by a combination of the eigen pictures and their projection along each eigen picture.

Turk and Pentland [21, 22] in 1991 developed a face recognition system based on the work of Sirovich and Kirby. The idea was that faces could be recognized by utilizing the projections along eigen pictures as classification features. Their face recognition system was initially tested using a dataset of 2,500 images (about 156 images per person) and achieved the most robust performance at that time. However, under different scales the system performed poorly. Over the years Turk and Pentland's face recognition system have been improved and tested using larger datasets. The performance also saw an improvement, however, the system seems to work well with only one image per person. When there are multiple images of the same person in the dataset the PCA retains unwanted variations and the performance of the system is reduced [1].

In 1994, Moses et al. [23], developed a method called Fisherfaces which relies on subspace projection before the Linear Discriminant Analysis (LDA) projection. They reported that this method was better than the PCA when tested under a wide range of conditions such as low and high lighting, different facial expression, and poses.

However, reference [24] proved that when the Fisherfaces method is tested using a small dataset, the PCA still performs better than the LDA.

Many other methods and techniques where developed after the Fisherfaces method to compensate for the drawbacks of the previous method. However, due to limited space we cannot present the details in this thesis. A comprehensive overview of the methods can be found in reference [9].

2.2.2 Modern Methods

The advancement of AI, big data, and powerful computer processors has brought about more interest in face recognition research in recent times. The modern methods use artificial neural networks (ANNs) and ML techniques to recognize faces. These methods generally produced far better results as compared to the traditional methods. We will present some of the relevant and the state-of-the-art methods in the following paragraphs.

In 1997, Lawrence et al.[25] first presented an AI technique that uses CNN to recognize faces. From that time there has been an explosive amount of face recognition research based on these CNNs, the ones significant to these research work are presented in the paragraphs following this. From the 2010's there have been tremendous gains in the performance and accuracy of these DL based systems.

In 2014, Y. Taigman et al. [7] presented their face recognition system called DeepFace developed in the Facebook research laboratory. This system was based on deep architectures for the face recognition. They used nine layers of the neural networks with more than 120 million connected weights. They trained their CNN with a dataset of 4.4 million labeled images (4000 different individuals) created from the internet, the largest dataset available at that time. Their system used a CNN feature extractor, which is a learnable function that uses a group of linear and non-linear operators. The Siamese network architecture was used to build DeepFace. This network works by applying the same CNN to a pair of faces to get the descriptors which are compared using the Euclidean distance. The goal of the training is to minimize the Euclidean distance between a pair of the same face and maximize that of 2 different faces, this is known as metric learning. DeepFace achieved the state-of-the-art performance of

97.35 % accuracy at that time [7]. From that time there have been various forms of DeepFace called DeepID – 2, 2+ and 3, built with more layers and 200 networks and tested on even larger datasets. These variants of DeepFace achieved an accuracy of 99.47% recently when tested on labelled faces in the wild (LFW) [26] .

It is worth to mention the Face Descriptor [27] and a lightened version of CNN [28] developed by the Visual Geometry Group (VGG) at the University of Oxford in 2014 and 2015. They used fewer layers and created their own dataset which was smaller in magnitude as compared to the Facebook dataset and yet they reported an accuracy of 98.95% which was impressive.

In 2015, the AI research group at Google presented FaceNet [6], a face recognition system that can be implemented on a very large scale. This system learns a mapping from face images to a compact Euclidean space directly. Then standard techniques using FaceNet embedding as feature vectors can be used to implement face recognition, verification, and clustering once the Euclidean space has been created [6]. FaceNet's CNN which is similar to that in references [29,30] was trained using a dataset of 200 million labeled images of 8 million different individuals. The CNN was very similar to that of DeepFace. However, FaceNet used a "triplet-based" loss function which minimizes the Euclidean distance between two faces of the same person and maximizes the Euclidean distances between a third face which is of a different person and first two faces. This method achieved the best performance on the LFW among all the state-of-the-art methods. It achieved an accuracy of 99.63% on the LFW [6].

Upon all these closed to human level recognition performance of these state-of-the-art systems, there are still major challenges in face recognition research. The training of these systems requires a large amount of labeled data and very expensive computer systems. This magnitude of data and computational resources are not available to the public or academic research groups yet. As a result, there is active research going on about how to reduce these state-of-the-art CNNs to an economic level in terms of both the data needed and computer hardware required while achieving near the state-of-the-art performance. The next paragraphs present 2 of the best-lightened versions of the state-of-the-art systems.

In 2016, B. Amos et al. [31] published their face recognition library called OpenFace. OpenFace is a general-purpose library for face recognition, particularly for real-time mobile scenarios. Their main goal was to develop a face recognition system light enough to be implemented on a typical 8- Core Intel Xeon E5-1630 v3 @ 3.70GHz CPU and train with small dataset which is available to the public. They built their system with Google's FaceNet architecture [6] and a combination of pre-trained open sources libraries such as dlib's library [32], OpenCV [33] and scikit-learn [34]. They reported an accuracy of 92.92% after testing their system on the LFW, which is good enough considering the system was tested on a typical inexpensive computer with a small dataset and better than the best OpenCV face recognition methods such as the popular Fisherface algorithm [1]. They released OpenFace to the public as a pre-trained face recognition library. However, they stated that they did not study the performance of this system on an embedded system or any other hardware for that matter except the 8-core Intel Xeon CPU [31]. So at the time of writing this thesis, there is still room to investigate the OpenFace method on other systems including embedded systems.

In 2017, Adam Geitgey released face_recognition [35] library which was built with dlib's state-of-the-art face recognition [36] which in turn was based on the reference [37] deep residual network (ResNet). Face_recognition is based on current deep learning techniques, it is a pre-trained model with 3 million images and has an accuracy of 99.38% [35] on the LFW. This model is a modified version of the ResNet-34 neural network [37] in which the convolutional layers have been reduced to 29 layers and the filters in each layer were reduced by 50%. This model has been pre-trained to map a face to 128-dimensional vector space, such that when an input image is passed through the network it generates 128 measurements called embedding for the face. It is lighter and has better accuracy than OpenFace [31]. It has the best accuracy among the pre-trained lighter CNN versions and outside the state-of-the-art methods as at March 2018.

In this research work, we used Face_recognition [35] for building our system because it is pre-trained with higher accuracy and also its simplicity to use. We designed and implemented our system with a key consideration to it running in real time on an inexpensive mobile hardware and requiring only 1 image of a person to generate the

face embeddings. We, therefore, implemented the system on a typical Dell Laptop and an inexpensive embedded system (Raspberry Pi 3B) in order to investigate the performance on an inexpensive hardware. The details of this method are presented in chapter 4. The results of our experiments are presented in chapter 5, and conclusion made in chapter 6.

CHAPTER 3

THEORETICAL BACKGROUND OF DEEP LEARNING (DL)

The introduction of the concept of DL is presented in this chapter. Before we dive into DL itself we need to first explain what ML and ANNs are. DL is a type of ML technique that is based on ANNs.

ML is a branch of AI and is based on statistical learning. Tom Mitchell [38] formally defined ML as: *"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."*

In other words, if an algorithm is fed with enough data or sample data it can learn patterns in the data without following any particular sequential program or code, and with this experience, the algorithm can predict or estimate the function or outcome from similar new data. Machine learning algorithms are generally grouped into two forms of learning:

Supervised learning : In supervised learning, the algorithm is trained with a dataset of examples with their corresponding labels also known targets. In this case, the algorithm learns features from the example dataset and their corresponding expected outcomes known by the user. It is called supervised learning because during the training both the input data and the corresponding targets are known by a so-called teacher or supervisor who shows the algorithm what to learn. To illustrate, if a dataset containing examples of different sizes of apartments in different neighborhoods and their associated prices is used to train by supervised learning, the algorithm will learn the patterns and features in the dataset and will be able to accurately predict prices of new apartments. Supervised learning algorithms are good for solving many machine learning problems including classification and regression. Most of the solved problems

in machine learning today are trained by supervised learning because research in this area is almost matured now as compare to unsupervised learning.

Unsupervised learning: With unsupervised learning, the training dataset has many features and patterns to learn. However, here the data is not labeled so there are no specific targets corresponding to the input data. Unsupervised learning algorithm looks to learn useful features and structure from data and make sense of it on its own without a supervisor instructing it on what to do. Many unsupervised learning algorithms are good in clustering data, that is grouping data into different clusters depending on the sense they make.

As mentioned above, supervised learning is more popular at the moment, however, in practice sometimes there is not a clear line distinguishing the two methods during training. It normally depends on the problem at hand, it may sometimes require that a mix of the two methods are used. There are also other forms of learning such as reinforcement learning which is beyond the scope of this thesis.

3.1 Artificial Neural Networks (ANNs)

An ANN is an interconnected web of nodes call neurons (activation functions) and edges which connect these neurons together. ANN is a technique used in ML to learn from data. ANN's principal function is to receive an external input data and perform a series of complex calculations and pass the results to the output of the network to be used to solve problems. Neural Networks are so good at solving problems that are very hard to solve by traditional logical sequence computer programs. For example, a simple classification of oranges and bananas into two different groups can be a daunting and time-consuming task if we are to program each step of the classification process by the tradition coding methods. This and even more challenging problems such as a computer identifying an object or a person's face on an image or video, speech recognition etc. are some problems ANNs are good at solving. The recent success in the field of AI and related fields is as a result of the ability to train neural networks properly. In the next subsection, a brief introduction of the technical concepts and structure of ANNs is presented.

3.1.1 Structure of ANN

The development of ANNs started with the development of the perceptron by Frank Rosenblatt [39] in the 1950's. His work was inspired by biological neurons and earlier works of McCulloch and Pitts [40]. The perceptron was a simple artificial neuron with several inputs, for example, x_1, x_2, x_3 etc. and an output. Figure 3.1 represents the structure of a basic perceptron. It works by taking in several inputs and computing the weighted sum of the input values to produce the output result of 0 or 1. The results depend on whether the weighted sum is more or less than a certain set point value.

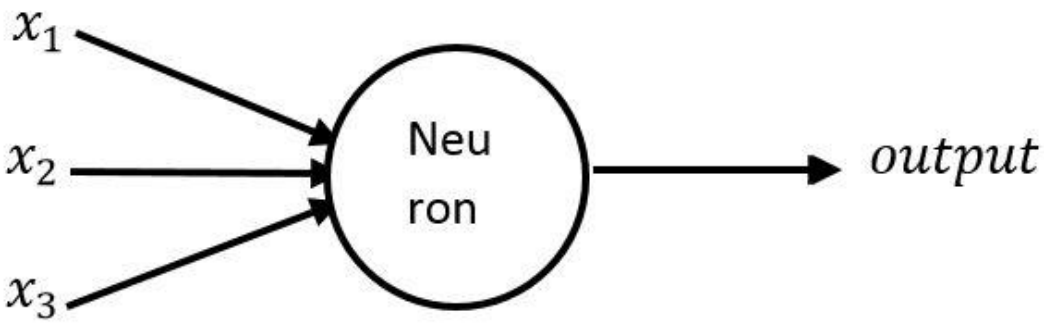

Figure 3.1. General structure of the perceptron

Later work and modern structure of ANNs are based on this structure as a starting point. An ANN is a highly structured network which is arranged in layers. The first layer is known as the input layer while the last layer is referred to as the output layer. Then either one or more layers in between the input and output layers called hidden layer. Every layer is composed of several many neurons and each neuron in a layer is connected to all the neurons in the previous layer. Figure 3.2 shows the basic structure of a multilayer ANN. The neurons in the network are connected to one another by edges which are normally associated weights. The weights show how strong one neuron is connected to another. Sometimes an additional branch called a bias is connected to the neurons. The bias has a value of +1 or -1 and it is not connected to the neurons of the previous layer. The purpose of the bias is to increase the capacity of the neural network to effectively solve problems.

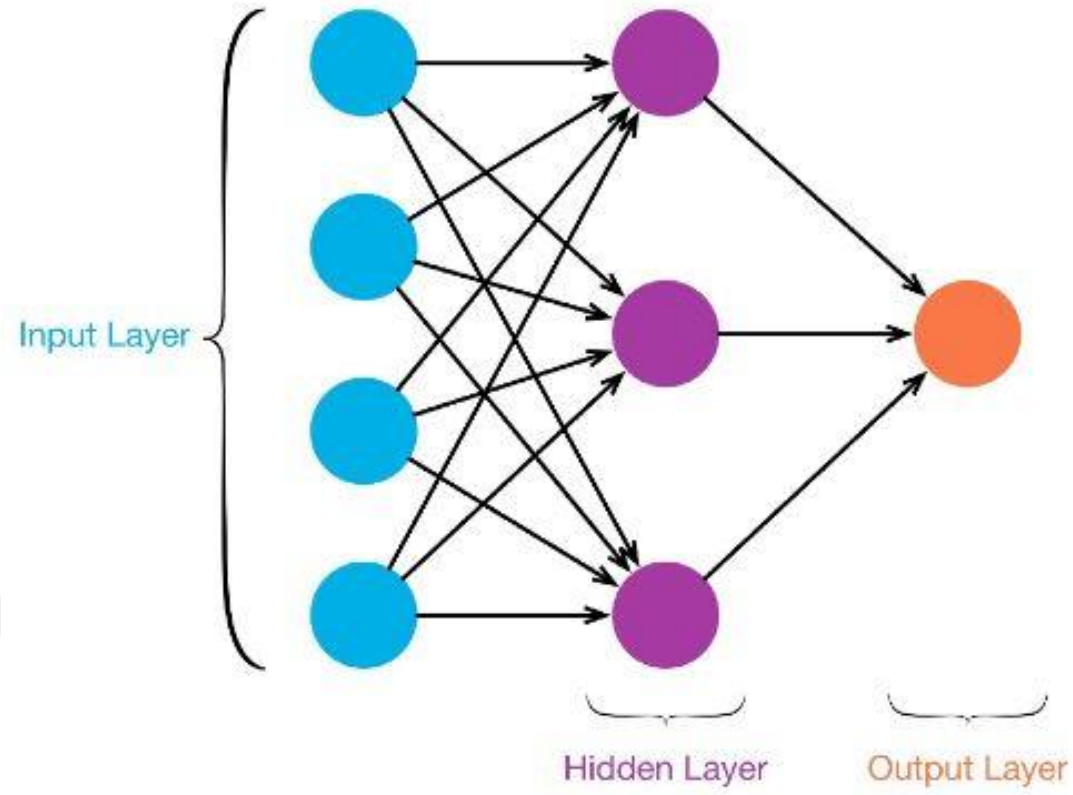

Figure 3.2. Simple Structure of a Multiple Layer ANN [41].

There are several variations of ANN such as Recurrent Neural Networks (RNNs), Feedforward Neural Networks, CNNs and all these variations originated from the basic structure we described above.

3.1.2 How ANN works

As explained in the above subsection the basic unit of an ANN is called a neuron (activation function). A neuron can be any mathematical function that takes an input and perform some mathematical operations and produce a result as an output.

To compute the activation value of a neuron n fundamental elements needed are the input values X_n, the activation function $g(m)$ and the weights W_n. The outputs of the previous layer become the input of the neurons in the following layer. A bias b is normally added to the neurons in the hidden layers, these biases are not coming from any previous neuron they are just on their own. The purpose of the bias is to improve the efficiency of the ANN. Given Z inputs, the value used by neuron n from layer q to compute the activation function is given by the equation below:

$$y_n = g(x_{q-1}) = \sum_m^Z (W_{n,m} \times x_{q-1,m}) + b_q \quad (3.1)$$

This equation depicts the linear addition of all the products of weights and input values of each neuron. The nonlinear activation function φ is represented by Equation 3.2.

$$f(x) = \varphi(g(x)) \quad (3.2)$$

Some of the most popular activation functions used as neurons of an ANN are :

1. The sigmoid or logistic function which squashes the input values between 1 and 0. It is widely used for classification problems and the mathematical Equation 3.3 and Figure 3.3 shows the logistic function:

$$\varphi(x) = \frac{1}{1+e^{-(x)}} \quad (3.3)$$

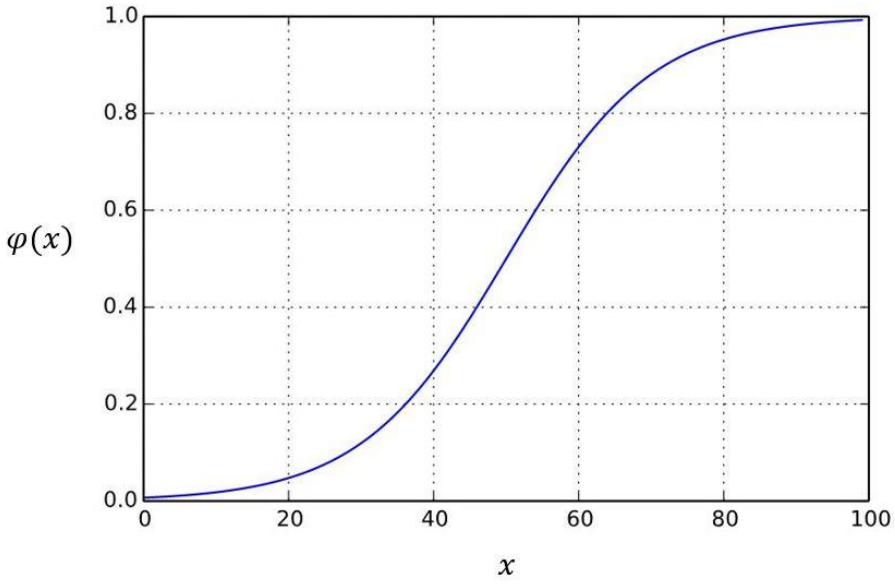

Figure 3.3. Graphical representation of the sigmoid function

2. The Hyperbolic Tangent or Tanh function which scales the input values within -1 and +1 by using a threshold value. It is similar to the sigmoid function. The Equation 3.4 and Figure 3.4 below depicts the Tanh function:

$$\varphi(x) = \frac{e^x}{1+e^x} \quad (3.4)$$

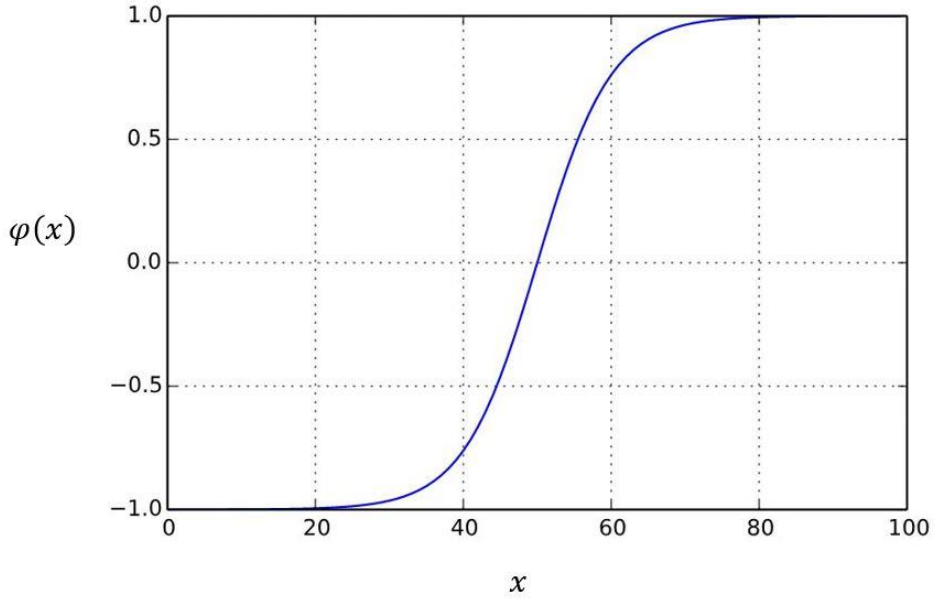

Figure 3.4. Graphical representation of the hyperbolic tangent function

3. The ReLU (Rectified Linear Unit) is a powerful function which allows only positive values to pass through and makes all negative values zeros. It is the most widely used activation function in DL and can only be used in the hidden layers. Equation 3.5 and Figure 3.5 represent the ReLU.

$$\varphi(x) = \max(0, x) \tag{3.5}$$

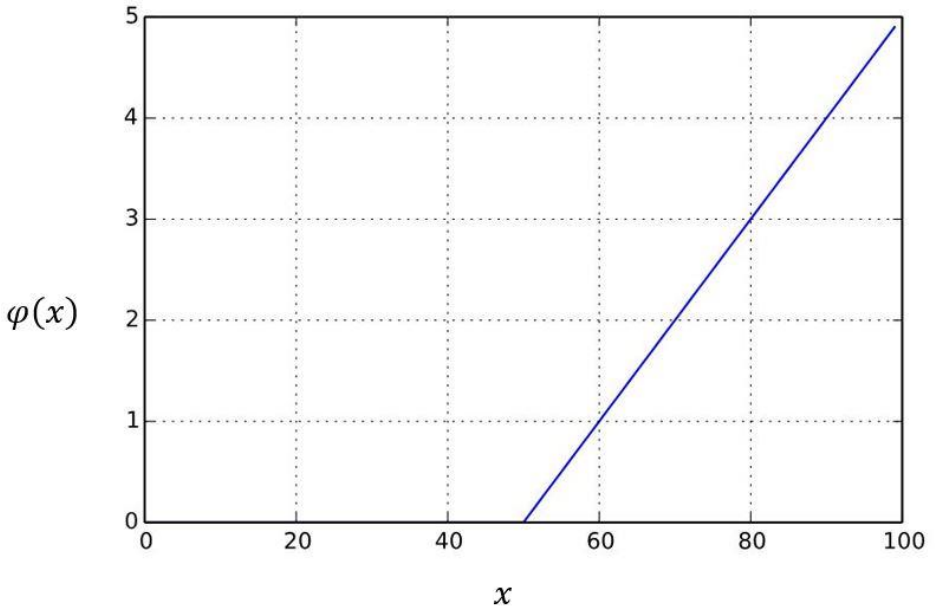

Figure 3.5. Graphical representation of the ReLU function

Which activation function to select for an ANN is not always straightforward. It depends on the type of problem at hand, and sometimes there is the need to combine 2 or 3 different activation functions in order to solve a problem.

An ANN works by taking in input values also known as input vector into the input layer first, then each neuron in the hidden layer (can be one layer or multiple layers) receives all the values from the input layer through the weights and in addition with the bias performs the calculation depending on the function selected as a neuron and produces output results. The next layer takes the output results of the previous layer as its input value and once again performs the calculation according to the activation function and produce output results. Depending on the number of hidden layers this continues until the final output layer where the values are used to solve the problem. This is called a feedforward neural network. In the section to come, we will see another method called back propagation where the process starts from the output layer and works its way back to the input layer. For simple classification problems the sigmoid or logistic function is preferred and for complex problems where accuracy is of paramount importance, a ReLU is preferred. The details on how the ReLU activation function works are presented in the coming sections.

ANNs are often regarded as 'black boxes' due to the fact that we don't fully understand exactly what is going on inside the hidden layers. Aside from the fact that we know how each individual activation function operates, it's very complex to understand how they work if we combine thousands or millions of neurons to perform together and produce the desired result.

In general, when designing neural networks to solve a problem, the number of layers to use, the number of neurons to use and the type or types of activation functions to use are the most important decisions to make. Every problem is different hence, there is not an exact network architecture that will fit all problems or even two similar problems. After the neural network architecture is determined, a good strategy is needed to train the network. The next subsection is devoted to how to train ANNs.

3.1.3 ANN training

First of all, it is important to state that training an ANN to function properly requires a lot of data. Without enough data, it is impossible to train ANNs to accurately solve problems.

The training begins by feeding the neural network with known and labeled data (sample data). In order words the expected results are known, for example, several images containing dogs and cats. After passing the data into the neural network it generates results called actual results at the output of the network. The second step will calculate the difference between the expected result and the actual result by using a loss function sometimes known as cost function. The purpose of the loss function is to find the error between the two results. After getting the error it becomes an optimization problem, the objective is to minimize the error to the lowest minimum possible by using an optimization algorithm. The most widely used algorithm for training neural networks is the backpropagation method which calculates by using the gradient descent or the stochastic gradient descent algorithms. The same sample data is fed into the neural network, a result is gotten at the output and the error is calculated. The algorithms first initialize the weights of the neural network and start to adjust the weights in a way that will slowly reduce the error until it becomes the lowest possible. It does so by calculating the gradient of each neuron and finding the direction of the gradient then updating the weights with respect to the gradient direction step by step. When all the weights are updated at the same time it is called batch gradient descent. On the other hand, when the weights are updated one at a time the process is called stochastic gradient. The process described is a summary of the general approach of training ANNs which may be called feedforward propagation.

The backpropagation algorithm is the fastest algorithm that makes it possible to train a neural network. Without this algorithm it was almost impossible to train neural networks in the past. This reference [42] by Geoffrey Hinton et al. shows the first time the backpropagation succeeded in training neural networks in a relatively shorter time. The purpose of the backpropagation algorithm is to compute the partial derivatives $\frac{\partial C}{\partial w}$ and $\frac{\partial C}{\partial b}$ of the loss function c with respect to any weight w or bias b in the ANN. It computes the partial derivatives from the output layer back towards the input layer

using the calculus chain rule, that is where the name comes from. The loss or cost function could be a quadratic function such as:

$$c = \frac{1}{2n}\sum_{x} \|y(x) - a^{L}(x)\|^{2} \tag{3.6}$$

n is the total number of training samples; the sum is over individual training samples, x; $y = y(x)$ is the corresponding desired output. The number of layers in the network is denoted by L.

With this method, the gradients are computed and the directions are known. Next, there are some assumptions that must be made for the training to work. Such as selecting a learning rate δ which is not too large or small and also the weights have to be first initialized before updating begins. This process is repeated several times maybe thousands or even millions of time until the optimization algorithm converges at the global minimum. Once all the weights are fine-tuned to give a near zero error, the neural network is considered trained. At that stage, it can solve the problem with a very high accuracy.

There are other algorithms proposed for training ANNs however, the backpropagation algorithm has proven to be the most successful and widely used one among them. Other neural network architectures such as the RNN and Radial Basis Function network (RBF) have been trained by variations of the backpropagation algorithm.

In this subsection, we presented a brief description of how an ANN is trained by the most widely used training algorithm. In the next subsection, an introduction to DL is presented.

3.2 Deep Learning (DL)

DL is a branch of ML which is based on ANNs. We based our description of DL on the CV problem because this thesis is about face recognition which is a computer vision problem. As explained in the literature above there is the need to extract features from the face on the image, the same applies to object recognition from an image in general. Because there are so many features on a face or any object for that matter when trying to recognize it from an image or video it is always difficult and time consuming to extract the unique features manually. Because of this, there have been

many algorithms developed to extract unique features of an object during detection or recognition. As humans it is easy for as to detect and recognize objects such as car, cat, dog etc. because we can identify objects by high-level features such as a car has 4 wheels, made by metal, has windscreen etc. For a computer, these high-level features are really difficult to make sense of on an image. So traditional coding methods cannot solve these kinds of problems efficiently. This is where DL comes in, DL is able to extract this high-level features from images and videos regardless of the variations. In simple terms DL is just neural networks with many more hidden layers, they can be hundred, thousands or millions of hidden layers.

DL has the ability to make an abstraction. They can build complex concepts from simple concepts after feeding an image through the network. When an image is fed into the DL network it can learn objects such as faces, cars, tables etc. by combining simple features like the edges, the shape, and corners. As the image passes through the layers, one by one each layer learn a simple feature before it goes to the next layer as an input, however, as the layers increase the network is able to learn even more complex features and in the end combine them to predict what is on the image. The name DL comes from the capability of the deeper layers to extract high-level features. Hence, the deeper the layers the better, Figure 3.6 gives a graphical example of how DL works.

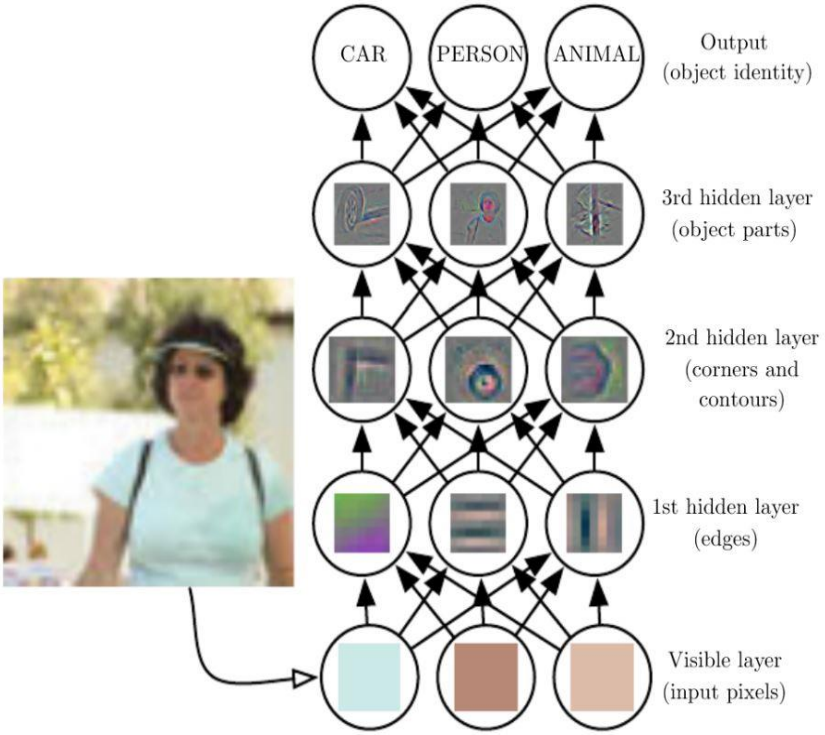

Figure 3.6. An example of how deep neural network learn features from each layer until it is able to recognize more complex abstractions [43].

3.2.1 Deep Neural Networks (DNNs)

The most popular DNNs are based on ANNs as explained above, the structure of DNN is arranged such that it has many hidden layers than the normal neural network. How many layers to include in the hidden layer depends on the problem under consideration. The multilayer perceptron (MLP) which has at least three hidden layers and trained by supervised learning, is the most widely used neural network in the DNN. Because it has many hidden layers the number of weights and neurons will be large in orders of magnitude. Due to this complex structure and nodes, the DNN requires a large amount of data and very high computational power to train. As we will see in chapter 4 this is the main reason why we could not train our DNN from scratch, we lack the proper computational resources and the required magnitude of data to build and train our own network. ANNs and the backpropagation method has been around for decades. However, before the internet and social media explosion in the late 2000's it was difficult to get large enough training data, and equally, there were not enough powerful computers available to researchers to train ANNs properly. The good news came in the late 2000's precisely in the year 2006 when Geoffrey Hinton et al. [42] published a paper which shows how the backpropagation method was able to train an ANN by considering each layer as a Restricted Boltzmann machine, and training 1 by 1 in good time and with significant accuracy after the training. Although there were other works on the training of ANNs until that year this was the first time in a long time that there was a renewed interest in ANNs and their applications. In recent years the emergence of graphical processing units GPUs and big data has spurred the development of more interesting training methods and networks. The field ANN and DNN have attracted a lot of attention from the media and researchers lately because of the success these systems are able to achieve, there are others who even believe DNN has brought close to achieving human-level AI machines.

3.2.2 Convolutional Neural Networks (CNNs)

CNN [44] is a unique type of DNN which also uses the architecture of the multilayer perceptron. CNNs are used for recognizing objects on images and it's used for solving problems in CV mostly. CNNs mimics how the visual cortex of animals and humans work. CNNs have over the years achieved great success in practical applications related to CV and image analysis. CNNs got the name 'convolutional' because the network uses a linear operation in mathematics called convolution. In place of the usual matrix multiplication, CNNs rather use the convolution operation in one or more of their layers.

It has many different layers which work together to recognize an object on an image, Figure 3.7 shows the structure of CNN, the first and second stack of layers has 6 layers each and the third stack has 16 layers and so on. It works by taking in an image, in the first hidden layer the neurons are able to learn simple patterns such as edges, shape of ears, or nose and then, the image is forwarded to the next layer which learns more pronounced features as compared to the ones in the first layer. As the layers increase the neurons begin to learn high-level features such as the whole face or a car. CNNs are very good at recognizing complex patterns in images. The principle of how CNN works is slightly different from the standard ANN. Each neuron in a layer concentrates on only a small portion of the image, one neuron looks for the same feature and leave the rest of the neurons to also look for different features in another portion of the image under consideration.

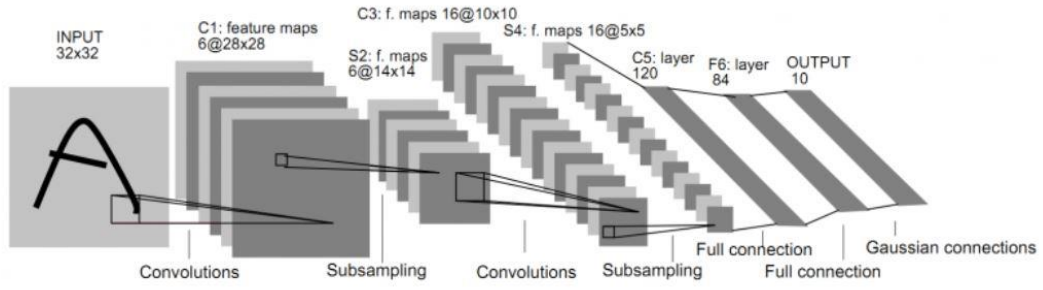

Figure 3.7. Structure of a CNN. The images reduces in size until it reaches the fully connected layer [45].

Like the MLP, CNN is also trained with the backpropagation method using the gradient descent algorithm. CNNs are arranged in a repetition of 4 major layers, the most used layers are the convolutional layer, pooling layer, and the fully connected layer.

Convolutional layer: The convolutional layer is the first hidden layer of CNN. In this layer, one image becomes a stack of filtered images. Convolutional layer general reduces the number of weights in the network so that they can be trained faster. A filter sometimes known as a kernel which is a set of weights is convolved with the input image to generate a smaller set of weights and pass it to the next layer.

The pooling layer: The pooling layer normally comes after the convolution layer. Its purpose is to reduce or shrink the image stacks and parameters gotten from the previous layer. It takes a window of size 2 or 3 and a stride of 2, and move it across the filtered images from the convolution layers. At each pass, the maximum value is retained while discarding the rest of the values within the window. The results of this will give smaller or reduced image which is usually about 75% of the original filtered image. At the pooling, the layer does not have weights to train it only uses the parameters of the stride and the window for training purposes. The smaller images reduce the time for training the network and also the likelihood of overfitting.

The ReLU layer: The ReLU layer is sometimes the second or fourth layer in the CNN architecture, its purpose is to normalize the pixels of the image by changing all negative values to zeros and retaining all positive values.

Fully Connected Layer: This layer is always at the end of the CNN even if there is a repetition of the other layers described above. In this layer, all the neurons are connected to all neurons of the previous layer similar to the standard neural network. It can be used as the output layer for CNN.

CHAPTER 4

DESIGN AND METHODOLOGY

The main design consideration is to develop a system by using a pre-trained CNN architecture which requires a small dataset to test while giving high accuracy and real-time execution. We have conducted a thorough review of the most effective face recognition methodologies as presented in Chapter 2. In this chapter, our goal is to give a detail description of the design and implementation of the methods and algorithms chosen for building our system.

Face recognition is a combination of different problems, hence, we had to develop the system in a step by step manner. We have a pipeline and at every stage of the pipeline we solved one problem and passed the results to the next stage of the pipeline until all the sub-problems are solved.

In this work, our aim is to develop a real-time face recognition system using DL based methodologies. The main features of our desired system are that the face recognition methodology chosen should be pre-trained. This is because we do not have the necessary computational resources and dataset large enough to train the CNN on our own. It should also be a lightened version, in other words, we should be able to implement it successfully on a typical computer processor and an inexpensive embedded system. After our review, we selected the best methods and techniques for the development of our system. Our system consists of four main stages, the detection stage, followed by the pose and projection of the face, generation of face embedding's, then the last stage is the classification of whose face it is. Figure 4.1 depicts a flowchart of the general structure of our system.

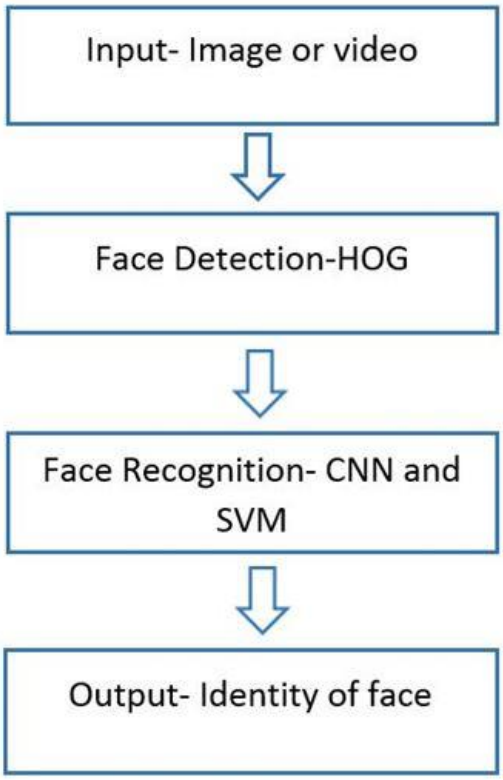

Figure 4.1. General structure of our system

4.1 Face detection

After acquiring the images or frames of a video the first stage in our pipeline is to search and locate all the faces in the image. We can do this by implementing so many different methods and algorithms in the literature. During our review we found that one of the best face detection methods is the standard Viola-Jones algorithm [46], initially, we intended to use this algorithm because it can detect faces in real-time. However, after further research, we found that it was not as reliable as we would have liked, because when the faces were rotated or in an awkward position or poor lighting conditions the algorithm seems to fail to detect the faces. Other methods were equally investigated in detail however, the most appropriate method for this project was the Histogram of Oriented Gradient (HOG) algorithm invented by Dalal and Triggs in 2005 [47]. This method is faster and accurate under so many different conditions such as illumination, scaling and shadowing as compared to the Viola-Jones detector [48]. There is also a neural network version of the HOG called CNN-HOG which is more

accurate than the original HOG detector however, we did not apply the CNN-HOG detector because it requires a GPU to run in real-time in order to take advantage of the parallel processing of the GPU. When we ran this detector on a general purpose CPU it was rather slower than the original HOG even though the accuracy was still high. As stated above our main objective is to develop a system that has an accuracy close to state-of-the-art systems and can run in real-time. Hence, we applied the original HOG detector in this project.

This paragraph presents the details of the methodology of the HOG detector. The HOG detector uses a combination of DL and image processing methods. The concept of HOG is based on the static feature extraction method. At the first stage, we have to preprocess the image we acquired by resizing it to 64x128 pixels and converting it to black and white if the original image is a color image. In paper [47] Dalal and Triggs also used the gamma correction technique to further preprocess the images, however, there was not any significant improvement over the images without the gamma correction technique. After preprocessing the image, the second step is to start calculating the HOG descriptor of the 64x128 image. There are different methods to calculate the horizontal and vertical gradients of the image. The Sobel operator is one of the most effective methods for calculating these gradients. We then find the direction and magnitude of gradients by Equations 4.1 and 4.2:

$$g = \sqrt{g_x^2 + g_y^2} \tag{4.1}$$

$$\theta = \arctan\left(\frac{gy}{gx}\right) \tag{4.2}$$

Every single pixel is replaced by a gradient with direction and magnitude. This stage eliminates a lot of information that is not useful to detect a face leaving only the outlines (major features) of the image, Figure 4.2 shows an image processed using the HOG. The direction of the gradients is pointing from lighter regions to the darker regions of the image. Once the whole image is represented by gradients pointing to the darker regions we move to the next step. In the third step, the 64x128 image is divided into cells of 16x16 images with 50% overlap in order to simplify the process, which is further divided into 2x2 cells of size 8x8. This division makes the representation

compact and robust to noise. Then the gradient orientations are quantized into 9 bins (0 to 180 degrees) histograms. Then the final stage is to concatenate the histograms to get the final feature vectors. This process, in the end, simplified the original image into a simple feature vector. This represents the basic structure of a face. We then repeat this process with thousands of images with faces to train the HOG feature descriptor. Once it is fully trained we can use it to generate the new features of our new image with a face and then use a linear Support Vector Machine (SVM) to classify whether there is a face or not. Figure 4.3 illustrates 2 faces detected from an image by the HOG method. After the face is detected it is passed to the next stage in the pipeline. This algorithm was built into the dlib [32] open source library invented by Devis King.

Figure 4.2. Representation of the major features of an image by the HOG method [48].

Figure 4.3. The faces on the center and left are detected by the HOG method from the image on the right.

4.2 Face posing and projecting

The next main stage of the system is posing and projecting faces. This stage is important because sometimes the faces may not always be in the same position or direction when the image was taken and the computer may see faces in different directions and positions as faces of different people. We need to warp all the images in order for the eyes and lips to be in the same sample position regardless of their original directions. Centralizing the faces is crucial for the next stage in our pipeline. Our literature review shows that there are many ways we can do this, but we applied a modified version of the face landmark estimation algorithm [49] developed by Vahid Kazemi and Josephine Sullivan in 2014. This method was chosen because it is really fast and is included in the dlib open source library hence, no license infringement issues. The methodology used in the original paper was that 68 specific points known as face landmarks will be generated from the face on an image, mostly the outside edges of both eyes, the top of the chin, the inner edges of both eyebrows, the nose etc. After which a machine learning algorithm is trained to learn these 68 landmarks so that when a new image with a face is passed through the algorithm it can automatically generate a new set of 68 landmarks. Once the eyes, mouth, nose, and cheeks are located on the image simple 2D image processing transformations such as shear, rotate, scale was used to center the face properly without any distortions, this transformation is called affine transformation illustrated in Figure 4.4. 3D transformations [50] although fancy, have the tendency to introduce distortions to the image so 3D transforms are avoided in this method. After the algorithm is successfully trained we can now roughly position the mouth, eyes, and nose in the same position regardless of the direction of the face. This will make the face recognition in later stages more accurate. However, the modified and improved version we have applied in our project generates only 5 face landmarks [36] instead of 68 as indicated in the original paper. From 68 face landmarks to only 5 has significantly reduced the execution time making the modified version even faster than the original one. This modified version is only recently included in the latest version of dlib 19.9.0 library.

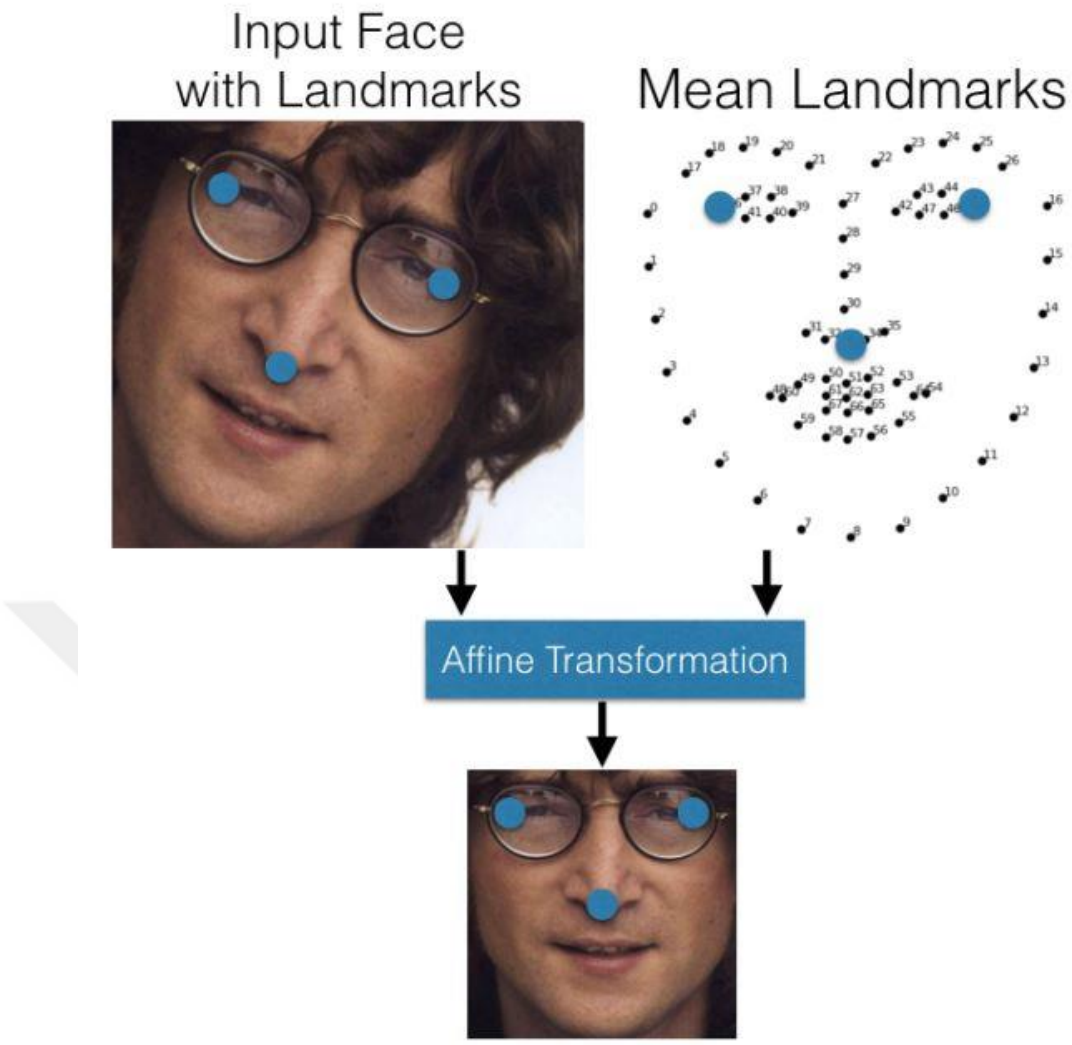

Figure 4.4. Demonstration of affin transformation of a face [31].

4.3 Face recognition

After the face has been detected and positioned we are now ready to begin the DL based face recognition process itself. State-of-the-art face recognition is currently limited to industrial giants such as Google, Facebook, Baidu and government research agencies. They have developed large-scale datasets which are not yet made available to the public and the general research community. Their face recognition systems are also built by very expensive and large computational resources that cannot be accessed by the public. The current best performing state-of-the-art face recognition model is FaceNet and its variations developed by Google in 2015 with an accuracy of 99.63% [6] on the LFW. DeepFace with an accuracy of 97.35% [7] on the LFW developed by

Facebook comes second. Although FaceNet is the best performing model out there, it is beyond our reach to implement this architecture, since it required 8 million different identities and between 100 to 200 million images to train and expensive computational resources. Due to this, the best models outside industry and government research institutions were used. Our review found that OpenFace [31] developed by B. Amos and face_recognition developed by Adam Geitgey are the best performing mobile DL based face recognition methods that are pre-trained and open source. Both models are modified and lightened versions of previous advanced models of Google and Facebook. Further research shows that Face_recognition has a higher accuracy and speed than the OpenFace method. Hence, we decided to implement the face_recognition model in our face recognition system because it is pre-trained and requires standard computational resources to perform close to the state-of-the-art systems.

This paragraph gives the details of how face_recognition method works. It is an open source library developed by Adam Geitgey in 2017. Face_recognition was built with dlib's state-of-the-art face recognition developed by Devis King. This model is based on a pruned down version of the model in this reference [37]. It was trained with 3 million images and has an accuracy of 99.38% [35] on the LFW. The convolutional layers have been reduced to 29 layers from 34 layers and the filters in each layer were reduced by 50%. This model has been pre-trained to map a face to 128-dimensional vector space, so when we passed an image through the network it generates 128 measurements known as embeddings for the face. Figure 4.5 illustrates the 128 embeddings generated from a face.

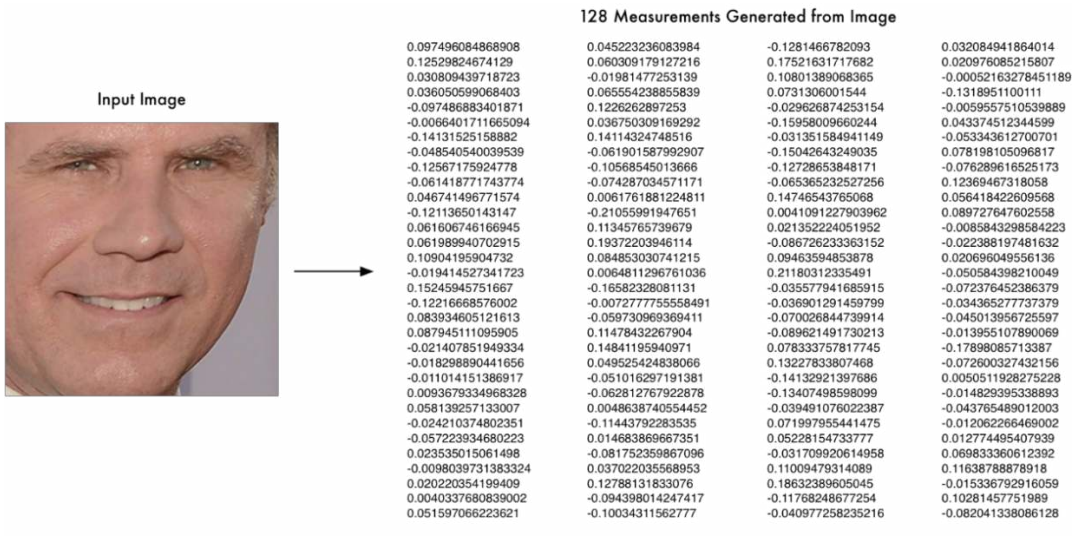

Figure 4.5. Iluustration of 128 measurements generated from an image [48].

We trained the deep neural network by loading an image of a person into the neural network then followed by loading a different image of the same person to pass through the neural and finally an image of a totally different person is loaded into the network. Then the algorithm generates 128-measurements called embeddings of all the 3 images. After which it compares the measurements of the first 2 images of the same person and tweak the parameters of the neural network so that the 2 measurements closer to each other, at the same time making sure that the measurements of the second image and the third are further apart, this process is called triplet loss training [6] because the neural network uses triplet based loss function. Figure 4.6 presents an illustration of the triplet loss process. We then repeat the training process for millions of images for different identities, once the neural network is fully trained to generate the 128 measurements of all the images it can also generate a new set of 128 measurements of any image whether known or unknown in the dataset. At the end of the training process, and 10 different images of a person should give approximately the same 128 measurements for all the 10 different images. As mentioned before, this deep neural network training requires days of training even with an expensive GPU such as the NVidia Tesla and a huge amount of data. We do not have such resources at our disposal, thankfully this face recognition method we have applied is already pre-trained and open source by Davis King so we do not need to train it again.

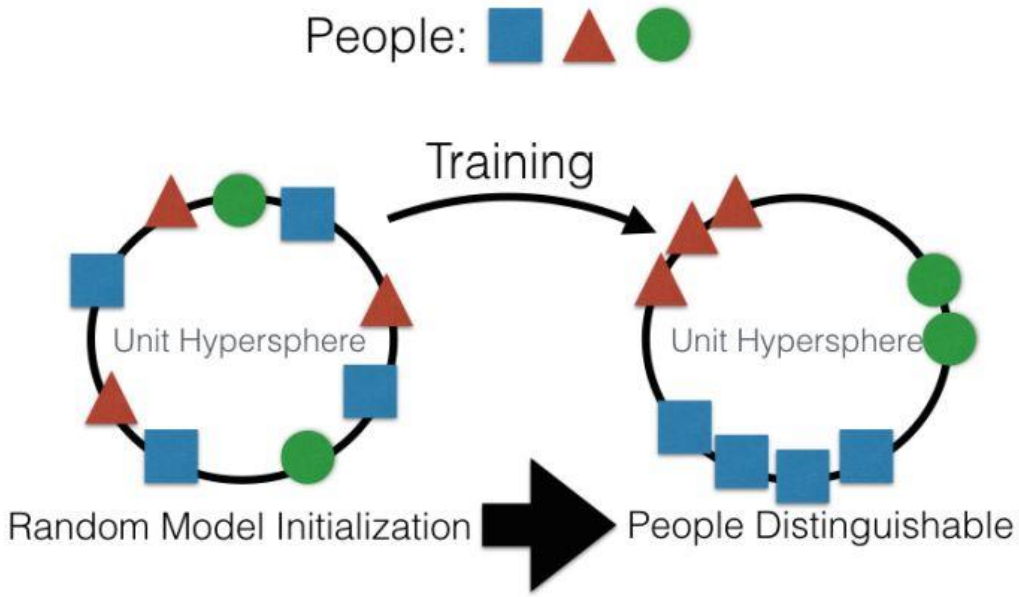

Figure 4.6. Demostration of FaceNet's triplet-loss training process [31].

4.4 Face classification

This is the final stage of our face recognition pipeline. Since the method applied is already trained to generate the 128 measurements of any image the system is ready to tell apart whose face the 128 measurement belongs to if we load a new image into the system. At this stage we deploy many simple machine learning classification algorithms. However, our research found that linear SVM classification algorithm is best for our project because it is fast, accurate and easy to implement. We trained the classifier to take in new 128 measurements and compare it to the 128 measurements in the dataset. It calculates the Euclidean distance between the new measurement and all the measurements in the dataset. If it is less than the default threshold of 0.6 [36] for any pair, then it is a match otherwise it is not a match, a name is printed out if the hit is labeled in the dataset otherwise it will print out unknown.

CHAPTER 5

EXPERIMENTS, RESULTS & DISCUSSION

Our face recognition system is an integrated system made up of hardware, software, and the best deep learning mobile face recognition method. As presented in chapter 2, we conducted a literature review of the best-performing face recognition methods and summarized them, their strengths and drawbacks. Then we came to a conclusion on which methods and techniques can be customized and use for the development of our new system.

In this chapter, we presented the experiments we carried out after a complete integration of our system. Furthermore, we tested our system and presented the results. The pre-trained model we used in our system can be trained with at least a single image of a person [35]. For a better evaluation of our system, we a created our own dataset for both the training and testing.

5.1 System hardware

We selected two different hardware to implement our system. This is because our objective in this project is to build a complete and real-time face recognition system on an inexpensive hardware. Since we did not know whether our selected DL based methods will be able to run in real-time and with high accuracy on a typical Dell laptop with Intel Core i7 processor or better yet an embedded system such as the Raspberry Pi 3, the system was tested on both hardware. In all our experiments and testing of our system, we used an 8-core Intel Core i7 processor Dell laptop after which we repeated the same process on an inexpensive embedded system (Raspberry Pi 3 model B) with an ARM v8 processor.

The face detection stage and all other stages except the embedding stage made use of only the CPU cores. The GPU of the laptop was not used because our aim as already

stated above is to develop a system that can run in with very minimal computational resources. We also use the webcam on the laptop to capture the images and videos for testing our system on the standard laptop. However, when it came to the to the Raspberry Pi we used an external camera called Raspberry Pi camera V2.1 to capture the images for testing on the Raspberry platform.

5.1.1 Dell Laptop with Intel core i7 processor

Figure 5.1 presents the first hardware platform used in this thesis project for both writing and running the code. It is a typical Dell Latitude E6530 model with the specifications below:

- Intel (R) 8- Core (TM) i7 – 3630QM CPU @ 2.40GHz
- 8.0 GB RAM
- NVidia NVS5200M GPU card
- On-board Intel® HD Graphics card

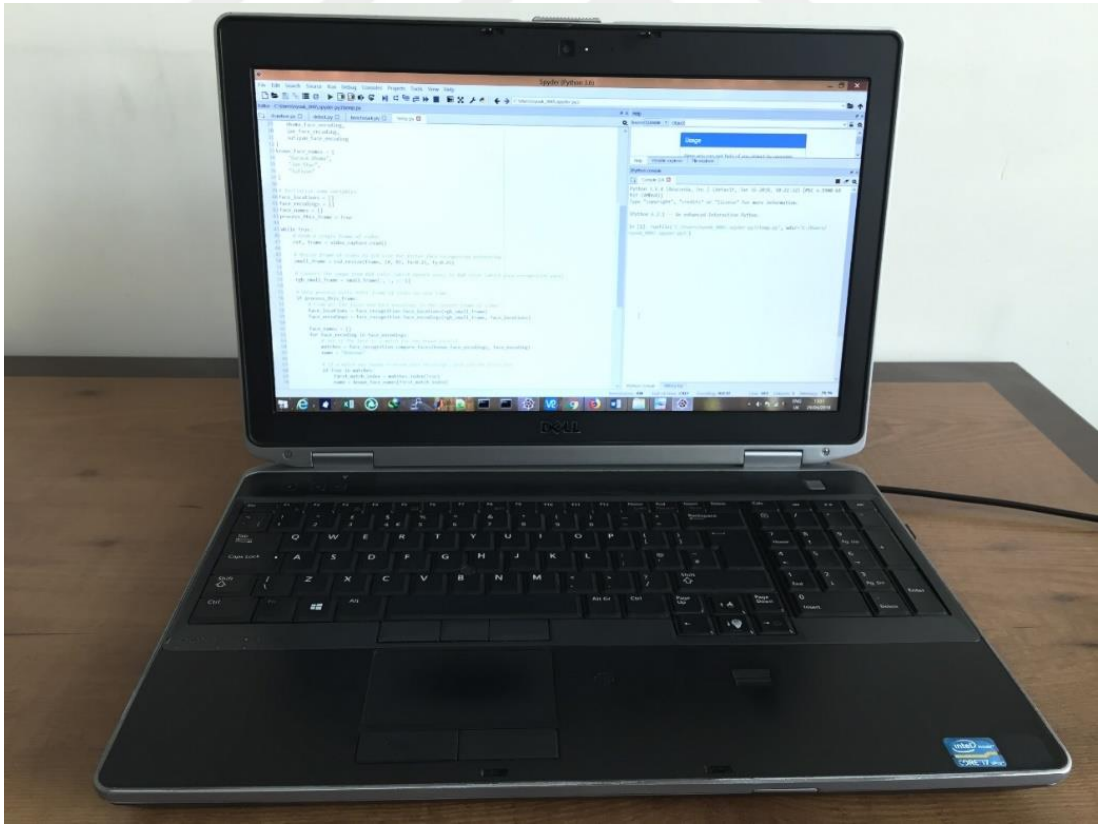

Figure 5.1. Dell Latitude E6530 Laptop

5.1.2 Embedded system (Raspberry Pi 3 model B)

Figure 5.2 presents the Raspberry Pi 3 B which we used as our second hardware platform. The Raspberry Pi 3 B has the following specifications:

- 4- Core ARM Cortex-A53 CPU @ 1.2GHz
- 1GB LPDDR2 @ 900 MHz RAM
- Broadcom Video Core IV GPU
- Broadcom BCM2837 SoC
- 10/100 Ethernet @ 2.4GHz 802.11n wireless Networking
- MicroSD
- HDMI, Ethernet, Camera Serial Interface (CSI) ports

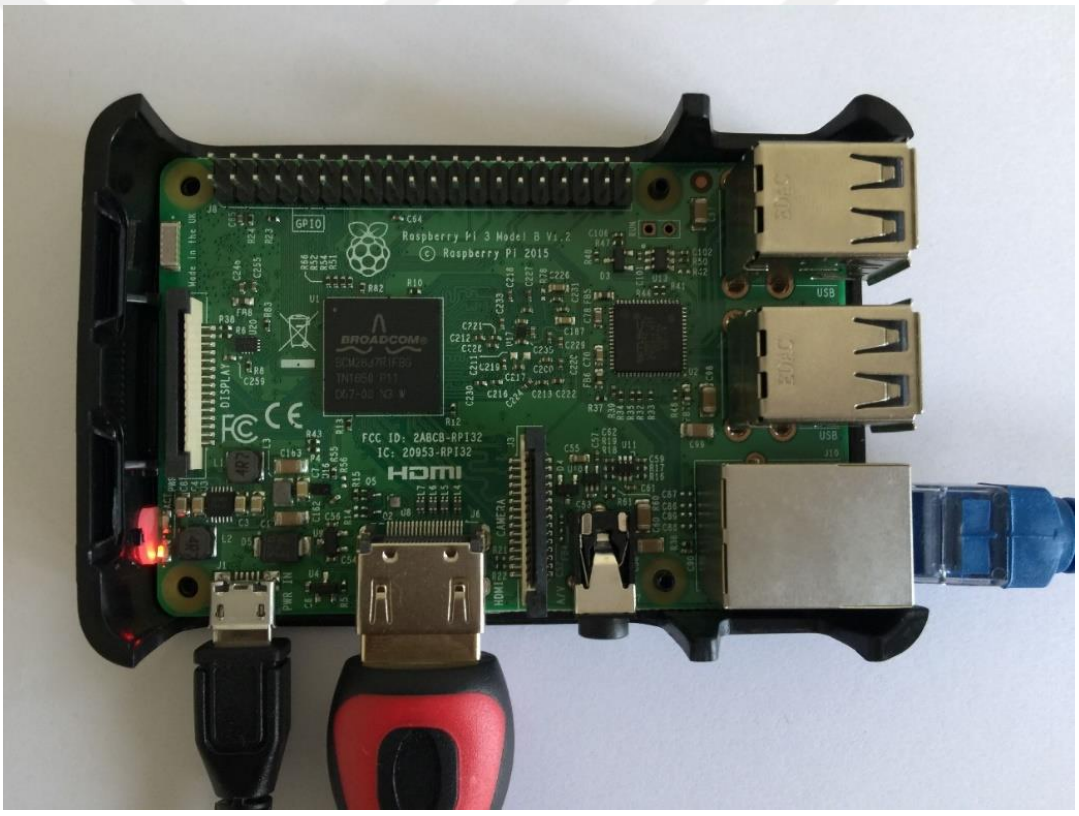

Figure 5.2. Raspberry Pi 3 B

5.1.3 Raspberry Pi camera V2.1

Figure 5.3 shows the Raspberry Pi camera model V2.1 which we used to capture both images and live video for testing our system on the Raspberry Pi 3 B. It is an upgrade

of the model V2.0 which can easily be integrated into the Raspberry Pi via a short ribbon cable. Below are its specifications:

- 8- megapixel resolution sensor-capable of 3280 x 2464 pixel static images
- Supports 1080p30, 720p60 and 640x480p90 video
- Optical size of 1/4"
- On-board fixed focus lens
- 1080p30, 720p60 and 640x480p90 video modes are all supported
- 25mm x 23mm x 9mm size
- Has a Sony IMX219PQ image sensor for fast video imaging

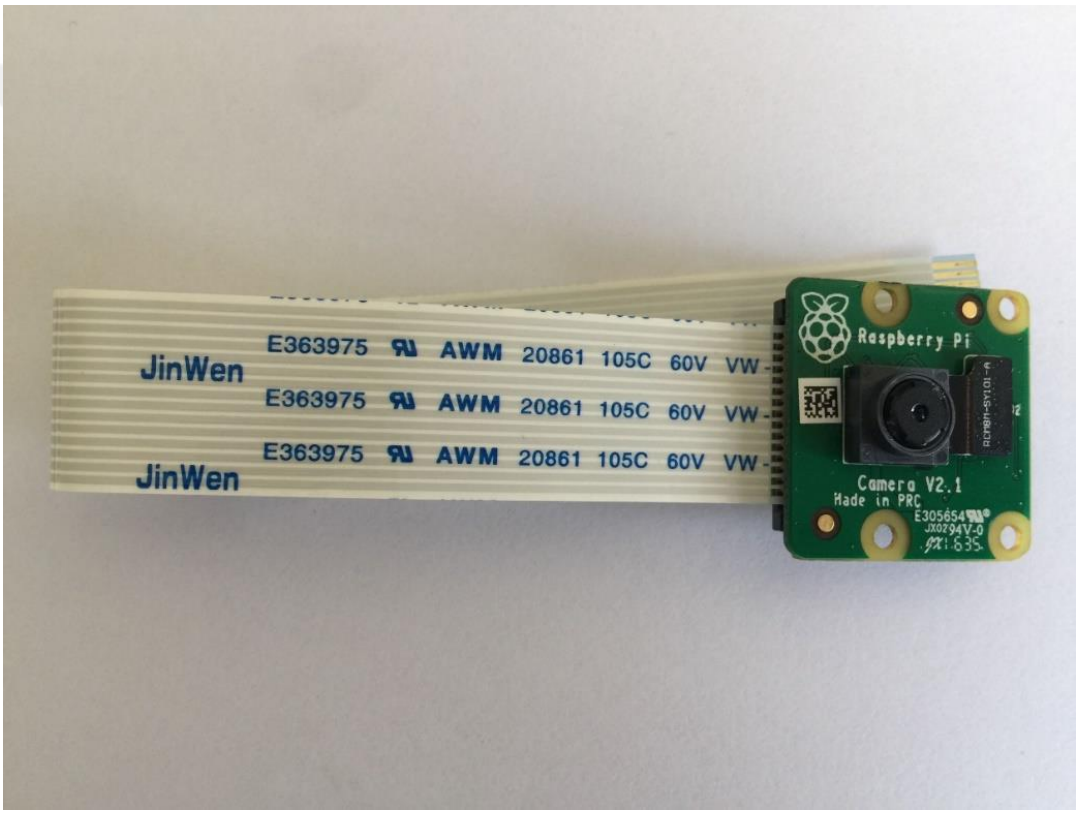

Figure 5.3. Raspberry Pi camera model V2.1

5.2 System Software

We implemented our face recognition methods with Python 3.6 and its libraries such as numpy [51] for algebra and array operations, and scikit-learn [34] for classification. We also used OpenCV's [33] library cv2 for reading from our webcam during testing.

On the laptop, the Spyder (Python 3.6), an open source software was used for writing and testing all the code for our project. While the integrated Python 3.5 that comes with the raspberry pi operating system raspbian was used to test the performance of our system on the Raspberry Pi 3B platform.

5.3 Datasets

As indicated in chapter 4 the face recognition model used is a pre-trained neural network so we needed at least one image each of the persons in our dataset to train the CNN. However, we need many more identities and images to evaluate the performance of our system under different conditions. Our dataset is made up of a total 40 images with 5 different images of 8 people.

To build the dataset without facing privacy issues and also because there are many images of celebrities on the internet, we used identities of popular politicians and actors. We also consulted all persons who are not celebrities that we added in our dataset, these people are Emile Njodzefon Wirsiy and Fatima Yakubu including myself. For the celebrities involved, we extracted their images from the Small Face Recognition dataset [52]. We collected 5 images of each person and at least one of the images was used by the CNN for generating the 128 embeddings while the other 4 images are used for testing and evaluating the system on both platforms. 5 images of each person were enough to train and test our system with different poses for both static images and video applications.

We could have built a larger dataset for the testing of our systems however, 40 images were sufficient for us to train and test our system. Figure 5.4 presents the training dataset we created, made up of 1 image each of the persons included in our experiment.

Figure 5.4. Images of the training dataset for this project

For the top row the names of the persons on the images of the training dataset in Figure 5.4. starting from top left to top right are:

- Barack Obama
- Ming-Na Wen
- Emile Njodzefon Wirsiy
- Hillary Clinton

For the bottom row the names of the persons on the images of the training dataset in figure 5.4. starting from bottom left to bottom right are:

- Brad Pitt
- Sufiyan N-Yo
- Fatima Yakubu
- Neil deGrasse Tyson

After creating the training dataset we also created a larger dataset of 32 images of the same 8 people for testing and evaluating our system. The 32 test images of the 8 people are given in Figure 5.5.

Figure 5.5. Images of dataset for testing our system

The names below correspond to the people on the test images from the top row to the bottom row of Figure 5.5:

- Barack Obama
- Fatima Yakubu
- Brad Pitt

- Ming-Na Wen
- Emile Njodzefon Wirsiy
- Neil deGrasse Tyson
- Hillary Clinton
- Sufiyan N-Yo

We also trained and tested our system with random images of children from the internet but due to privacy issues, we did not put the images on this report.

5.4 Applications of our system

After the integration of the complete system, it can be used for many applications, but we presented only 2 applications that we used to test our system for results.

5.4.1 Identifying faces on a dataset of images

The main object of this application is to test the accuracy of our system when recognizing faces on still images in a created dataset. During this application, we used the dataset in Figure 5.4 to train our system first. In order words, all the faces in our dataset are known to our system after the CNN training. All the embedding of the faces on the images are generated and learned by the CNN. In the second part, we used the dataset shown in Figure 5.5 to test our system accuracy. After testing our system on the test dataset the performance was very good for the accuracy.

We tested our system with all the 32 images in our dataset and have presented some of the results in Figure 5.6. The accuracy was very high, all the images were successfully recognized except the one at the bottom right. It can be seen that even when some of the people are wearing glasses and hats or caps and with different kind of facial expressions our system was able to successfully recognize the faces. Our system was not able to recognize one of the faces of Neil deGrasse Tyson, the image on the last row and second column because of an extreme facial expression and an unusual glasses. In general, the accuracy of our system was good.

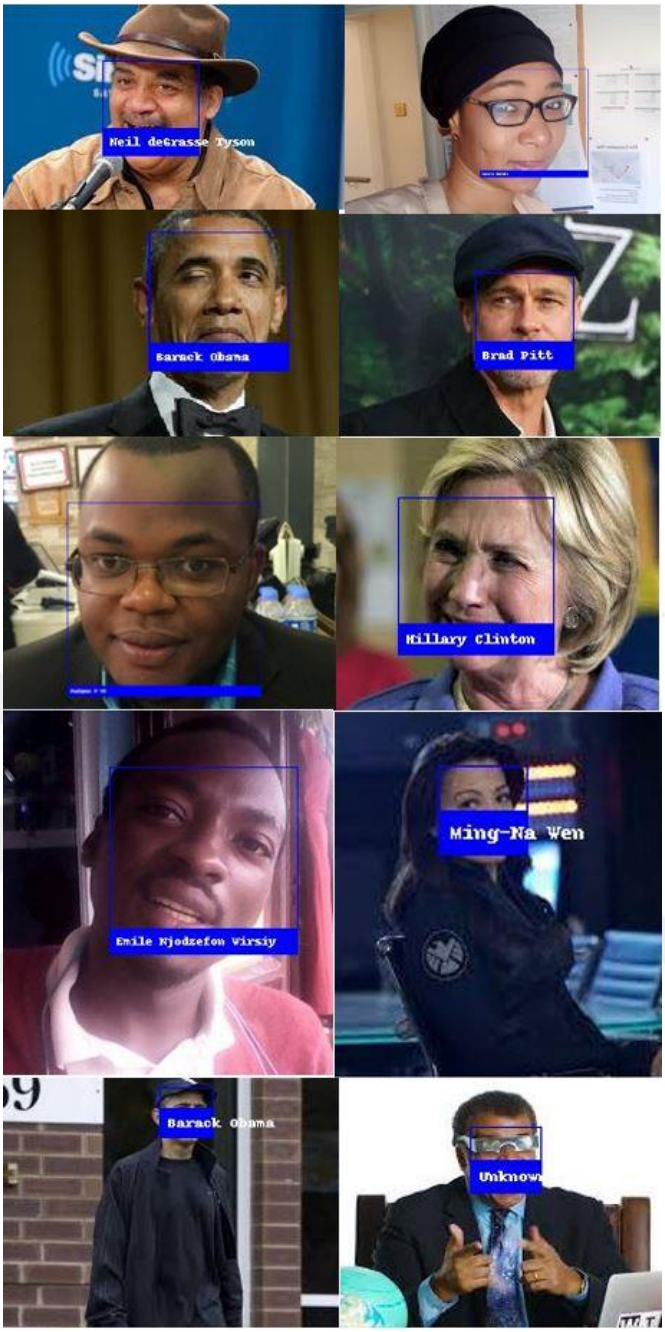

Figure 5.6. Results of face recognition on still images

We also tested the system on images containing 2 faces and Figure 5.7 shows the results of the test.

Figure 5.7. Recognition of more than 1 face on an image

5.4.2 Identifying faces on a live video from our webcam.

From the literature, we found out that most of the face recognition systems developed are tested on only still images but not videos. However, in our work, we extended the test to images in motion such as videos. With videos, it is much more difficult to recognize faces because the images are moving it easily creates blurry situations. For testing our system on videos too we used the same training dataset created in Figure 5.4 to quickly train our system. After that, we used the webcam to capture live videos for the testing on the Intel i7 core platform. We also used the Raspberry Pi camera V2.1 to capture live videos for testing them on the Raspberry Pi platform.

Figure 5.8 shows face recognition from a live video of 1 person, it draws a box around the face and identifies the face. As shown in the results, the system was able to successfully recognized the face and printed out the name.

Figure 5.8. Face recognition from a live video

In Figure 5.9 the system successfully recognized 2 faces in the video. The face of Barack Obama was had too much light at the background and from a phone yet the system was able to recognize him successfully.

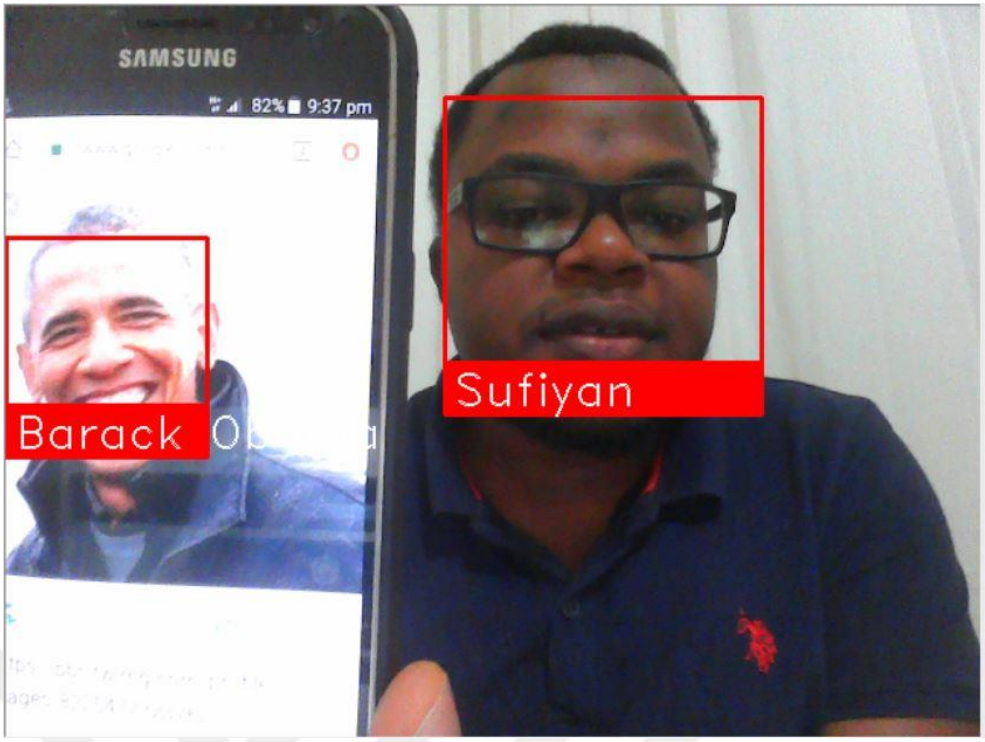

Figure 5.9. Face recognition of 2 faces from a live video

Figure 5.10. Shows the system was successful in recognizing another 2 people in front of a live camera. It is also possible to recognize more than 2 faces in a video under different conditions.

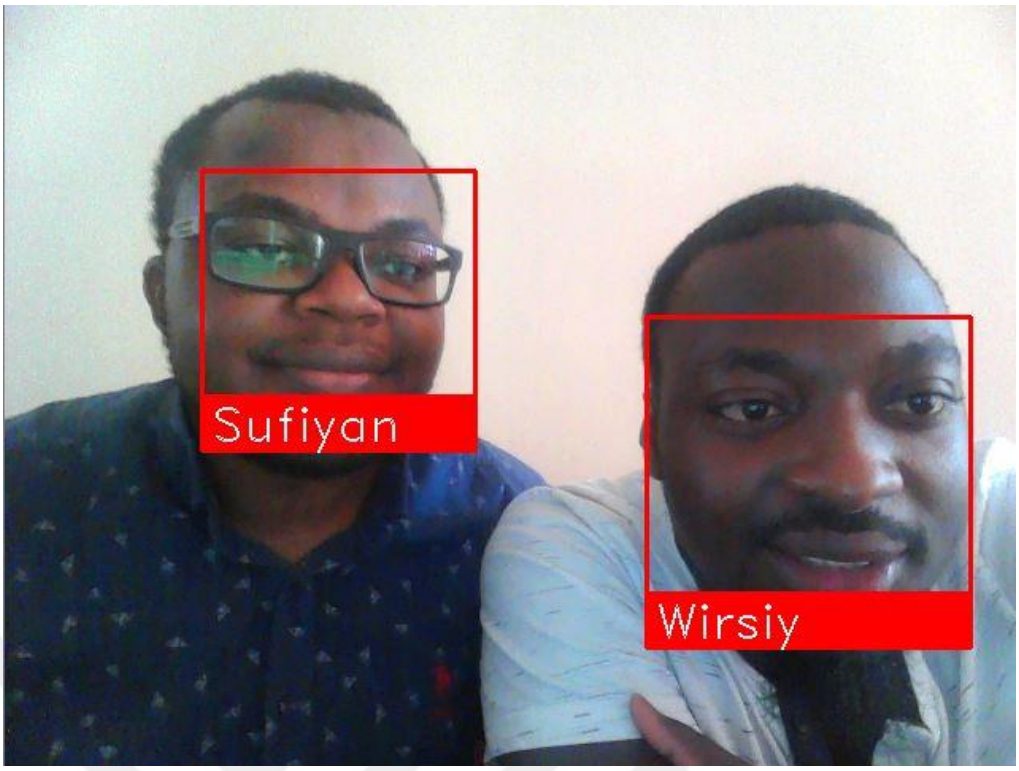

Figure 5.10. Face recognition of 2 people in front of a live camera

5.5 Results and Discussion

In this section, we provided the results obtained from both hardware platforms. After that, we compared and discussed the performance analysis of the two systems.

5.5.1 Results for test on Intel core i7 laptop and Raspberry Pi 3

5.5.1.1 System accuracy

For end-to-end real-time face recognition, we tested our system with 40 images of 8 different individuals. The test results confirmed that the face_recognition method on both systems of hardware achieved a performance of approximately 96.88% accuracy which is close to the 99.38% reported [35]. We got a slightly lower accuracy because we trained and tested our system with a relatively smaller dataset. During the experiments, we found out that the larger the dataset for both training and testing the better the accuracy of the system.

5.5.1.1.1 Comparison of results with the state-of-the-art

Table 5.1 shows the results of the recent state-of-the-art methods and the lightened versions. We used technique number 7 which has a accuracy 99.83% for developing our system. Our system achieved a relatively low accuracy of 96.88% because we tested our system with a very small dataset of 32 images as compared to 3M images of the original method used.

Comparing our system accuracy with the rest of the methods you will observe that it is not far from the state-of-the-art accuracies.

Table 5.1. Comparison of the accuracies of different face recognition techniques

No.	Technique	Images	Accuracy %
1	Human-Level [53]	-	97.53
2	DeepFace [7]	4M	97.35
3	FaceNet [6]	200M	98.87
4	FaceNet [6] + Alignment	200M	99.63
5	Face Descriptor[27]	2.6M	98.95
6	OpenFace [31]	500K	92.92
7	Face_recognition [35]	3M	99.38

5.5.1.2 System speed

As mentioned in chapter 1 our main focus is about system accuracy and speed. This subsection provided the time analysis of both hardware systems. We ran a benchmark code with Python for the time analysis of the systems.

Table 5.2 shows how long each step of the face recognition system took to complete a task while running on only one CPU Core of a typical Dell latitude laptop.

Table 5.3 shows how long each step of the face recognition system took to complete a task while running on only one ARM Core of the Raspberry Pi 3B. Units of measurement for time to process a face is in frames per seconds (fps) and image size is in pixels (p).

Table 5.2. Time Analysis of the System on a Typical Intel Core i7 Laptop

Image Size(p)	Face Locations Time		Face Landmarks Time		Encode Face Time		End-to-End Time	
	s	*fps*	*s*	*fps*	*s*	*fps*	*s*	*fps*
180	0.18	5.55	0.01	110.45	0.12	8.24	0.33	3.06
240	0.32	3.24	0.01	111.38	0.13	7.46	0.51	2.00
450	1.04	0.96	0.01	111.26	0.14	7.30	1.23	0.82
730	2.71	0.38	0.01	108.66	0.14	7.31	2.89	0.35

Table 5.3. Time Analysis of the System on Raspberry Pi 3B

Image Size(p)	Face Locations Time		Face Landmarks Time		Encode Face Time		End-to-End Time	
	s	*fps*	*s*	*fps*	*s*	*fps*	*s*	*fps*
180	2.25	0.43	0.02	46.46	0.63	1.58	2.88	0.36
240	4.12	0.24	0.02	46.14	0.63	1.61	4.74	0.20
450	14.89	0.07	0.02	45.31	0.62	1.60	15.60	0.06
730	40.00	0.02	0.02	44.91	0.62	1.60	40.30	0.02

5.5.2 Comparison of the results of the Intel core i7 and Raspberry Pi 3

As shown in both Table 5.2 and 5.3 we obtained the results of the time analysis of our system during the test for both platforms. It can be seen that the Intel Core i7 outperforms the Raspberry Pi 3 B when it comes to the time they both took to execute the various steps of our system.

Both platforms were tested with four images of different sizes ranging from 180 to 730 pixels. It can be seen in the cells highlighted green on both Table 5.2 and 5.3 that the images with the smallest pixels generally produced faster execution times as compared to larger ones. Nevertheless, there is a limit to how small this images can be, if it's too

small there is a chance of missing the face in the first place. So it has to be in a good range to get both good accuracy and fast execution time.

The test image of size 180 pixel is the smallest among the images so a test with this image gives the best results in both platforms across all four steps we conducted our experiments on. The results in the cells highlighted green in Table 5.3 shows that it took the Raspberry Pi 3 2.88 seconds (0.35 fps) for the whole process of recognition (end-to-end) while it took just 0.33s (3.06fps) for the Intel i7 core standard laptop in Table 5.2. Both Tables 5.2 and 5.3 also show a trend that the Intel i7 core performs significantly better in the rest of the 3 stages. Another point of interest in the interpretation of the results in both platforms is, the face location step. It took too long to execute this step on both platforms if the end-to-end execution time is taken into consideration. This is because we used only a single CPU core during the experiments. This can be improved by either using a GPU or tying up all the 4 CPU core processes in the Raspberry Pi3 and all the 8 CPU cores in the Intel i7 core platform. However, that is not our objective for this project. Our system was tested on the bare minimal computational resources so that we develop a real-time mobile face recognition system.

5.5.2.1 Which platform can support real-time face recognition system?

Even though both platforms had the same performance for the accuracy, our time analysis experiments show that the Intel i7 core platform performs significantly better when it comes to real-time recognition using recent DL based face recognition methods. The execution times for the Raspberry Pi 3 are lagging behind hence, cannot support real-time face recognition with the lightened version of the state-of-the-art DL based methods even after significantly reducing the image sizes.

5.6 Limitation of our system

During the test experimentations, our system shows some limitations that we have presented in this subsection. Below are the limitations we encountered in our system:

- Real-time end-to-end recognition is not possible with images of size 450p and above. This is because the time taken to execute the face location step has become significantly longer in both platforms. Hence, real-time recognition of our system is limited to images of size from 180p to 240p.

- We also found that the accuracy drops slightly when we tested our system on faces of Black women, Asian women and children and a relatively small dataset in general. The original CNN was trained on a very large dataset with images of celebrities which is bigger than our dataset by very large magnitudes.

- The system can detect and recognize faces within approximately 30 to 100cm with both the webcam and the Raspberry Pi camera V2.1. Outside this range, the system performs poorly.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

6.1 Conclusion

In this work, a detail research survey on face recognition was conducted which guided us on which method to select for this project. From the literature survey, the best methods were selected and used to develop a real-time face recognition system. With the best mixture of methods, we design and implement a deep learning based real-time face recognition system. The face_recognition method was used to face recognition, the HOG for face detection, the affin transformation for centralizing the face and SVM for classification. Two different platforms were selected for the implementation of our system however, the Intel i7 core platform was able to support real-time face recognition.

We have successfully achieved the purpose of this project which is to develop a real-time face recognition on an inexpensive platform. The system has achieved closed to the expected results for the accuracy on both platforms using a small dataset and the Intel i7 core platform achieve real-time recognition. However, it was not able to run in real time when tested on the raspberry pi 3B. Our system also performed well under various conditions such as in both bright light and low illumination environments, different facial expressions, faces with glasses etc. Hence, it is a pretty robust system considering it is implanted on an inexpensive platform. Our system has many applications including video surveillance, crime prevention, person identification and so many more.

6.2 Future work

As indicated on the section on the limitations of our system, there are some drawbacks that we found in our system. Hence, in this section, we presented some of the suggestions that can improve the system and the algorithms used during this project.

First of all, in future work, our system performance can be improved by using a more powerful camera to capture the live videos during the testing as the participant doesn't have to go as close to 30 to 100 cm distance to the camera before we can get good performance.

In addition, we propose that in the future the training dataset for the CNN should include more images for underrepresented groups such as Black and Asian women and children. These groups are not well represented in the training dataset that is why our system has slightly low performance when testing with these group of people. Also in general, just increasing the size of the dataset can improve the accuracy of the system.

Furthermore, in general, the modern lightened versions of the state-of-the-art deep learning methods can still be further reduced to even perform real-time face recognition on embedded systems such as the Raspberry Pi and even microcontroller based platforms.

REFERENCES

[1] Belhumeur, P. N. Hespanha, J. P. Kriegman, D.J. (1997). Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection‖, *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **19**, 711-720.

[2] Matthew T., Alex P. (1991). Eigenfaces for recognition, *Journal of cognitive neuroscience*, **3**, 71–86.

[3] Ahonen T., Hadid A., M. P. (2004). Face recognition with local binary patterns, *In Computer vision-eccv Springer*, 469–481.

[4] Lowe, David G. (1999). Object recognition from local scale-invariant features, *Proceedings of the International Conference on Computer Vision*, **2**, 1150–1157.

[5] Yoshua B. , Goodfellow Ian J., Aaron C. (2015). Deep learning, Book in preparation for MIT Press.

[6] Florian S. , Dmitry K. , James P. (2015). Facenet: A unified embedding for face recognition and clustering, *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 815–823.

[7] Yaniv T. , Ming Y. , Marc'Aurelio R. , Lior W. (2014). DeepFace: Closing the gap to human-level performance in face verification, *In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference*, 1701–1708.

[8] Takeo K. (1973). Picture processing system by computer complex and recognition of human faces, *Doctoral dissertation Kyoto University*, 83–97.

[9] Rabia J. and Hamid R. (2009). A survey of face recognition techniques, *JIPS*, **5**, 41–68.

[10] Brunelli R. , Poggio T.(1993). Face recognition: features versus templates, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **15** , 1042-1052.

[11] Nixon M. (1985). Eye spacing measurement for facial recognition, *in SPIE Proceedings,* 279-285.

[12] Graf H. P. , Chen T. , Petajan E. , Cosatto E. (1995). Locating faces and facial parts, *in International Workshop on Automatic Face and Gesture-Recognition*, 41-46.

[13] Reisfeld D. (1994).Generalized symmetry transforms: attentional mechanisms and face recognition, *Tel-Aviv University, PhD. Thesis, technical report*.

[14] Cox I. J. , Ghosn J. , Yianilos P. N. (1996). Feature-based face recognition using mixture-distance, *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* 209-216.

[15] Wiskott L. , Fellous J.-M. , Krüger N. , C. von der Malsburg. (1997). Face Recognition by Elastic Bunch Graph Matching, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **19** , 775-779.

[16] Baron R. J. (1981). Mechanisms of Human Facial Recognition, *International Journal of Man-Machine Studies*, **15** , 137-178.

[17] Huang J. (1998). Detection Strategies for face recognition using learning and evolution, *George Mason University, Fairfax, Virginia, Ph. D. Dissertation*.

[18] Jain A. K. , Dubes R. C. (1988). Algorithms for Clustering Data, New Jersey: Prentice-Hall.

[19] Fukunaga K. (1990). Introduction to Statistical Pattern Recognition, second ed. Boston, MA: Academic Press.

[20] Sirovich L., Kirby M. (1987). Low-dimensional Procedure for the Characterization of Human Faces, *Journal of the Optical Society of America A: Optics, Image Science, and Vision*, **4**, 519-524.

[21] Turk M. , Pentland A. (1991). Face Recognition Using Eigenfaces, *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 586-591.

[22] Turk M. , Pentland A. (1991). Eigenfaces For Recognition, *Journal of Cognitive Neuroscience,* **3** , 71-86.

[23] Moses Y. , Adini Y., Ullman S. (1994). Face recognition: the problem of compensating for changes in illumination direction, *in European Conf. Computer Vision,* 286-296.

[24] Martínez A. M. , Kak A. C. (2001). PCA versus LDA, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **23**, 228-233.

[25] Steve L. , C Lee G., Ah Chung T. , Andrew D B. (1997). Face recognition: A convolutional neural-network approach, *Neural Networks, IEEE Transactions on,* **8** ,98–113.

[26] Gary B H., Manu R., Tamara B. , Erik L. -M. (2007). Labeled faces in the wild: A database for studying face recognition in unconstrained environments, *Technical report, Technical Report 07-49, University of Massachusetts, Amherst*.

[27] Omkar M P. , Andrea V., Andrew Z. (2015). Deep face recognition, *Proceed-ings of the British Machine Vision,* **1**, 6.

[28] Xiang W. , Ran H. , Zhenan S. (2015). A lightened cnn for deep face representation, *arXiv preprint arXiv:1511.02683*.

[29] Sermanet P., Eigen D., Zhang X. , Mathieu M. , Fergus R., LeCun Y. (2013). Overfeat: Inte-grated recognition, localization and detection using convolutional networks, *arXiv preprint arXiv:1312.6229*.

[30] Szegedy C. , Liu W. , Jia Y. , Sermanet P. , Reed S. , Anguelov D. , Erhan D. , Vanhoucke V. , Rabinovich A. (2014). Going deeper with convolutions, *CoRR, abs/1409.4842*.

[31] Amos B. , Ludwiczuk B. , Satyanarayanan M. (2016). OpenFace: A general-purpose face recognition library with mobile applications, *CMU-CS-16-118, CMU School of Computer Science, Tech. Rep*.

[32] Davis E.K.(2009). Dlib-ml: A machine learning toolkit, *The Journal of Machine Learning Research,* **10** ,1755–1758.

[33] Gary B. et al. (2000). The opencv library, *Doctor Dobbs Journal,* **25** ,120–126.

[34] Fabian P. et al. (2011). Scikit-learn: Machine learning in python, *The Journal of Machine Learning Research,***12** ,2825–2830.

[35] Adam G. 2017. Ageitgey/face_recognition. [Online] Available at: https://github.com/ageitgey/face_recognition#face-recognition, 2017 [Accessed 24 Feb. 2018].

[36] Davis E.K. 2017. High Quality Face Recognition with Deep Metric Learning. [Online] Blog.dlib.net. Available at: http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html,2017 [Accessed 24 Feb. 2018].

[37] He, Kaiming, et al. (2016). Deep residual learning for image recognition, *Proceedings of the IEEE conference on computer vision and pattern recognition*.

[38] Mitchell T.(1997). Machine Learning, McGraw Hill, ISBN 0-07-042807-7 p. 2.

[39] Rosenblatt F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain, *Psychological review,* **6** ,386.

[40] McCulloch, Warren S., Walter P. (1945). A logical calculus of the ideas immanent in nervous activity, *The bulletin of mathematical biophysics,* **5** ,115-133.

[41] Kang, N. 2018. Multi-Layer Neural Networks with Sigmoid Function— Deep Learning for Rookies (2). [online] Towards Data Science. Available at: https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f [Accessed 17 Jun. 2018].

[42] Hinton, G. et al. (2006). A fast learn-ing algorithm for deep belief nets, *In: Neural computation,* **7**, 1527–1554.

[43] Ian G., Yoshua B., Aaron C. (2016). Deep Learning. http://www.deeplearningbook.org. MIT Press.

[44] LeCun Y. , et al. (1989). Backpropagation applied to handwritten zip code recognition, *Neural computation*, 541-551.

[45] Pham, Dung V. (2012). Online handwriting recognition using multi convolution neural networks, *In: Asia-Pacific Conference on Simulated Evolution and Learning. Springer*, 310–319.

[46] Viola P. , Jones M. (2004). Robust real-time object detection. *International Journal of Computer Vision,* **57**, 137–154.

[47] Dalal N. , Triggs B. (2005). Histograms of oriented gradients for human detection, *inProc. IEEE Int. Conference on Computer Vision Pattern Recognition,* 886–893.

[48] Adam G. 2017. Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning. [online] Available at: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78, [Accessed 24 Feb. 2018].

[49] Kazemi V. , Josephine S. (2014). One millisecond face alignment with an ensemble of regression trees, *In CVPR, Columbus, OH, USA ,ISSN:1063-6919., 10.1109/CVPR,* 1867-1874.

[50] Tony S. J. (1995). 3D pose estimation and normalization for face recognition, PhD thesis, McGill University.

[51] Travis E.O. (2006). A guide to NumPy, volume 1. Trelgol Publishing USA.

[52] Xavier S. (2017). Face recognition using Deep Learning, Polytechnic University of Catalonia, Barcelona, Master thesis.

[53] Neeraj K. et al. (2009). Attribute and simile classifiers for face verification, *In Computer Vision, 2009 IEEE 12th International Conference on,* 365–372.