School of Electronic Engineering and Computer Science

Final Project Report

Programme of study:
B.Eng Computer systems engineering

Project Title:

Ace Security: A face recognition security system.

Supervisor:

Dr. John Schormans

Student Name:

Akash Premdhari Pandey

Student ID: 180896615

Queen Mary
University of London

Date: 25 May 2020

Disclaimer

I thoroughly understand the nature of plagiarism and the university’s policies regarding plagiarism.

The report is a part of my final year degree project for B.Eng. in Computer systems Engineering at Queen Mary University of London. The entire project is a result of my individual work unless where indicated in the text. The work produced in my report can be used by another individual for further study and development provided that précised Acknowledgement is used.

Abstract

From last decade face recognition is been a popular topic for the research among the researchers due to its high security feature and demand of security measures for protection of data and identification authentication. It mainly consists of three steps such as face detection, extraction and recognition. Previous face recognition was based on the frontal view only but due to technological advancement, there are some algorithms which can support the side view face detection as well. Face Recognition is been used widely in organisation's entry gates, crime department to analyse the crime records, defence institutions to restrict the intruders etc. In this paper, I would introduce my project which is based on machine learning technique in order to develop a face recognition model.

Table of Contents

Table of figures

1. Introduction

Digital world is taking over the real world due to which accuracy in identifying the users and protection of data is becoming significant. In today's time corporations are facing major security issues for which they need to recruit certain professionals. These professionals are humans and human tend to make mistakes which can cost millions and billions to the company. This is where automated security system comes into picture which is more reliable than hiring professionals. Not only today for civilians but during the time of 9-11 attack government insisted to set up identification methods to ramp up the security. [1]

Traditional security measures involve username and password setup by the user which totally depends on his memory or physical objects such as key or an ID. These measures are not reliable enough as the measures related to memory and objects can be forgotten or stolen which can be a real security threat to the information saved by the user. Some Corporations have established security systems on the entry gates and office premises for attendance monitoring of employees and other surveillance tasks through PIN (Personal Identification Number) and token system such as card readers. This system is vulnerable to forgery and robbery which can corrupt the quality of user's data. To tackle this problem biometrics are now used by all the institutions to maintain the standard of security in data.[1]

Biometrics is an ability of the computer to recognise the physical unique characteristic of human. It involves using traits such as speech, iris, fingerprint, face and so on. Biometrics is dominant on the traditional measures due to the advantages attached to it such as,
Easy reach: The feature in a human being is readily available as it is attached to its body which can't be stolen or forgotten.
Lack of Imitation: The human is born with the unique traits which can't be changed and mimicked by others.
Convenient: Biometrics doesn't need co-operation of an individual.
Based on the above advantages, corporates are more interested in using Biometrics for security and safety purpose.[1] Some Prediction indicates that in in coming century biometrics will be at its peak for identity authentication and restrict unauthorised access to

networks.[1] The traditional measures are been substituted by this method in majority of institutions. With the evolution of artificial intelligence and computers the biometric security system can be incorporated within the domestic life of people as well. The examples of such domestic usages are face unlock for mobile phones, fingerprint scanner for password entry, Apple Pay etc. These uses comes with certain limitation
Such as fingers should be clean for fingerprint scanner and the software fails to identify the prints if it is injured. Out of other traits face and iris scanner are more stable at is engages physiological characteristics which can't be changed other than severe injury. Speech recognition is not an ideal option because the speech can be mimicked easily. However, in case of Iris, people are less likely to keep their eyes near to the scanner as compared to the face.

Face Recognition is one of the most compelling biometric technique used for security purpose [2]. Although, there are other methods which are more reliable such as fingerprint scanner and iris recognition, but these measures are intruding and are highly dependent on human co-operation [3]. Due to this reason, face recognition is ubiquitous, convenient and non-intrusive method. It is an appropriate solution for mass scanning which can be difficult task in other measures. Moreover, the techniques in which multiple individual has to share the same equipment to conduct biometric recognition can be unhygienic and probably people are more exposed to germs transmission. However, this problem is absent in case of face recognition. Face Recognition is an ability of computer or an equipment to recognise the user's face and provide security. A face recognition system works by recognising an image or a video of human face with existing database. The structure, shape and different elements of face is been taken into account while recognising a face. In addition to these elements distance between eyes, nose, mouth and jaw, the sides of mouth etc are also compared [4].

While using a facial recognition system, different pictures of the person from unique angles and with different expressions has to be taken (4). Image of an individual's face can be acquired by means of two ways i) Scanning the existing picture or ii) record a live picture of the subject. Face Detection is the first step in classifying the picture known or unknown from an existing database. The more reliable is the face detection process the more desired will be the result. A successful face detector should be able to classify the image in single shot irrespective of different factors such as age, position, scale and expression of the subject. Therefore, the face detection process involves differentiating the face from non-face and extract a particular face region from cluttered views. It is essential to apply Pre-processing model before image extraction because it processes the image and improve the input which lead to quality results and smoother functioning of Face Detection Model [4]. The central issue in Face detection model is feature extraction. Feature Extraction uses different algorithms to distinguish unique faces from the database [5]. Face recognition is a resource intensive method. Face recognition is bound to time constraints where real time has to be taken into consideration where using algorithms in sequence is not possible to handle large number of databases specially in case of high-resolution images. With the latest update in technology, PCs and laptops are equipped with multi-core CPUs and GPU to ramp up the speed of facial recognition on real time basis. GPU (Graphic Processing Unit) is a robust co-processor that can act as an advantage to existing CPUs. The modern GPU is able to provide multiple streaming cores and can handle thousands of threads which makes it fit to computation intensive applications. Multi-core processors are the hub where multiple execution cores are stored in a single chip. Each execution core is been served independently by the multicore processor as different functional units. With the help of parallel programming, it is possible to implement computation intensive application on set of cores parallelly which can reduce the execution time as compared to single core execution time.

If we have a deeper look in the history of facial recognition, we can conclude the fact that this algorithm has come a long way with many different approaches made to make this algorithm to work better and accurate. Starting from the 1960s with a mathematical approach which plots some facial features against the pictures, made the potential start of the facial recognition. [5] Then, in early 90s Elgen face approach was made which used linear algebra for the facial recognition. Coming down to the early 2000s, with invention of powerful GPUs recorded an exponential increase in the development of facial recognition.[5] Around 2009-2010, ImageNet has founded democratize mage data for machine learning research. Till this time period no machine learning algorithm had been used for the facial recognition. [5] As a matter of fact, when the machine learning algorithm started to enter the facial recognition models, it showed a significant improvement in recognising the face. Neural Network, a machine learning algorithm was then and even now the most and widely used technique for the facial recognition.[6]

1.1 Motivation

The Motivation behind the face recognition system built initiated in my minds when I used to work part-time in Iceland food store. Every day I walk into the store and find new case of shoplifting. Being just a regular employee at first, was easy to listen the story and let go the problem. But as time passed and I became the Duty manager of the store, I had to deal with the problem by myself. As, now I’m responsible to look out the whole store, help the security guy in dealing with the same. Because, at end the day it was a loss of the store that I’m managing. The engineer side of mine was never happy with the on-going theft problem.

That was the time when I decided to at least make an attempt to deal with the problem. I had to define my engineering capabilities. I stared case-studying about the problem and found that this was a major problem for small stores. Small cottage industries suffered due to theft. Also, they couldn’t afford such high prices to install an efficient system to deal with the problem. This made me to build a system which had to be economical but at the same time is efficient in dealing with theft problem.

2.THEORETICAL BACKGROUND:

In introduction we have seen the popularity of facial recognition system in order is to maintain the security. There has been a significant amount of research conducted by different institutions in this field. I have examined some of professional journals, books and academic papers in order to explain the background study in the field of Facial recognition.

2.1 Face Detection and Face Training:

The article *Robust Real-time Object Detection* by M.J Paul Viola is the most popular article in the series of article provided by the researcher. This article gives us detail about the different methodologies and algorithms used for face detection. [7] The *Multiview adaboost model* is an article which provides us knowledge about the application of real adaboost for object detection and also presented a more practical approach in multi-face detection framework. There has been an impressive result on the cascade structure improvements by the nest structure.[8] The article '*Tracking in Low Frame Rate Video'* is a great combination of face detection and tracking.[9] The above 3 articles state the problem related to face detection at its tracking. From the discussed articles we can make real time face detection systems. The sole purpose is to search the position and size of each face in a picture but for tracking it is significant to determine the correlation between each face in the picture.

2.2 Face Position and alignment:

In previous time, the facial characteristics only focused on two or three main points such as to detect the centre of an eyeball and centre of the mouth, but afterwards different studies have provided additional features to be considered during the facial recognition to improve the accuracy of result.[10] In the *Shape models system* researcher states a dozen of facial traits and different texture and relationship in their positions for purpose of computation. *Active Shape Models* is based on the direct relationship between the training samples and shape constraints. Facial characteristics point positioning determines the facial features on basis of detection made by the face detector. [11] The above articles present the methods for face positioning and its alignment such as SDM (Supervised descent model), LBF (Local Binary features regression, HPO, Python, Open CV etc.

2.3 Facial Traits Extraction:

The oldest algorithm used for facial recognition is PCA-based eigenfaces. Today PCA is been used to reduce the dimensions in real system than classification. The article *Gabor binary model* states different practical system involving PCA and LDA to extract the authenticated information. [12] PCA resolves the problem of LDA matrix singularity and afterwards uses LDA to extract the features for the classification. Facial feature extraction converts the face image into string of numbers which is called Face Feature. The input to the facial recognition system will be a face map and facial features coordinates and the output will be the string of numbers. Afterwards, different algorithms are used to decode such strings of numbers for the extraction of features. [12] The article mentioned above shows the evolution in this process where they stated that, earlier due to the length of algorithms face feature used to be slow and not widely accepted. However, recent studies show that the model size and operation speed can be amped up and can be available to mobile terminals as well.

2.4 Existing Facial Recognition Programs:

2.4.1 Facewatch UK:

It is a user-friendly software which provides an automated turnkey mobile and video surveillance facial recognition system. It generates an alert whenever it recognises a face and check the image in its existing database. It is more realistic and practical to use because of low resolution feature. [13]

2.4.2 Morpho Trak:

It is leading biometric company and largest innovators in the field of face recognition and fingerprint identification system. It provides security equipment to broad range of market such as border restriction, driving license, law enforcement, IT security etc.[13]

2.4.3 Cross Matching Technologies:

It provides the biometric management system, applications and related technologies to government, law enforcement agencies and corporates. The services are capable of wireless, stationary or mobile use that includes facial recognition system and other security related systems. [13]

2.5 Methods for facial recognition:

2.5.1 Eigenface:

The Eigenface is a common type of algorithm used for face recognition. Karhunen-Loeve theorem is related to Eigenface where PCA (Principal Component Analysis) is used. As discussed, earlier PCA is used to reduce the dimensionality. Eigenface are the principal components which divides the face into different vectors. The information regarding the feature vector can be obtained through covariance matrix. These vectors are been used to find the variance among the different faces. The linear combination of highest eigenvalues characterises the human face. The most appropriate Eigenface defines a M dimensional space namely face space.

It is the simplest algorithm which enables the implementation of eigenface recognition system smoother. The accuracy of this algorithm depends on different factors. This method is highly sensitive to light and position of head. Another disadvantage is finding eigenvectors and eigenvalue with the help of PCA is tedious job.[14]

2.5.2 Neural Networks:

There are several usages of Neural Network, it is used in applications such as pattern recognition, object recognition, character recognition and autonomous robot driving. The main aim of Neural Network is to train a system in order to capture complex images. It is nonlinear in nature and accepted widely for supporting facial recognition security system. According to the researchers they have achieved accuracy rate of 96.2% in the process of face recognition.[15] This research was based on 40 samples and analysing 400 images. The time taken for classification was around 0.5 second but the training process is bit lengthy that is of 4 hours.[15] The method of Gabor waveletes is been used to support the neural network for determining feature co-ordinates and extract the feature vectors. There are two more techniques which is widely used in Neural Network such as PNN (Polynomial Neural Networ) and SRKDA (Spectral Regression Kernel Discriminate Analysis). PNN is a technique in which the feature is extracted through PCA. PNN uses single network which provides more reliable results and less deviation from true results in case of more complex pictures. SRKDA was proposed in the existing model of PNN for improved facial recognition with the geometry of 3D faces. This is based on the spectral graphs and regression model. SRKDA can help recognising the face when the large sample vectors are non-linear. It is cost effective because it doesn’t involve any calculation of eigenvectors for regression.

2.5.3 Fisher faces:

R.A Fisher introduced a linear discriminant analysis for face recognition. Many researchers used LDA in their research and found out that it lacks behind because there are larger number of pixels than the images present within the single scatter matrix which can rise the error rate due to disturbed lighting and changing poses in an image. Fisher face overcome this problem because this algorithm is based on within-class information which reduces the deviation in the class and hence lighting variation within the images can be combated. Fisher face produces sub space projection with the help of PCA and linear discriminative analysis similar to Eigenface. The disadvantage related to this method is it becomes more difficult than eigenfaces to find projection space.[16]

2.5.4 Elastic BunchGraph matching:

In this method the face recognition is conducted through face graph and query image. Elastic bunch graph matching is based on Gabor weavelets. The similar features are represented in the same type of graph such as faces in identical pose and then it is converted into a bunch graph where, the objects with same structure representing local textures are put together. The similarity index is calculated through comparing the bunch graphs with the query image. This method is completely different as compared to the above three methods it uses graphs to locate the features such as eyes, nose, mouth etc. This method does have a disadvantage of changing lighting conditions and the graphs used in this technique has to be created manually on the face. When there are dynamic lighting conditions then the facial recognition process using this method can get affected. [17]

2.5.5 Geometric feature matching:

Geometric feature matching is dependent on calculating the distances between the features to find matches in large number of databases. It is based on calculating the geometric features from the image. The features can be described through vectors representing the facial feature’s size and position like eyes and eyebrows, nose and mouth and a boundary of face. This method was initially used in 1973 where the recognition rate was about 75% using the sample size of 20 people.[18] Afterwards, the Bayes classification in was conducted with database of 47 samples and the recognition rate of 90% because of the development made by R. Bruneli.[18]

3.Requirement analysis

Requirement analysis is an analysis of both functional and non-functional requirement. To develop any system requirement analysis plays a key role. It also determines the expectations of the user for the system. It basically means to analyze, validate and test the system requirement for an indented final solution.

3.1 Functional requirement:

- The system allows the user to input the image for processing through an android application
- The back-end system compares the image with previously analyzed image by using machine learning technique.
- The output image will be a result of similarity percentage. By using **Neural network algorithm**, if and only if the similarity is higher than 80% the image will be identified in the cloud and signals will be sent to the **Node-MCU**.
- The signals received by the Microcontroller (N**ODE-MCU**) will take further action with the **servo motor**.

3.2 Non-functional requirement

The non-functional requirement includes the implementation of the system such as cloud security, authentication, accuracy, Machine learning models; Neural network, PCA, SVM. Facial recognition models; Eigen faces, Markov model.

Further performance attributes Includes:

- Accessibility: The system should run on all android phones with the minimum requirement of a camera. System requirement is as basic a s 0.5GHz processor and 1GB RAM.
- Robustness: The application should be responsive. The comparison of images should be quick and the screen change time should be as less as 0.5-1 second.
- Portability: The similar application can be developed in IOS environment with the current machine learning code.
- Maintainability: The system should require less updates over time to maintain its consistency of speed.
- Security: The system should be secured from system tampering and hacking.

3.3 Swot Analysis:

Strength	**Opportunities**
• **Faster than any other comparative biometric indentifiers.** • **Highest accuracy** • **Open source libraries can be used easily.**	• **A Reliable security based system in stores** • **Cheaper than any other proposed model in the market**
Weakness	Threats
• **Includes the accuracy in the real time video capturing and in-accurate facial recognition in it.** • **Uses too many images taken in different angles for the training and testing of data.** • **Use of external face covering masks could be the possible weakness of the system**	• External cause of damage of the system • Cloud server not being active. • Other robust application built in competition. • Authentication error while authenticating

Figure 1 : SWOT analysis

4. Design and Description of the system

4.1 Design

The Ace security system follows the basic principle of the **Machine learning** for recognizing the facial coordinates with Neural Network model. The system follows an accurate and effective way of recognizing a face.

4.1.1 Software module:

The system consists of an android application built with an open source development platform **Flutter Firebase** which captures/ allows to upload an image in the app using mobile camera, which sends the data to the cloud which already has **trained and tested** data of the recognized image.

4.1.2 Hardware module:

The hardware module consists of a microcontroller (NODE-MCU; Wi-Fi enabled) and a Servo motor. The Microcontroller receives the data from the cloud with the Wi-fi module enabled and connected to the mobile hotspot. The recognized image will be received by the MCU, which would further signal the **servo motor**.

4.2 Description:

4.2.1 Hardware:

Micro-controller:

Node MCU is the microcontroller which is been used in the Ace security system. An open source firmware which is used in Internet of things (IOT) based systems. The technical specification of the system includes; 32-bit **Xtensa** Processor, 160MHZ clock frequency with RTOS enabled functionality. Comes with 128kb of RAM and 4MB of ROM, making it a highly efficient to run extensive programs. The microcontroller comes with a built-in Wi-fi transceiver making it a fully equipped microcontroller.

The MCU operates between 3-6volt voltage, with an LDO regulator making it steady at 3.3volts. Current supply is 600mA. The regulator output. Is broken out into one of the sides of the board and labelled as 3V3, the power connectivity in this particular system could be a 5V battery or USB connection through **Vin pin**. The board has 17 General Purpose Input Output **(GPIO)** pin. There would be engagement of 2 out of 17 pins. For the **Servo Motor.**

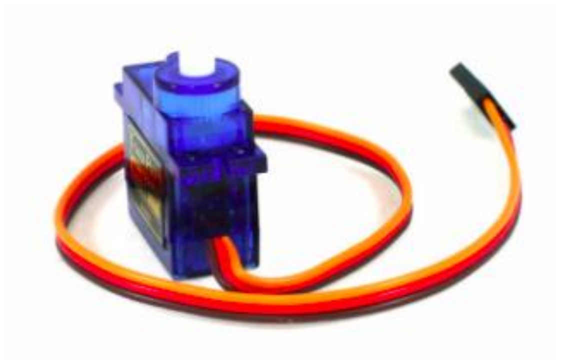


Figure 2: Servo motor

The Node-MCU has two added feature of a reset button and the other being a flash button for **firmware** update

NODE-MCU allows to work extensively and effectively with its top-notch system performance, small size less maintainability, high memory. The extensive 17 pin layout helps in various multipurpose use of the MCU and the **ESP8266** chip helping in network connection.

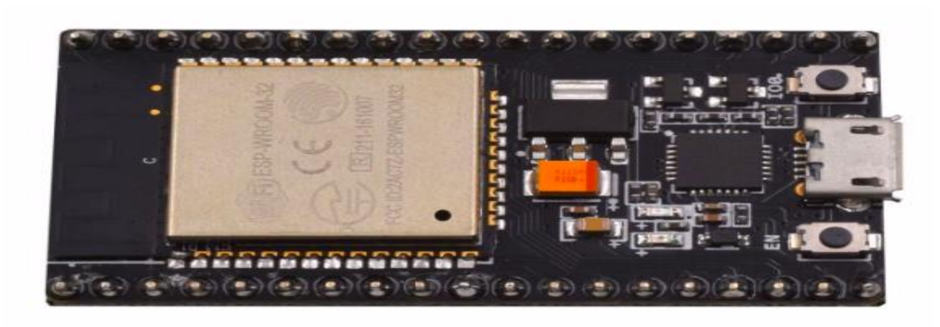


Figure 3: Node MCU

4.2.2 Software:

4.2.2.1 Flutter firebase:

Flutter is Google's mobile UI framework for crafting high-quality native interfaces on iOS and Android in record time. Flutter works with existing code, is used by developers and organizations around the world, and is free and open source.

Flutter firebase is an opensource cross-platform tool for creating Android and IOS applications using a modern framework. The idea of Flutter Firebase revolves around **DART**, which is a simple object-oriented language. It works with the Widgets. The whole UI system of Firebase is a fusion of numerous widgets, different styles of the elements such as font scheme or the color scheme, widgets etc.

Since the date of release of the Flutter Firebase it has made into top100 software on GitHub Stats. If we look at the current stats of the Flutter Firebase, it has made among the top **30 Software repository.**

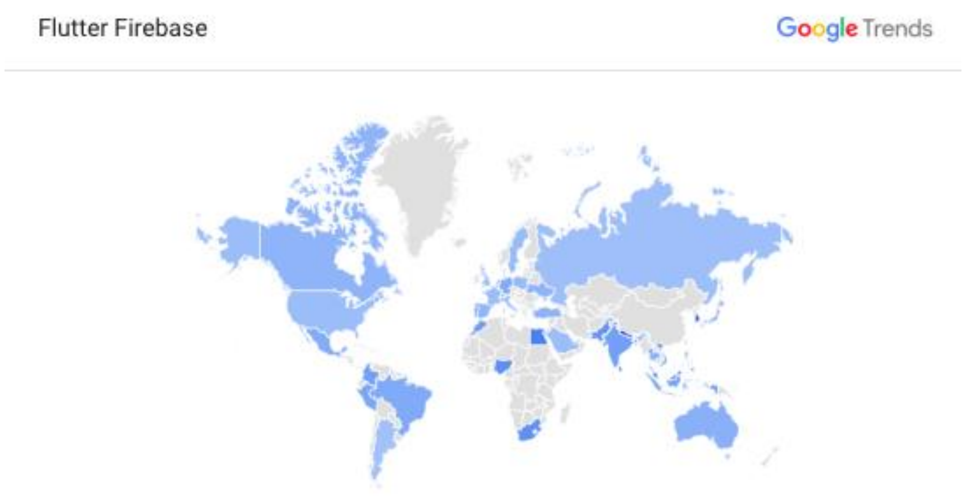


Figure 4: Google Trends of Flutter Firebase usage worldwide

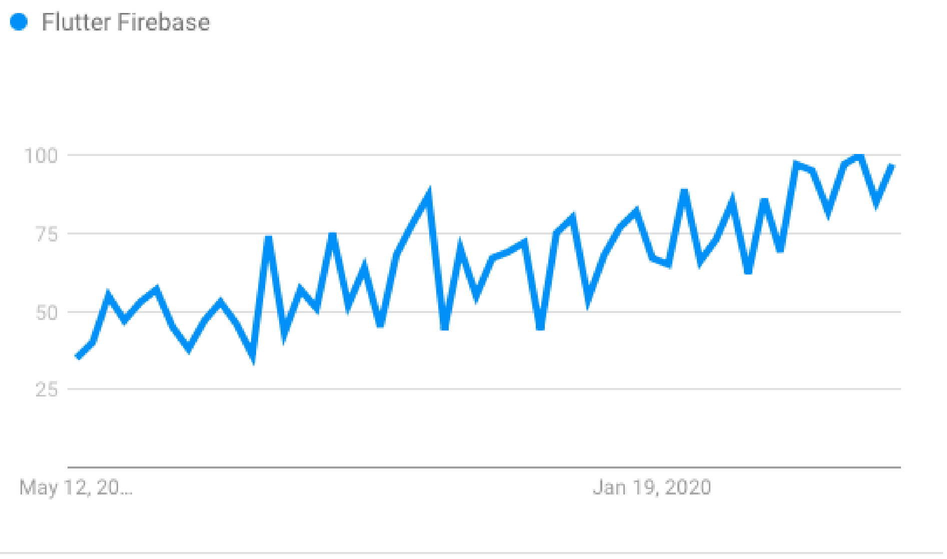


Figure 5 : Increasing Demand of Flutter Firebase over years

4.2.2.2 Why flutter Firebase?

Firebase is a platform owned by Google which is helpful in building different applications. Firebase provides wide variety of useful tools to the application developer in order to ensure high standard of quality maintenance in app development. The advantage attached to using Firebase is we can prepare a database for our app even without looking at the application structure which is a time-consuming factor.
Firebase is a very simple and straightforward platform which doesn't require any prior knowledge in the field of machine learning to perform task. Also, knowledge of neural networks or model optimisation also doesn't have any part to play in operation of this application. Firebase is a link between the proficiency of Google in Machine Learning and various applications of IOS and Android. Firebase also ensures customisation of TensorFlow Lite models to those having a deeper knowledge in Machine Learning.

Key Features of Firebase ML kit:

1. Readymade Application Programming Interface (APIs):
   Firebase ML Kit comes with different choices of APIs for the application developer. These APIs provides APIs depending upon the device availability such as it provides Text recognition, image labelling and landmark recognition on cloud. Whereas, on device it provides Smart Reply, Barcode Scanning, Face Screening etc. The user just has to do some coding and the full-fledged app is ready to use.

2. Availability of Custom Models Choices:
   If the Firebase fails to give you the choice of models which you want for your application programming, then you can develop your own model on TensorFlow. The models made on TensorFlow can be imported on Firebase which can then help the

user in hosting and serving of the application. However, this process needs expertise in Machine Learning.

3. On device and Cloud APIs:
   The application developer can make a choice between the Cloud APIs and on device APIs. Cloud APIs tend to take larger space than the on-device models. Cloud APIs are processed on Google Cloud whereas the on-device APIs are functional offline and with greater pace. The drawback to on device APIs is it doesn't provide reliable models. There are some models which works on-device and cloud as well such as text recognition, and image labelling.

4. Works on varied platforms:
   The Software Development Kit of ML can be used on **Android and on IOS** as well. However, IOS has its own platform namely Core ML which is still dominated over this google based platform but has a high chance to defeat it. The Models of ML kit shows a great performance in latest Android version such as Pie. It also has a capability to work on the earlier Android versions such as Android 4.0.

Capabilities of APIs:

1. Text Recognition:
   Text Recognition API works on device and on cloud as well. It enables the user to recognize Latin language (generally every language) which is presented in text format. This model makes the tedious job of monotonous data entry on business cards, credit cards, invitation etc easy by automation. This feature in Cloud based model also retrieve the data from images. Afterwards, this retrieved data can be useful for the purpose of document translation. The apps with text recognition can also obtain the numbers from surfaces.

2. Face Detection:
   Face Detection feature can help to obtain various facial features such as the position of nose, eyes, mouth etc and also to obtain the face contours. This feature is used to intensify the selfies by adding various filters. This does also work on real time basis where we can apply filters and make emoticons based on your real facial features on our visual calling as well. However, this feature can be applied offline only.

3. Image Labelling:
   Image Labelling is available on-device and on cloud as well. It helps to identify the contents of the image you have captured. The contents of an image can be recognised without providing an additional metadata. While using Image Labelling APIs can provide the list of contents such as animal, people, infrastructure, place, action etc. A confidence score is provided with the list which indicated the level of accuracy of ML in identifying the object.

4. Smart reply:
   Smart Reply can be used in Informal conversations. As we see on our emails, after receiving a mail we get some suggestions to reply which is automatically generated by the app. This enables the replies quicker for the respondents. These reply suggestions are not only of one word but have a context to it. However, English is only language supported currently for this feature.

5. Landmark Recognition:
   This feature is only available on the cloud platform. It enables the user to identify the popular landmark in a picture. The geographical coordinates of the picture taken can also be identifies through this feature which can gave an idea about the place where it has been captured.

6. Barcode Scanning:
   Barcode scanning helps the user to encode the encrypted data. Scanning can be performed without internet connection. The user opens a barcode scanner and capture the barcode into it. The API will help to encode the data which can be some contact information, ATM card related information etc. Barcode ensures maximum privacy to transfer the date from one source to others.

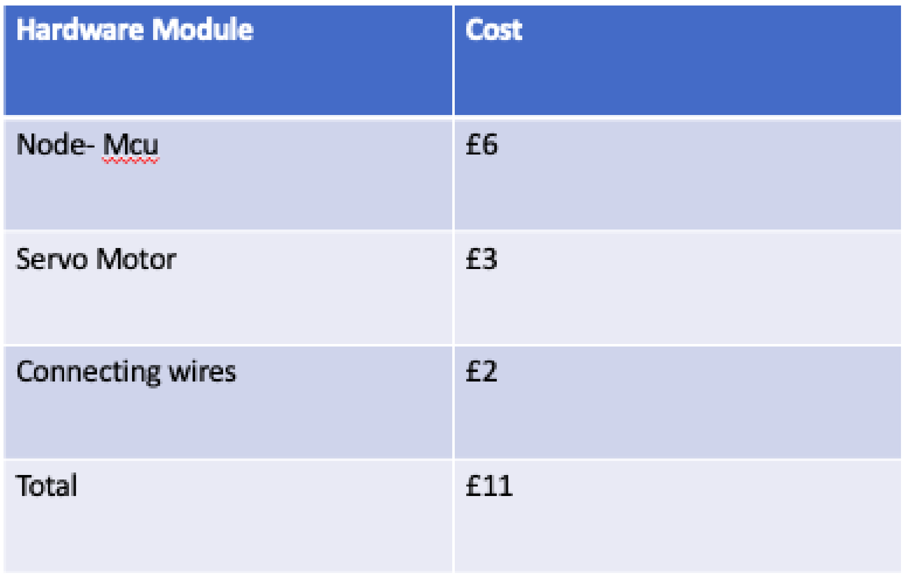
7. Translation:
   The Translation feature is available offline, and it is based on the idea of Google Translate app. This API provides the user to translate 59 specified languages. The user can switch between these 59 languages.

8. Content Detection and Tracking:
   The content detection enables the user to locate the object in the image. It provides the name of location where the picture has been taken and also enables to know the location in the live camera feed. This feature can be useful while making a live visual search. It will enable the user to know the object's buying places.

9. **AutoML Vision Edge:**
   AutoML Vision Edge is an updated part to the feature of Image Labelling. In this feature a user can train the API to recognize and classify the objects in the picture into 400 categories. For example, the user wants to classify the specific model of the car. This can be achieved by uploading the picture and training the model.

10. Custom Model:
    The Custom Model enables the user to custom made the models through TensorFlow Lite with ML kit if they do not find the appropriate model readily available in the Firebase.

4.2.2.3 Cost sheet



Hardware Module	Cost
Node- Mcu	£6
Servo Motor	£3
Connecting wires	£2
Total	£11

Figure 6: Cost sheet

5. Implementation

5.1 NEURAL NETWORKS:

Neural network is one of the topics which had an exponential rise since past 3 decades.[13] It relates to the human brain working of networking the information, processing it and creating a simple model, yet effective. The neural network work on the principle of having a large number of nodes interconnected to each other, known as **activation function.** Another term known as the weighing value, is the memory of two interconnected nodes. The weights all together comprises of the whole memory of the neural network. The network is built either on the basis of logic strategy or probably any approximate algorithm of the model.

The exponential rise in the research and development area of the Neural Network was a boon to the society. It had solved numerous problems in the field of automation, robotics, Networking, medicine, financial economy. The inability of the computer to solve realistic and practical problems were now been solved by the means of artificial Neural network.[13]

Neural network consists of numerous learning neurons, with constant learning in a topology of network. It has been majorly divided into Feedforward and Feed backward network connection.

Feed Forward:
The feed forward receives the input from the previous stage, process it and send the output to the different stage. There is no feedback released in the network and hence can be represented with directed loop free graph. The capability of processing the input to the output comes from the recombination of multiple non-linear function

Feedback network:
The feedback network involves the transmission of feedback from the neurons. This type of connection can be diagrammatically represented by the undirected complete graph. Moreover, the transmission state in this connection can be handled by dynamic system theory. Associative memory function is the key to the function's stability.

5.1.1 Why Use Neural Network?

Neural Network is widely used due to its ability to compute information from irregular, complicated or inaccurate data which can be used to know the exact patterns and trends which are difficult to compute by human or computer techniques. The well-trained neural network is said to be the 'expert' to provide accurate information for analysis.
Other advantages include; The neural network in adaptive in nature, it has an ability to learn about how to perform a task as per the guidelines and training data provided to the system.
It self-organizes and represents the data or information on its own which is provided during learning time.

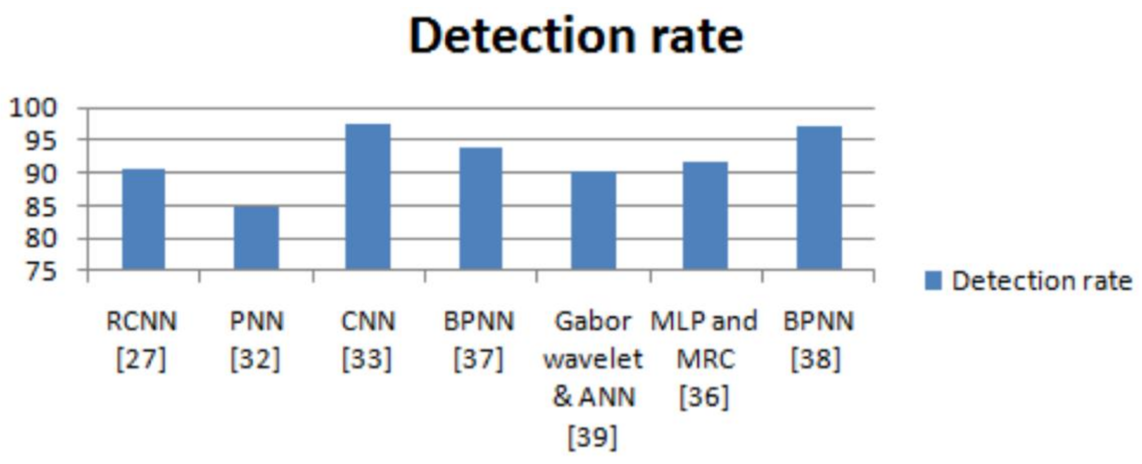
The artificial neural networks calculations can be conducted parallelly for which exquisite hardware are been produced to take an advantage of this trait. In neural network the capabilities remain stable even if there is major network damage.

5.1.2 Learning Method:

Supervised Learning: The ANN is based on the supervised learning. Supervised learning involves the comparison of actual output provided by neural network and the desired output. Weights are set initially for the network to adjust its working in order to provide closer results to the desired results in the upcoming cycles. The weights are adjusted on regular basis in order to make the network more reliable and accurate. The training is provided by giving some data about inputs and outputs to the system which is known as training sets. In prototype system this learning can take weeks because of its low processing power. The learning is said to be completed when the network starts providing the desired results of the user. When the system reaches at this stage weights are set to be frozen. However, some systems are gradually trained in their life span as well. [20]

From all the models in ANN such as Multilayer Perceptron (MLP); Back Propagation Neural Networks (BPNN); Retinal Connected Neural Network (RCNN); Rotation Invariant Neural Network (RINN); Fast Neural Networks (FNN); Convolutional Neural Network; Polynomial Neural Network (PNN), Convolutional Neural Network is used for the preparation of the model due to its less error rate and higher detection rate as compared to other models. The below graph shows the detection rate of all the models in Neural Network. [20]

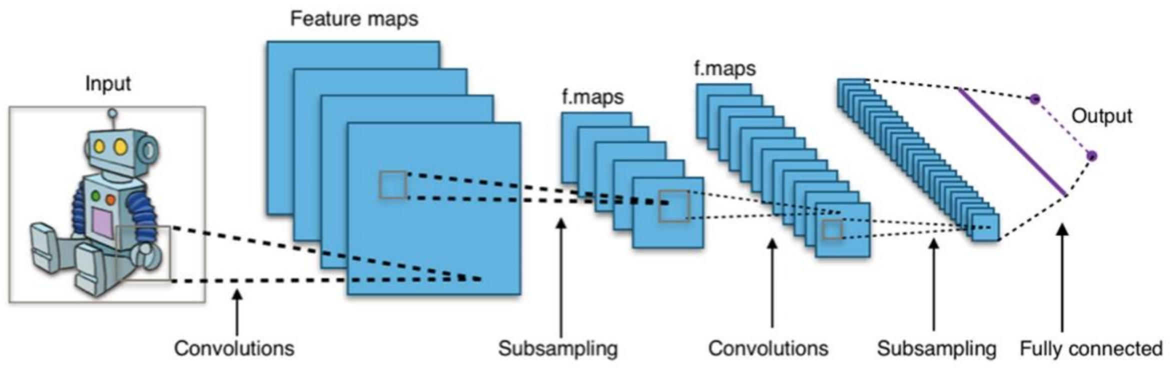
Figure 7: Detection rate of different Machine Learning models



Source: The International Journal of Multimedia & Its Applications (2014)

5.1.3 C-NN and Face Recognition building

Figure 8: Multilayer Depiction of CNN Model



Source: The International Journal of Multimedia & Its Applications (2014)

Convolutional neural Network (CNN) works on the principle of multi-layer perception of processing an image. The input goes through a multiple layered filters (kernel), to classify the image in a probability of 0 or 1. The initial layers consists of convolutional layer and a pooled layer for simulating the feature extraction of the particular image.

5.1.4 Building Face Recognition Model with CNN

There are mainly two face recognition algorithms:

Feature-based method which runs on the principle of extracting features which in turn sends the classifier for face recognition.

Representation-based methods runs on the principle of converting the 2D image input into 3D and then analyzing the image using mathematical formulae. For example; Fisher face, State vector Machine, Eigenface etc.

The CNN methods revolve around the feature extraction method. Feature based recognition system recognizes the feature through set features and **Hidden Markov Model**. In this method the features are extracted, classified and sent for the recognition with ways stated above. The CNN model is distinct from the traditional feature extractor and high-performance classifier for feature's classification. There is less burden of training the samples and also less layer of process in this method. This method is generally distributed in two stages, in the first stage the convolutional dimension reduction is conducted and in second stage multi-layer non-linear mapping is done to extract features. The main characteristic of this model is it can develop classifier and extractor on its own by learning through unprocessed samples.

CNN is based on the idea of multiple layer perceptron which emerged on the basis of biological vision and simplest form of preprocessing operation. The network factor is the major distinction between multi-layer perceptron and CNN. The CNN consists of convolutional layer at the first stage and the later stages are the combination of simple and complex cells for high level feature extraction in visual layer. CNN is a forward feedback neural network.

The previous layer (the local receptive field) sends the output to the convolutional neurons and the convolutional neurons responds to that as an input at this stage by extracting high-level features from the given output. Then the obtained features are then passed on to later stages for processing. To avoid any deformation or displacement of the input the overlapping between the areas are minimized.

Popular CNN Architecture:

LeNet : This architecture was first introduced by LeCun et al.in the year 1998 for OCR and digit recognition in documents. It is a combination of 5 convolutional layers and one fully connected layer. The LeNet architecture on the basis of memory footprints are said to be simple and straightforward. This architecture can also run on CPU but using GPU can give faster training process.

AlexNet : AlexNet is the combination of 8 layers in which 5 layers is convolutional and 3 fully connected layers. Some of convolutional layers in this architecture are followed by max-pooling layers. It was more effective architecture in computer vision task.

VGG-16: It doesn’t use hyper parameters making it a simpler version of architecture. It uses 3x3 filters with stride of 1 in convolutional layer and uses SAME padding in pooling layers 2x2 with stride of 2.

GoogLeNet:
The error rate in this architecture falls under top 5 with 6.67%. It travels deep in parallel paths with 22 layers in total.

CNN Model:

The model uses the model of LeNet-5 which is discussed in detail below:

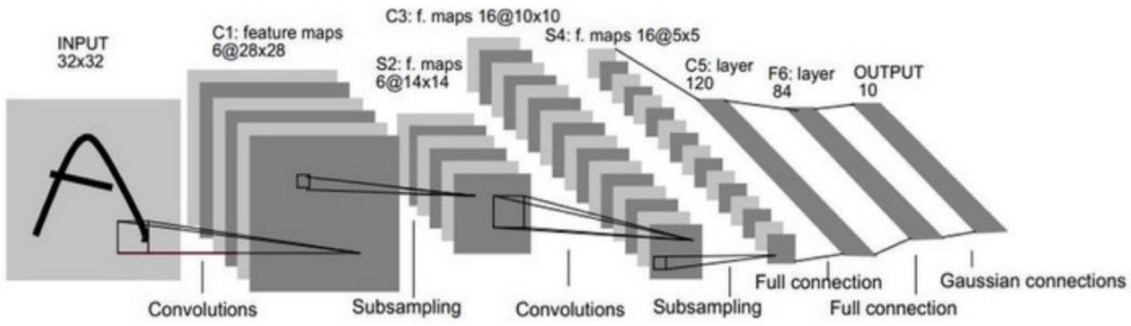


Figure 9: Working of LeNet-5 Model ()

The above diagram represents the working of LeNet-5 CNN. The explanation of the above diagram is distributed on the basis of layers.

Convolution Layer: The convolution layer extracts the prime feature using simple connection and weight sharing it to local receptive fields. Local connection shows the connectivity between the neuron on convolutional layer and neurons fixed in an area in previous feature map. However, weight sharing means that the neurons on same feature map uses the same strength of connection as of previous layer. This type of connections can reduce the training time because the same strength acts as a feature extractor which is termed as convolution kennel in computing process which is randomly expected on initial stage but computed with network training.

The pooling layer:

The pooling layer mimics the complex cell which does the screening and convert the primary visual features in advanced and more abstract versions. This is implemented through sampling in the network. The sampling by the pooling layer does not change the number of output feature maps but shrinks the size of the feature maps which can reduce the calculation difficulties and ensure smaller displacement changes. The model uses the sampling size of 2*2, it divides the non-overlapping rectangles in dimension of 2*2 and the maximum value of each rectangle so the output feature map is produced. The length and width of the output is half of the input provided.

Fully connected layer: The fully connected layer represents the connection between the previous layer and the forwards layers but the neurons on the same layers remains detached. This is done to justify the non-linearity of the network and also to restrict the size of the network, where the network gains the feature from the 4 layers and connect in into a fully connected layer.

6. Evaluation and Result

The system follows a basic facial recognition process but with higher versatility. It contains an android application, developed in an open source environment of Flutter Firebase with the Machine learning algorithm in it. A step by step procedure is followed in order to make the system work;

Firstly, an image is trained and tested under the neural network algorithm of Machine learning which gets uploaded in the application's database. This allows the system to analyze the image and store it for future comparison with another image which would be uploaded using the android application.

The next step is the setup of the Microcontroller NODE-MCU and the servomotor using a PC/Laptop. The NODE-MCU will be connected to the internet with the same network as the mobile android device. Making it highly versatile system due to its mobile nature. The servomotor which is connected to the NODE-MCU signifies the output or the result of the whole system. Another alternate of the servomotor could be a buzzer, notification with GPS coordinates, SOS signal etc. The sole purpose of the servomotor is to conclude that the system works perfectly well, i.e. if the facial coordinates uploaded for checking in the app, matches the previously trained and tested image. If the face coordinates match, it rotates the servo motor and signals that, the uploaded image is similar to the previously trained image/images.

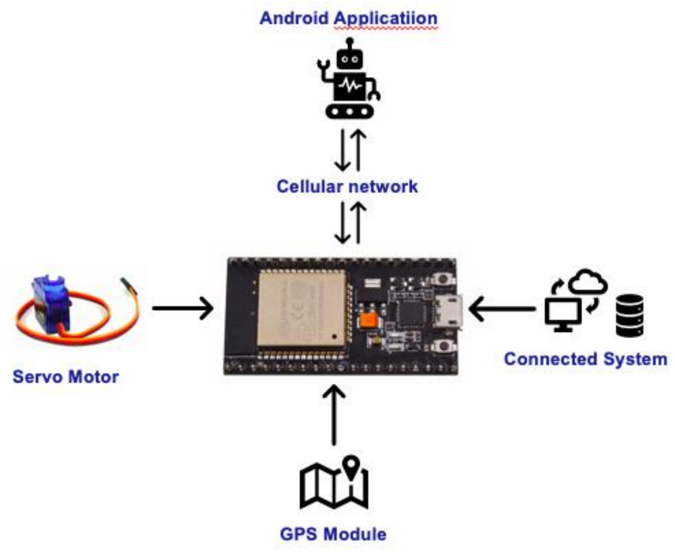


Figure 10: Architecture of Ace Security Model

The user uploads an image in the application. Two options have been made available to upload the picture, one is through system storage and another one is system camera. Once the facial image has been uploaded, the application would signal if at all it is an instance of facial data.

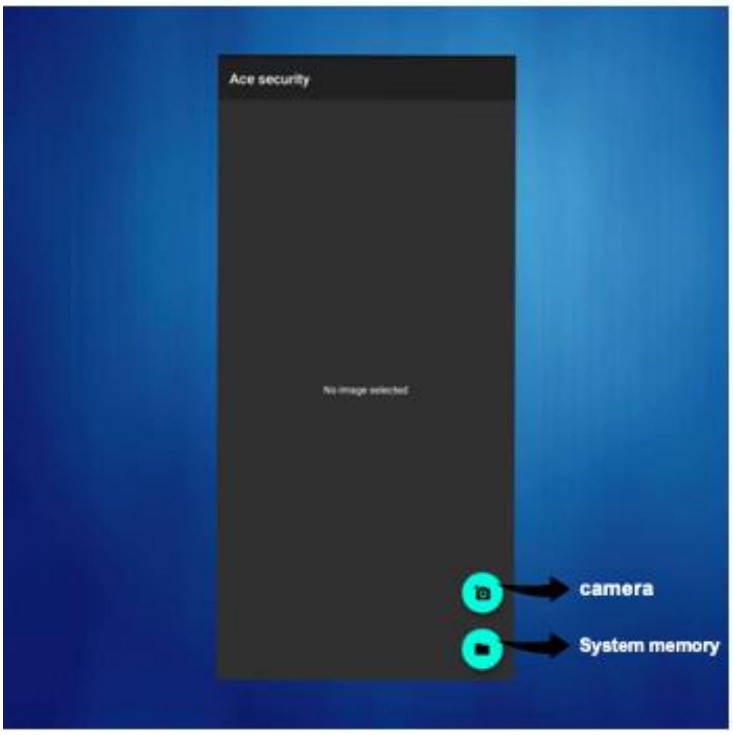
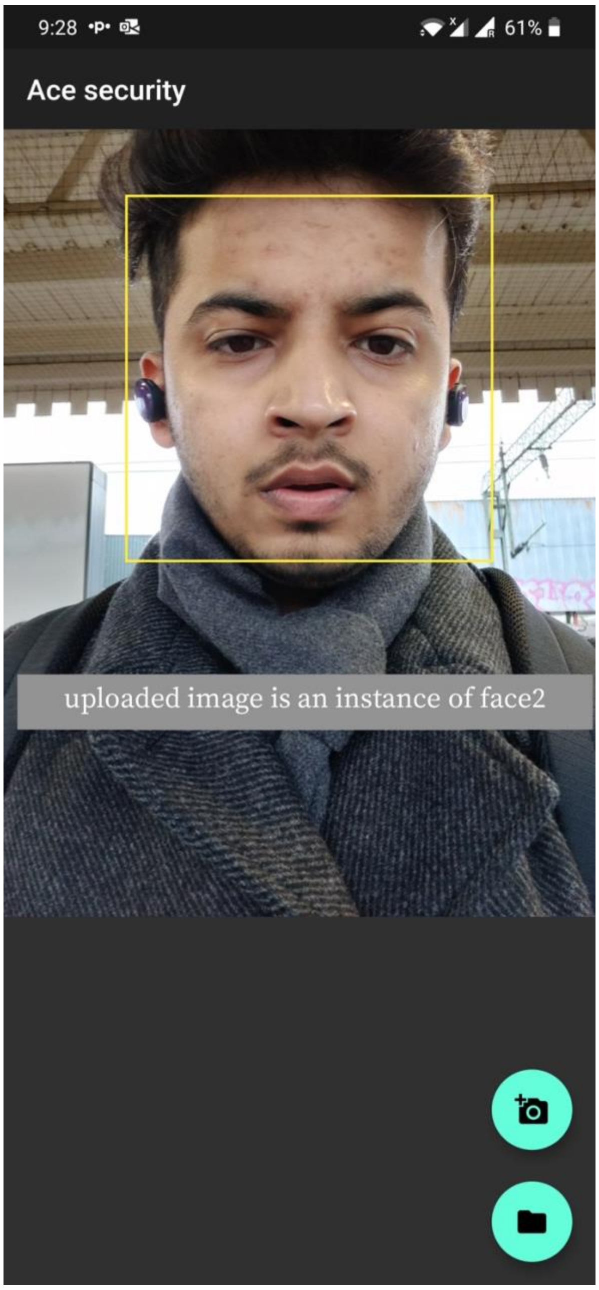


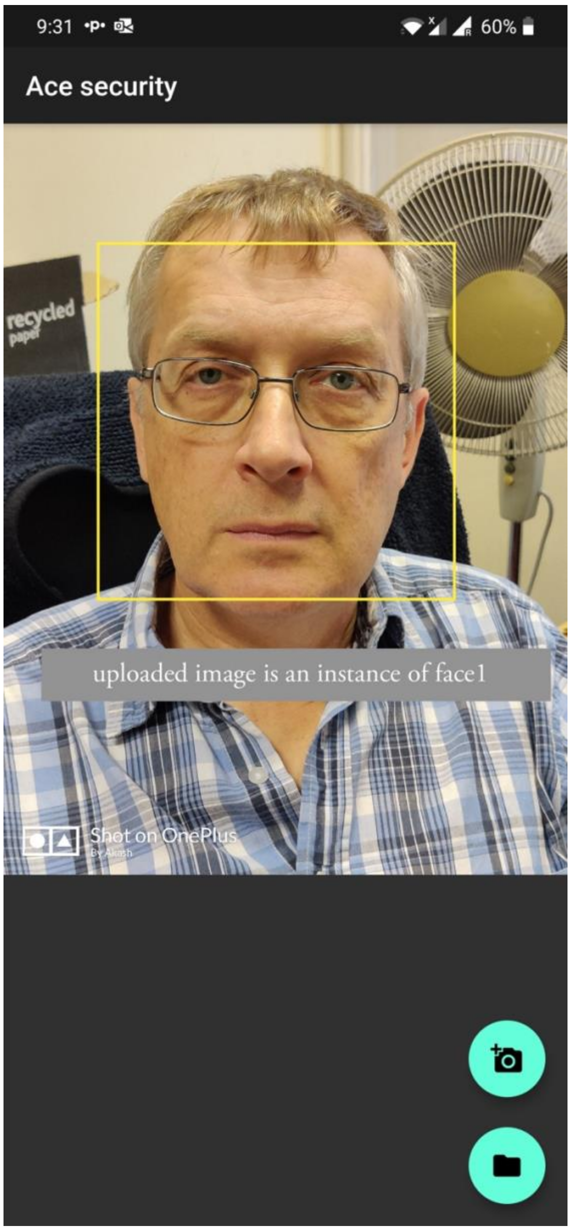
Figure 11: Insight of Ace Security Android Application

The system contains facial data of two persons. **Instance 1 and instance 2,** the system is been built in such a way that if the first person’s face data is been recognized then, the servomotor rotates clockwise. If the second person’s facial data is been recognized, the servomotor would rotate anti-clockwise. This signifies that the system opens for one person and closes for another. If, an image uploaded is not similar to either of the facial data, the servomotor doesn’t rotate but instead the application notifies the similarity percentage of the facial image uploaded based on its features.



uploaded image is an instance of face2

Figure 12: Testing result and confirmation of Matching Image of Face2



uploaded image is an instance of face1

Figure 13: Testing result and confirmation of Matching Image of Face2

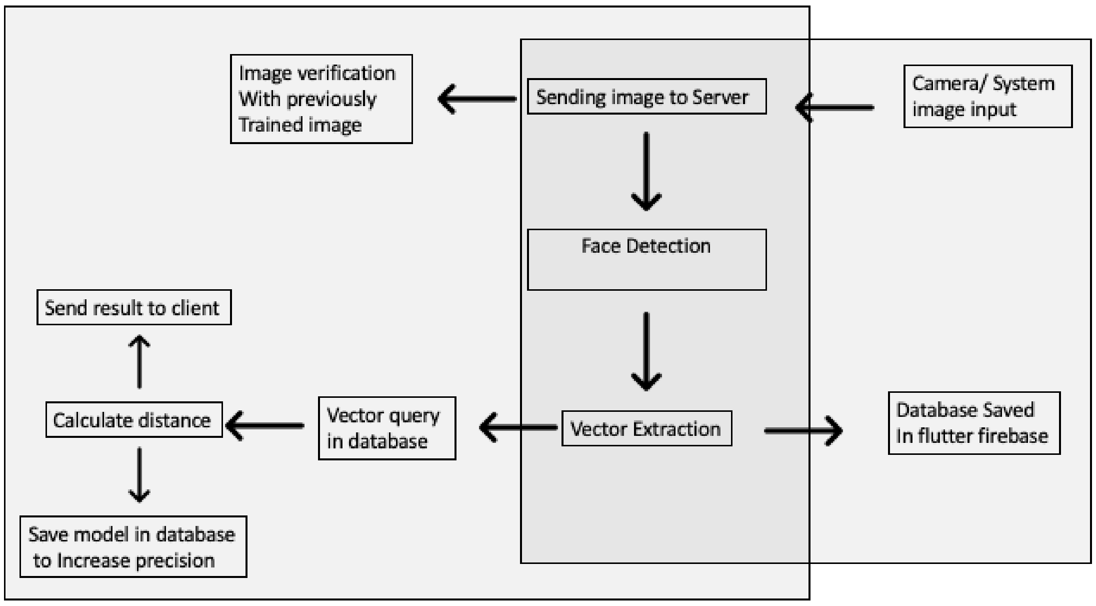


Image verification With previously Trained image
Sending image to Server
Camera/ System image input
Face Detection
Send result to client
Calculate distance
Vector query in database
Vector Extraction
Database Saved In flutter firebase
Save model in database to Increase precision

Figure 14: Diagram depicting Basic Working of Machine learning model

7. Conclusion

The face recognition system servers a wider range of application starting from home door locks to the industrial usage of the system. Ace security system believes in being economical as well as being accurate and robust in nature. All the available face recognition system in the market are priced too high. To get a reliable and a robust system you need to pay a huge amount. The problem arises for the cottage industries where they cannot afford the huge amount to make their store/office highly secure. For instance, small food stores, off-licence shops etc. suffer from security issues. Even after installing the CCTV cameras they fail to identify previously known thief. This increases the theft rate in the small industries. The turnover of small shops couldn't afford a high valued face recognition system to meet their security issue.

The sole purpose to build this system was to make it available for all.

To conclude, I would say that the outcome of the system was positive. Starting from an idea which struck me while I was working part-time, to a swift delivery of the system with all the aspects included. A system can be built only and only with a consistent hard work. The hurdles or the real time compilation error which I received made me highly experienced in building something which has the potential to change the way that a system works. A highly technical forward system needs a high maintenance, my system doesn't. You have to pay a huge amount to setup such an accurate system, my system costs hardly $20 (excluding labour charge). To wrap it up, I would like to say that the system design formation has been one of the biggest projects that I've ever worked on. The chosen project was a fusion of software engineering plus hardware/electronic engineering topics, which defines my course Computer systems engineering. I take pride to say that I applied all the theoretical as well as the practical knowledge gained from my coursework since year 1 and the result was a system which serves a cause i.e. making it available to all.

7.1 Future Scope

All the applications built; systems developed always has a scope of improvement in several areas. Thus, my system can also be improved in several ways to make it more robust in the coming future.

Firstly, A cross-platform system application can be built in the iPhone Operating System (IOS), making it available to all the mobile platforms.

Secondly, the CCTV camera could be used to take down the facial data which could then be connected directly to the database where the machine learning algorithm trains and tests the images.

Lastly, the micro-controller used can be updated according to the future need, to make it cop up with the future technologies.

7.2 Challenges Faced

The time frame of two semesters allotted to complete a project with huge complexities was an initial challenge to me. I always had an impression of not making it on time or may be make it an incomplete project. Also, expertise in certain topics needed to be acquired before starting that particular module. The Exams, part-time job and on-going assessments were some add on challenges which I had to carry along with the project.

Another challenge was to overcome the GDPR law, which states that the system was against taking down and saving the facial coordinates into the database. I overcame the challenge as, my system serves multiple purposes starting from personal security to the industrial security. The home door lock doesn’t come under the GDPR. The industrial usage of the system was a problem. But my system would only deal with the already identified theft. Also, to protect your business you have to make it secure. This challenge took me a month to overcome. The fact that you’re dealing with the already identified faces makes it overcome the GDPR.

Finally, the on-going pandemic has adversely affected the implementation of the project. I had almost finished my project a long time back. But, due to this pandemic I had a real hard time in implementing the final system and created a rush due to which I couldn’t complete and submit my report on time.

8. Reference

1. Monika Shah, Deepika Shukla, Dhara Pandya (2020), Smart Gate: Intelligent Security System Based on Face Recognition: *International Journal of Innovative Technology and Exploring Engineering (IJITEE*): Vol 9(3), pp.601-60
2. See, J.; Eswaran, C. and Fauzi (2011): Video-Based Face Recognition Using Spatio-Temporal Representations in Reviews, Refinements and New Ideas in Face Recognition, Corcoran P., Ed., Intech, Croatia, pp. 273-293, 2011.
3. Rady H. (2011): Face Recognition using Principle Component Analysis with Different Distance Classifiers: *International Journal of Computer Science and Network Security*, Vol. 11 No. 10, pp. 134-143
4. Qasemjaber, Zahraa & Younis, Mohammed. (2014). Design and Implementation of Real Time Face Recognition System (RTFRS): *International Journal of Computer Applications,* Vol.94(12),pp.15-22
5. Krishna B.; Bindu V., Durga K. and AshokKumar G. (2012): An Efficient Face Recognition System by Declining Rejection Rate using PCA: *International Journal of Engineering Science & Advanced Technology*, Vol. 2, No. 1, pp. 93 – 98
6. Nguyen, Trung & Lakshmanan, Barth & Sheng, Weihua. (2018). A Smart Security System with Face Recognition. Available at: https://www.researchgate.net/publication/329884419_A_Smart_Security_System_with_Face_Recognition Viewed on: 28 April 2020

7. M. J. Paul Viola (2004): Robust Real-time Object Detection: *International Journal of Computer Vision*, pp. 137-154.
8. Bo Wu, Shihong Lao, Chang Huang (2004): Fast rotation invariant multi-view face detection based on real Adaboost: *Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 79-84.
9. Y. Li, H. Ai, T. Yamashita, S. Lao and M. Kawade (2008): "Tracking in Low Frame Rate Video: A Cascade Particle Filter with Discriminative Observers of Different Life Spans," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 10, pp. 1728-1740.
10. Cootes T, F. Taylor, J. Cooper & H. Graham (1995): Active Shape Models-Their Training and Application: *Computer Vision and Image Understanding*, Vol.16(1) pp. 38-59.
11. X. Cao, Y. Wei, F. Wen and J. Sun (2012): Face alignment by Explicit Shape Regression: *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, pp. 2887-2894
12. Wenchao Zhang, Shinguang Shan, Wen Gao (2005): Local Gabor binary pattern histogram sequence (LGBPHS): a novel non-statistical model for face representation and recognition: *Tenth IEEE International Conference on Computer Vision*, pp. 786-791

13. Owayjan, Michel & Dergham, Amer & Haber, Gerges & Fakih, Nidal & Hamoush, Ahmad & Abdo, Elie. (2013). Face Recognition Security System. Available at: https://www.researchgate.net/publication/259027363_Face_Recognition_Security_System
14. Sushma Jaiswal, Dr. (Smt.) Sarita Singh Bhadauria, Dr. Rakesh Singh Jadon (2011): Comparison between face recognition algorithm-Eigenfaces, Fisherfaces and Elastic bunch graph matching, *Journal of Global Research in Computer Science* Volume 2 (7).
15. Ming-Hsuan Yang, David J. Kriegma and Narendra Ahuja (2002): Detecting Analysis and Machine Intelligence*: IEEE Transactions on Pattern Analysis and machine intelligence*, Vol. 24(1), pp.34-58
16. R. Bruneli and t. Poggio (1993): Face recognition: features versus templates, *IEEE Transaction Pattern Analysis and Machine Intelligence*, vol. 15, pp. 1042-1052
17. V., Hespanda, J., Kiregeman, D. (1997): Eigenfaces vs fishersfaces: recognition using class specific linear projection, *IEEE*, V. 19, pp. 711-720
18. Eaurenz Wiskott, Jean-Marc Fellous, Norbert Krüger, and Christoph von der Malsburg (1997): Face Recognition by Elastic Bunch Graph Matching: *IEEE Transactions on pattern analysis and machine intelligence*, Vol. 19(7), pp. 775-779

19. Le, Thai. (2011). Applying Artificial Neural Networks for Face Recognition. Advances in Artificial Neural Systems. 2011. 10.1155/2011/673016. Available at: https://www.researchgate.net/publication/258380240_Applying_Artificial_Neural_Networks_for_Face_Recognition viewed on: 1st May 2020

20. Omaima N. A. AL-Allaf (2014): Review of face detection systems based artificial neural networks algorithms: *The International Journal of Multimedia & Its Applications*: Vol.6(1), pp. 1-16

Websites for term definitions:

https://www.investopedia.com/terms/n/neuralnetwork.asp

https://www.electronicwings.com/nodemcu/introduction-to-nodemcu

https://circuitdigest.com/article/servo-motor-basics

https://expertsystem.com/machine-learning-definition/

https://flutter.dev/docs/development/data-and-backend/firebase

http://www.linux-xtensa.org

https://community.estimote.com/hc/en-us/articles/217429867-What-is-GPIO-

https://techterms.com/definition/firmware

https://www.espressif.com/en/products/socs/esp8266/overview

https://www.techopedia.com/definition/32890/software-repository).

https://www.mulesoft.com/resources/api/what-is-an-api

https://thenextweb.com/topic/ios/)

https://developers.google.com/ml-kit/vision/auto-ml-vision-edge

https://web.stanford.edu/~jurafsky/slp3/A.pdf

https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d)

9. Appendix

Python Code for the Neural Network algorithm model used:

Setup.py

```
from cx_Freeze import setup, Executable
import sys,os
PYTHON_INSTALL_DIR = os.path.dirname(os.path.dirname(os.__file__))
os.environ['TCL_LIBRARY'] = os.path.join(PYTHON_INSTALL_DIR, 'tcl', 'tcl8.6')
os.environ['TK_LIBRARY'] = os.path.join(PYTHON_INSTALL_DIR, 'tcl', 'tk8.6')

base = None

if sys.platform == 'win32':
    base = None


executables = [Executable("train.py", base=base)]

packages = ["idna","os","sys","cx_Freeze","tkinter","cv2","setup",
            "numpy","PIL","pandas","datetime","time"]
options = {
    'build_exe': {

        'packages':packages,
    },

}

setup(
    name = "ToolBox",
    options = options,
    version = "0.0.1",
    description = 'Vision ToolBox',
    executables = executables
)
```

Training and testing CNN model code

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

from datetime import datetime
import os.path
import os
import time
import sys
sys.path.insert(0,'lib')
sys.path.insert(0,'networks')
import tensorflow as tf
from tensorflow.python.client import timeline
from tensorflow.contrib import slim
import tensorflow.contrib.data as tf_data
from collections import Counter
import numpy as np
import importlib
import itertools
import tensorflow.contrib.slim as slim
from tensorflow.contrib.slim.nets import resnet_v1, resnet_v2
import argparse
import utils
import sphere_network as network
import inception_resnet_v1 as inception_net
import resface as resface
#import lfw
import pdb
#import cv2
#import pylab as plt
```

```
debug = False
softmax_ind = 0

from tensorflow.python.ops import data_flow_ops

def _from_tensor_slices(tensors_x,tensors_y):
    #return TensorSliceDataset((tensors_x,tensors_y))
    return tf_data.Dataset.from_tensor_slices((tensors_x,tensors_y))



def main(args):

    #network = importlib.import_module(args.model_def)

    subdir = datetime.strftime(datetime.now(), '%Y%m%d-%H%M%S')
    log_dir = os.path.join(os.path.expanduser(args.logs_base_dir), subdir)
    if not os.path.isdir(log_dir):  # Create the log directory if it doesn't exist
        os.makedirs(log_dir)
    model_dir = os.path.join(os.path.expanduser(args.models_base_dir), subdir)
    if not os.path.isdir(model_dir):  # Create the model directory if it doesn't exist
        os.makedirs(model_dir)

    # Write arguments to a text file
    utils.write_arguments_to_file(args, os.path.join(log_dir, 'arguments.txt'))

    # Store some git revision info in a text file in the log directory
    src_path,_ = os.path.split(os.path.realpath(__file__))
    utils.store_revision_info(src_path, log_dir, ' '.join(sys.argv))

    np.random.seed(seed=args.seed)

    #train_set = utils.get_dataset(args.data_dir)
```

```
train_set = utils.dataset_from_list(args.data_dir,args.list_file)
nrof_classes = len(train_set)
print('nrof_classes: ',nrof_classes)
image_list, label_list = utils.get_image_paths_and_labels(train_set)
print('total images: ',len(image_list))
image_list = np.array(image_list)
label_list = np.array(label_list,dtype=np.int32)

dataset_size = len(image_list)
single_batch_size = args.people_per_batch*args.images_per_person
indices = range(dataset_size)
np.random.shuffle(indices)

def _sample_people_softmax(x):
    global softmax_ind
    if softmax_ind >= dataset_size:
        np.random.shuffle(indices)
        softmax_ind = 0
    true_num_batch = min(single_batch_size,dataset_size - softmax_ind)

    sample_paths = image_list[indices[softmax_ind:softmax_ind+true_num_batch]]
    sample_labels = label_list[indices[softmax_ind:softmax_ind+true_num_batch]]

    softmax_ind += true_num_batch

    return (np.array(sample_paths), np.array(sample_labels,dtype=np.int32))

def _sample_people(x):
    '''We sample people based on tf.data, where we can use transform and prefetch.

    '''

    image_paths, num_per_class =
sample_people(train_set,args.people_per_batch*(args.num_gpus-1),args.images_per_person)
```

```
    labels = []
    for i in range(len(num_per_class)):
        labels.extend([i]*num_per_class[i])
    return (np.array(image_paths),np.array(labels,dtype=np.int32))

def _parse_function(filename,label):
    file_contents = tf.read_file(filename)
    image = tf.image.decode_image(file_contents, channels=3)
    #image = tf.image.decode_jpeg(file_contents, channels=3)
    print(image.shape)

    if args.random_crop:
        print('use random crop')
        image = tf.random_crop(image, [args.image_size, args.image_size, 3])
    else:
        print('Not use random crop')
        #image.set_shape((args.image_size, args.image_size, 3))
        image.set_shape((None,None, 3))
        image = tf.image.resize_images(image, size=(args.image_height, args.image_width))
        #print(image.shape)
    if args.random_flip:
        image = tf.image.random_flip_left_right(image)

    #pylint: disable=no-member
    #image.set_shape((args.image_size, args.image_size, 3))
    image.set_shape((args.image_height, args.image_width, 3))
    if debug:
        image = tf.cast(image,tf.float32)
    else:
        image = tf.cast(image,tf.float32)
        image = tf.subtract(image,127.5)
        image = tf.div(image,128.)
        #image = tf.image.per_image_standardization(image)
    return image, label
    print('Model directory: %s' % model_dir)
    print('Log directory: %s' % log_dir)
    if args.pretrained_model:
        print('Pre-trained model: %s' % os.path.expanduser(args.pretrained_model))


    with tf.Graph().as_default():
        tf.set_random_seed(args.seed)
        global_step = tf.Variable(0, trainable=False,name='global_step')

        # Placeholder for the learning rate
        learning_rate_placeholder = tf.placeholder(tf.float32, name='learning_rate')


        phase_train_placeholder = tf.placeholder(tf.bool, name='phase_train')




        #the image is generated by sequence
        with tf.device("/cpu:0"):

            softmax_dataset =
tf_data.Dataset.range(args.epoch_size*args.max_nrof_epochs*100)
            softmax_dataset = softmax_dataset.map(lambda x:
tf.py_func(_sample_people_softmax,[x],[tf.string,tf.int32]))
            softmax_dataset = softmax_dataset.flat_map(_from_tensor_slices)
            softmax_dataset =
softmax_dataset.map(_parse_function,num_threads=8,output_buffer_size=2000)
            softmax_dataset = softmax_dataset.batch(args.num_gpus*single_batch_size)
            softmax_iterator = softmax_dataset.make_initializable_iterator()
            softmax_next_element = softmax_iterator.get_next()
            softmax_next_element[0].set_shape((args.num_gpus*single_batch_size,
args.image_height,args.image_width,3))
            softmax_next_element[1].set_shape(args.num_gpus*single_batch_size)
            batch_image_split = tf.split(softmax_next_element[0],args.num_gpus)
            batch_label_split = tf.split(softmax_next_element[1],args.num_gpus)







        learning_rate = tf.train.exponential_decay(learning_rate_placeholder, global_step,
            args.learning_rate_decay_epochs*args.epoch_size, args.learning_rate_decay_factor,
staircase=True)
        tf.summary.scalar('learning_rate', learning_rate)

        print('Using optimizer: {}'.format(args.optimizer))
        if args.optimizer == 'ADAGRAD':
            opt = tf.train.AdagradOptimizer(learning_rate)
        elif args.optimizer == 'MOM':
            opt = tf.train.MomentumOptimizer(learning_rate,0.9)
        elif args.optimizer == 'ADAM':
            opt = tf.train.AdamOptimizer(learning_rate, beta1=0.9, beta2=0.999, epsilon=0.1)
        else:
            raise Exception("Not supported optimizer: {}".format(args.optimizer))
        tower_losses = []
        tower_cross = []
        tower_dist = []
        tower_reg= []
        for i in range(args.num_gpus):
            with tf.device("/gpu:" + str(i)):
                with tf.name_scope("tower_" + str(i)) as scope:
                  with slim.arg_scope([slim.model_variable, slim.variable], device="/cpu:0"):
                    with tf.variable_scope(tf.get_variable_scope()) as var_scope:
                        reuse = False if i ==0 else True
                        #with slim.arg_scope(resnet_v2.resnet_arg_scope(args.weight_decay)):
                            #prelogits, end_points =
resnet_v2.resnet_v2_50(batch_image_split[i],is_training=True,
                            #       output_stride=16,num_classes=args.embedding_size,reuse=reuse)
                        #prelogits, end_points = network.inference(batch_image_split[i],
args.keep_probability,
                        #    phase_train=phase_train_placeholder,
bottleneck_layer_size=args.embedding_size,
                        #    weight_decay=args.weight_decay, reuse=reuse)
                        if args.network ==  'sphere_network':
                          prelogits = network.infer(batch_image_split[i],args.embedding_size)
                          print(prelogits)
                        elif args.network == 'resface':
                          prelogits, _ =
resface.inference(batch_image_split[i],1.0,bottleneck_layer_size=args.embedding_size,weigh
t_decay=args.weight_decay,reuse=reuse)
                        elif args.network == 'inception_net':
                          prelogits, endpoints =
inception_net.inference(batch_image_split[i],1,phase_train=True,bottleneck_layer_size=args.
embedding_size,weight_decay=args.weight_decay,reuse=reuse)
                          print(prelogits)

                        elif args.network == 'resnet_v2':
                          with slim.arg_scope(resnet_v2.resnet_arg_scope(args.weight_decay)):
                            prelogits, end_points =
resnet_v2.resnet_v2_50(batch_image_split[i],is_training=True,
                    output_stride=16,num_classes=args.embedding_size,reuse=reuse)
                            prelogits = tf.squeeze(prelogits,axis=[1,2])

                        else:
                          raise Exception("Not supported network: {}".format(args.network))
                        if args.fc_bn:
                            prelogits = slim.batch_norm(prelogits, is_training=True, 
decay=0.997,epsilon=1e-
5,scale=True,updates_collections=tf.GraphKeys.UPDATE_OPS,reuse=reuse,scope='softmax
_bn')
                        if args.loss_type == 'softmax':
                            cross_entropy_mean = utils.softmax_loss(prelogits,batch_label_split[i], 
len(train_set),args.weight_decay,reuse)
                            regularization_losses = 
tf.get_collection(tf.GraphKeys.REGULARIZATION_LOSSES)
                            tower_cross.append(cross_entropy_mean)
                            #loss = cross_entropy_mean + 
args.weight_decay*tf.add_n(regularization_losses)
                            loss = cross_entropy_mean + tf.add_n(regularization_losses)
                            #tower_dist.append(0)
                            #tower_cross.append(cross_entropy_mean)
                            #tower_th.append(0)
                            tower_losses.append(loss)
                            tower_reg.append(regularization_losses)
                        elif args.loss_type == 'cosface':
                            label_reshape = tf.reshape(batch_label_split[i],[single_batch_size])
                            label_reshape = tf.cast(label_reshape,tf.int64)
                            coco_loss = utils.cos_loss(prelogits,label_reshape, 
len(train_set),reuse,alpha=args.alpha,scale=args.scale)
                            #scatter_loss, _ = facenet.coco_loss(prelogits,label_reshape, 
len(train_set),reuse,alpha=args.alpha,scale=args.scale)
                            #coco_loss = scatter_loss['loss_total']
                            regularization_losses = 
tf.get_collection(tf.GraphKeys.REGULARIZATION_LOSSES)
                            if args.network == 'sphere_network':
                    print('reg loss using weight_decay * tf.add_n')
                    reg_loss  = args.weight_decay*tf.add_n(regularization_losses)
                            else:
                    print('reg loss using tf.add_n')
                    reg_loss =
tf.get_collection(tf.GraphKeys.REGULARIZATION_LOSSES)
                            loss = coco_loss + reg_loss

                            tower_losses.append(loss)
                            tower_reg.append(reg_loss)

                        #loss = tf.add_n([cross_entropy_mean] + regularization_losses,
name='total_loss')
                        tf.get_variable_scope().reuse_variables()
              total_loss = tf.reduce_mean(tower_losses)
              total_reg = tf.reduce_mean(tower_reg)
              losses = {}
              losses['total_loss'] = total_loss
              losses['total_reg'] = total_reg

              grads =
opt.compute_gradients(total_loss,tf.trainable_variables(),colocate_gradients_with_ops=True)
              apply_gradient_op = opt.apply_gradients(grads,global_step=global_step)
              update_ops = tf.get_collection(tf.GraphKeys.UPDATE_OPS)
              with tf.control_dependencies(update_ops):
                  train_op = tf.group(apply_gradient_op)

              save_vars = [var for var in tf.global_variables() if 'Adagrad' not in var.name and
'global_step' not in var.name]

              #saver = tf.train.Saver(tf.trainable_variables(), max_to_keep=3)
              saver = tf.train.Saver(save_vars, max_to_keep=3)

              # Build the summary operation based on the TF collection of Summaries.
              summary_op = tf.summary.merge_all()

              # Start running operations on the Graph.
```

```
        gpu_options =
tf.GPUOptions(per_process_gpu_memory_fraction=args.gpu_memory_fraction)
        sess =
tf.Session(config=tf.ConfigProto(gpu_options=gpu_options,allow_soft_placement=True))

        # Initialize variables
        sess.run(tf.global_variables_initializer(), feed_dict={phase_train_placeholder:True})
        sess.run(tf.local_variables_initializer(), feed_dict={phase_train_placeholder:True})

        #sess.run(iterator.initializer)
        sess.run(softmax_iterator.initializer)

        summary_writer = tf.summary.FileWriter(log_dir, sess.graph)
        coord = tf.train.Coordinator()
        tf.train.start_queue_runners(coord=coord, sess=sess)

        with sess.as_default():
            #pdb.set_trace()

            if args.pretrained_model:
                print('Restoring pretrained model: %s' % args.pretrained_model)
                saver.restore(sess, os.path.expanduser(args.pretrained_model))

            # Training and validation loop
            epoch = 0
            while epoch < args.max_nrof_epochs:
                step = sess.run(global_step, feed_dict=None)
                epoch = step // args.epoch_size
                if debug:
                    debug_train(args, sess, train_set, epoch, image_batch_gather,
enqueue_op,batch_size_placeholder,
image_batch_split,image_paths_split,num_per_class_split,
                        image_paths_placeholder,image_paths_split_placeholder,
labels_placeholder, labels_batch,
num_per_class_placeholder,num_per_class_split_placeholder,len(gpus))
            # Train for one epoch
            train(args, sess, epoch,
                learning_rate_placeholder, phase_train_placeholder, global_step,
                losses, train_op, summary_op, summary_writer,
args.learning_rate_schedule_file)

            # Save variables and the metagraph if it doesn't exist already
            save_variables_and_metagraph(sess, saver, summary_writer, model_dir, subdir,
step)

    return model_dir

def train(args, sess, epoch,
        learning_rate_placeholder, phase_train_placeholder, global_step,
        loss, train_op, summary_op, summary_writer, learning_rate_schedule_file):
    batch_number = 0

    if args.learning_rate>0.0:
        lr = args.learning_rate
    else:
        lr = utils.get_learning_rate_from_file(learning_rate_schedule_file, epoch)
    while batch_number < args.epoch_size:
        start_time = time.time()

        print('Running forward pass on sampled images: ', end='')
        feed_dict = {learning_rate_placeholder: lr, phase_train_placeholder: True}
        start_time = time.time()
        total_err, reg_err, _, step = sess.run([loss['total_loss'], loss['total_reg'], train_op,
global_step ], feed_dict=feed_dict)
        duration = time.time() - start_time
        print('Epoch: [%d][%d/%d]\tTime %.3f\tTotal Loss %2.3f\tReg Loss %2.3f, lr %2.5f' %
                (epoch, batch_number+1, args.epoch_size, duration, total_err, reg_err, lr))

        batch_number += 1
    return step


def save_variables_and_metagraph(sess, saver, summary_writer, model_dir, model_name,
step):
    # Save the model checkpoint
    print('Saving variables')
    start_time = time.time()
    checkpoint_path = os.path.join(model_dir, 'model-%s.ckpt' % model_name)
    saver.save(sess, checkpoint_path, global_step=step, write_meta_graph=False)
    save_time_variables = time.time() - start_time
    print('Variables saved in %.2f seconds' % save_time_variables)
    metagraph_filename = os.path.join(model_dir, 'model-%s.meta' % model_name)
    save_time_metagraph = 0
    if not os.path.exists(metagraph_filename):
        print('Saving metagraph')
        start_time = time.time()
        saver.export_meta_graph(metagraph_filename)
        save_time_metagraph = time.time() - start_time
        print('Metagraph saved in %.2f seconds' % save_time_metagraph)
    summary = tf.Summary()
    #pylint: disable=maybe-no-member
    summary.value.add(tag='time/save_variables', simple_value=save_time_variables)
    summary.value.add(tag='time/save_metagraph', simple_value=save_time_metagraph)
    summary_writer.add_summary(summary, step)


def get_learning_rate_from_file(filename, epoch):
    with open(filename, 'r') as f:
        for line in f.readlines():
            line = line.split('#', 1)[0]
            if line:
                par = line.strip().split(':')
                e = int(par[0])
                lr = float(par[1])
                if e <= epoch:
                    learning_rate = lr
                else:
                    return learning_rate


def parse_arguments(argv):
    parser = argparse.ArgumentParser()

    parser.add_argument('--logs_base_dir', type=str,
        help='Directory where to write event logs.', default='logs/facenet_ms_mp')
    parser.add_argument('--models_base_dir', type=str,
        help='Directory where to write trained models and checkpoints.',
default='models/facenet_ms_mp')
    parser.add_argument('--gpu_memory_fraction', type=float,
        help='Upper bound on the amount of GPU memory that will be used by the process.',
default=.9)
    parser.add_argument('--pretrained_model', type=str,
        help='Load a pretrained model before training starts.')
    parser.add_argument('--loss_type', type=str,
        help='Which type loss to be used.',default='softmax')
    parser.add_argument('--network', type=str,
        help='which network is used to extract feature.',default='resnet50')
    parser.add_argument('--data_dir', type=str,
        help='Path to the data directory containing aligned face patches. Multiple directories are
separated with colon.',
        default='~/datasets/casia/casia_maxpy_mtcnnalign_182_160')
    parser.add_argument('--list_file', type=str,
        help='Image list file')
    parser.add_argument('--model_def', type=str,
        help='Model definition. Points to a module containing the definition of the inference
graph.', default='models.inception_resnet_v1')
    parser.add_argument('--max_nrof_epochs', type=int,
        help='Number of epochs to run.', default=500)
    parser.add_argument('--batch_size', type=int,
        help='Number of images to process in a batch.', default=90)
    parser.add_argument('--image_size', type=int,
        help='Image size (height, width) in pixels.', default=160)
    parser.add_argument('--image_src_size', type=int,
        help='Src Image size (height, width) in pixels.', default=256)
    parser.add_argument('--image_height', type=int,
        help='Image size (height, width) in pixels.', default=112)
    parser.add_argument('--image_width', type=int,
        help='Image size (height, width) in pixels.', default=96)
    parser.add_argument('--people_per_batch', type=int,
        help='Number of people per batch.', default=30)
    parser.add_argument('--num_gpus', type=int,
        help='Number of gpus.', default=4)
    parser.add_argument('--images_per_person', type=int,
        help='Number of images per person.', default=5)
    parser.add_argument('--epoch_size', type=int,
        help='Number of batches per epoch.', default=600)
    parser.add_argument('--alpha', type=float,
        help='Margin for cos margin.', default=0.15)
    parser.add_argument('--scale', type=float,
        help='Scale as the fixed norm of weight and feature.', default=64.)
    parser.add_argument('--weight', type=float,
        help='weiht to balance the dist and th loss.', default=3.)
    parser.add_argument('--embedding_size', type=int,
        help='Dimensionality of the embedding.', default=256)
    parser.add_argument('--random_crop',
        help='Performs random cropping of training images. If false, the center image_size
pixels from the training images are used. ' +
        'If the size of the images in the data directory is equal to image_size no cropping is
performed', action='store_true')
    parser.add_argument('--random_flip',
        help='Performs random horizontal flipping of training images.', action='store_true')
    parser.add_argument('--fc_bn',
        help='Wheater use bn after fc.', action='store_true')
    parser.add_argument('--keep_probability', type=float,
        help='Keep probability of dropout for the fully connected layer(s).', default=1.0)
    parser.add_argument('--weight_decay', type=float,
        help='L2 weight regularization.', default=0.0)
    parser.add_argument('--optimizer', type=str, choices=['ADAGRAD', 'ADADELTA',
'ADAM', 'RMSPROP', 'MOM','SGD'],
        help='The optimization algorithm to use', default='ADAGRAD')
    parser.add_argument('--center_loss_factor', type=float,
        help='Center loss factor.', default=0.0)
    parser.add_argument('--center_loss_alfa', type=float,
        help='Center update rate for center loss.', default=0.95)
    parser.add_argument('--learning_rate', type=float,
        help='Initial learning rate. If set to a negative value a learning rate ' +
        'schedule can be specified in the file "learning_rate_schedule.txt"', default=0.1)
    parser.add_argument('--learning_rate_decay_epochs', type=int,
        help='Number of epochs between learning rate decay.', default=100)
    parser.add_argument('--learning_rate_decay_factor', type=float,
        help='Learning rate decay factor.', default=1.0)
    parser.add_argument('--moving_average_decay', type=float,
        help='Exponential decay for tracking of training parameters.', default=0.9999)
    parser.add_argument('--seed', type=int,
        help='Random seed.', default=666)
    parser.add_argument('--learning_rate_schedule_file', type=str,
        help='File containing the learning rate schedule that is used when learning_rate is set to
to -1.', default='data/learning_rate_schedule.txt')


    return parser.parse_args(argv)
```

```
if __name__ == '__main__':
    main(parse_arguments(sys.argv[1:]))
```

Multiprog.py (setup code)

```
# -*- coding: utf-8 -*-
"""

@author:Raja raman
"""

import tkinter as tk
from tkinter import Message ,Text
import cv2,os
import shutil
import csv
import numpy as np
from PIL import Image, ImageTk
import pandas as pd
import datetime
import time
import tkinter.ttk as ttk
import tkinter.font as font

window = tk.Tk()
#helv36 = tk.Font(family='Helvetica', size=36, weight='bold')
window.title("Face_Recogniser")
```

```
dialog_title = 'QUIT'
dialog_text = 'Are you sure?'
#answer = messagebox.askquestion(dialog_title, dialog_text)

#window.geometry('1280x720')
window.configure(background='blue')

#window.attributes('-fullscreen', True)

window.grid_rowconfigure(0, weight=1)
window.grid_columnconfigure(0, weight=1)

#path = "profile.jpg"

#Creates a Tkinter-compatible photo image, which can be used everywhere Tkinter expects
an image object.
#img = ImageTk.PhotoImage(Image.open(path))

#The Label widget is a standard Tkinter widget used to display a text or image on the screen.
#panel = tk.Label(window, image = img)



#panel.pack(side = "left", fill = "y", expand = "no")

#cv_img = cv2.imread("img541.jpg")
#x, y, no_channels = cv_img.shape
#canvas = tk.Canvas(window, width = x, height =y)
#canvas.pack(side="left")
#photo = PIL.ImageTk.PhotoImage(image = PIL.Image.fromarray(cv_img))
# Add a PhotoImage to the Canvas
#canvas.create_image(0, 0, image=photo, anchor=tk.NW)

#msg = Message(window, text='Hello, world!')
```

```
# Font is a tuple of (font_family, size_in_points, style_modifier_string)



message = tk.Label(window, text="Face-Recognition-Based-Attendance-Management-
System" ,bg="Green"  ,fg="white"  ,width=50  ,height=3,font=('times', 30, 'italic bold
underline'))

message.place(x=200, y=20)

lbl = tk.Label(window, text="Enter ID",width=20  ,height=2  ,fg="red"  ,bg="yellow"
,font=('times', 15, ' bold ') )
lbl.place(x=400, y=200)

txt = tk.Entry(window,width=20  ,bg="yellow" ,fg="red",font=('times', 15, ' bold '))
txt.place(x=700, y=215)

lbl2 = tk.Label(window, text="Enter Name",width=20  ,fg="red"  ,bg="yellow"    ,height=2
,font=('times', 15, ' bold '))
lbl2.place(x=400, y=300)

txt2 = tk.Entry(window,width=20  ,bg="yellow"  ,fg="red",font=('times', 15, ' bold ')  )
txt2.place(x=700, y=315)

lbl3 = tk.Label(window, text="Notification : ",width=20  ,fg="red"  ,bg="yellow"  ,height=2
,font=('times', 15, ' bold underline '))
lbl3.place(x=400, y=400)

message = tk.Label(window, text="" ,bg="yellow"  ,fg="red"  ,width=30  ,height=2,
activebackground = "yellow" ,font=('times', 15, ' bold '))
message.place(x=700, y=400)

lbl3 = tk.Label(window, text="Attendance : ",width=20  ,fg="red"  ,bg="yellow"  ,height=2
,font=('times', 15, ' bold  underline'))
```

```
lbl3.place(x=400, y=650)



message2 = tk.Label(window, text="" ,fg="red"   ,bg="yellow",activeforeground =
"green",width=30  ,height=2  ,font=('times', 15, ' bold '))
message2.place(x=700, y=650)

def clear():
    txt.delete(0, 'end')
    res = ""
    message.configure(text= res)

def clear2():
    txt2.delete(0, 'end')
    res = ""
    message.configure(text= res)

def is_number(s):
    try:
        float(s)
        return True
    except ValueError:
        pass

    try:
        import unicodedata
        unicodedata.numeric(s)
        return True
    except (TypeError, ValueError):
        pass

    return False

def TakeImages():
    Id=(txt.get())
    name=(txt2.get())
    if(is_number(Id) and name.isalpha()):
        cam = cv2.VideoCapture(0)
        harcascadePath = "haarcascade_frontalface_default.xml"
        detector=cv2.CascadeClassifier(harcascadePath)
        sampleNum=0
        while(True):
            ret, img = cam.read()
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = detector.detectMultiScale(gray, 1.3, 5)
            for (x,y,w,h) in faces:
                cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
                #incrementing sample number
                sampleNum=sampleNum+1
                #saving the captured face in the dataset folder TrainingImage
                cv2.imwrite("TrainingImage\ "+name +"."+Id +'.'+ str(sampleNum) + ".jpg",
gray[y:y+h,x:x+w])
                #display the frame
                cv2.imshow('frame',img)
            #wait for 100 miliseconds
            if cv2.waitKey(100) & 0xFF == ord('q'):
                break
            # break if the sample number is morethan 100
            elif sampleNum>60:
                break
        cam.release()
        cv2.destroyAllWindows()
        res = "Images Saved for ID : " + Id +" Name : "+ name
        row = [Id , name]
        with open('StudentDetails\StudentDetails.csv','a+') as csvFile:
            writer = csv.writer(csvFile)
            writer.writerow(row)
        csvFile.close()
        message.configure(text= res)
    else:
        if(is_number(Id)):
            res = "Enter Alphabetical Name"
            message.configure(text= res)
        if(name.isalpha()):
            res = "Enter Numeric Id"
            message.configure(text= res)

def TrainImages():
    recognizer = cv2.face_LBPHFaceRecognizer.create()#recognizer =
cv2.face.LBPHFaceRecognizer_create()#$cv2.createLBPHFaceRecognizer()
    harcascadePath = "haarcascade_frontalface_default.xml"
    detector =cv2.CascadeClassifier(harcascadePath)
    faces,Id = getImagesAndLabels("TrainingImage")
    recognizer.train(faces, np.array(Id))
    recognizer.save("TrainingImageLabel\Trainner.yml")
    res = "Image Trained"#+",".join(str(f) for f in Id)
    message.configure(text= res)

def getImagesAndLabels(path):
    #get the path of all the files in the folder
    imagePaths=[os.path.join(path,f) for f in os.listdir(path)]
    #print(imagePaths)

    #create empth face list
    faces=[]
    #create empty ID list
    Ids=[]
    #now looping through all the image paths and loading the Ids and the images
    for imagePath in imagePaths:
        #loading the image and converting it to gray scale
        pilImage=Image.open(imagePath).convert('L')
        #Now we are converting the PIL image into numpy array
```

```
            imageNp=np.array(pilImage,'uint8')
            #getting the Id from the image
            Id=int(os.path.split(imagePath)[-1].split(".")[1])
            # extract the face from the training image sample
            faces.append(imageNp)
            Ids.append(Id)
        return faces,Ids

    def TrackImages():
        recognizer = cv2.face.LBPHFaceRecognizer_create()#cv2.createLBPHFaceRecognizer()
        recognizer.read("TrainingImageLabel\Trainner.yml")
        harcascadePath = "haarcascade_frontalface_default.xml"
        faceCascade = cv2.CascadeClassifier(harcascadePath);
        df=pd.read_csv("StudentDetails\StudentDetails.csv")
        cam = cv2.VideoCapture(0)
        font = cv2.FONT_HERSHEY_SIMPLEX
        col_names =  ['Id','Name','Date','Time']
        attendance = pd.DataFrame(columns = col_names)
        while True:
            ret, im =cam.read()
            gray=cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
            faces=faceCascade.detectMultiScale(gray, 1.2,5)
            for(x,y,w,h) in faces:
                cv2.rectangle(im,(x,y),(x+w,y+h),(225,0,0),2)
                Id, conf = recognizer.predict(gray[y:y+h,x:x+w])
                if(conf < 50):
                    ts = time.time()
                    date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
                    timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
                    aa=df.loc[df['Id'] == Id]['Name'].values
                    tt=str(Id)+"-"+aa
                    attendance.loc[len(attendance)] = [Id,aa,date,timeStamp]

                else:
```

```
                Id='Unknown'
                tt=str(Id)
            if(conf > 75):
                noOfFile=len(os.listdir("ImagesUnknown"))+1
                cv2.imwrite("ImagesUnknown\Image"+str(noOfFile) + ".jpg", im[y:y+h,x:x+w])
            cv2.putText(im,str(tt),(x,y+h), font, 1,(255,255,255),2)
        attendance=attendance.drop_duplicates(subset=['Id'],keep='first')
        cv2.imshow('im',im)
        if (cv2.waitKey(1)==ord('q')):
            break
    ts = time.time()
    date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
    timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
    Hour,Minute,Second=timeStamp.split(":")
    fileName="Attendance\Attendance_"+date+"_"+Hour+"-"+Minute+"-"+Second+".csv"
    attendance.to_csv(fileName,index=False)
    cam.release()
    cv2.destroyAllWindows()
    #print(attendance)
    res=attendance
    message2.configure(text= res)


clearButton = tk.Button(window, text="Clear", command=clear  ,fg="red"  ,bg="yellow"
,width=20  ,height=2 ,activebackground = "Red" ,font=('times', 15, ' bold '))
clearButton.place(x=950, y=200)
clearButton2 = tk.Button(window, text="Clear", command=clear2  ,fg="red"  ,bg="yellow"
,width=20  ,height=2, activebackground = "Red" ,font=('times', 15, ' bold '))
clearButton2.place(x=950, y=300)
takeImg = tk.Button(window, text="Take Images", command=TakeImages  ,fg="red"
,bg="yellow"  ,width=20  ,height=3, activebackground = "Red" ,font=('times', 15, ' bold '))
takeImg.place(x=200, y=500)
trainImg = tk.Button(window, text="Train Images", command=TrainImages  ,fg="red"
,bg="yellow"  ,width=20  ,height=3, activebackground = "Red" ,font=('times', 15, ' bold '))
```

```
trainImg.place(x=500, y=500)
trackImg = tk.Button(window, text="Track Images", command=TrackImages  ,fg="red"
,bg="yellow"  ,width=20  ,height=3, activebackground = "Red" ,font=('times', 15, ' bold '))
trackImg.place(x=800, y=500)
quitWindow = tk.Button(window, text="Quit", command=window.destroy  ,fg="red"
,bg="yellow"  ,width=20  ,height=3, activebackground = "Red" ,font=('times', 15, ' bold '))
quitWindow.place(x=1100, y=500)
copyWrite = tk.Text(window, background=window.cget("background"),
borderwidth=0,font=('times', 30, 'italic bold underline'))
copyWrite.tag_configure("superscript", offset=10)
copyWrite.insert("insert", "Developed by Ashish","", "TEAM", "superscript")
copyWrite.configure(state="disabled",fg="red"  )
copyWrite.pack(side="left")
copyWrite.place(x=800, y=750)

window.mainloop()
```