Blind Person's Assistant

Submitted in partial fulfilment of the requirements

of the Degree of

Bachelor of Engineering

by

Ankita Mandal	D19B 27
Chaitanya Moregaonkar	D19B 34
Aishwarya Patange	D19B 44

Supervisor:

Mr. Mrugendra Vasmatkar

Assistant Professor, EXTC Department

Department of Electronics and Telecommunication

Vivekanand Education Society's Institute of Technology

Mumbai University

2020 – 2021

Project Approval for B.E.

Project entitled **BLIND PERSON’S ASSISTANT** by **ANKITA MANDAL**, **CHAITANYA MOREGAONKAR**, and **AISHWARYA PATANGE** is approved for the degree of Bachelor of Engineering.

Examiners:

1.______________

2.______________

Supervisors:

1.______________

2.______________

Date : ______________

Place: ______________

Certificate

This is to certify that **Ankita Mandal**, **Chaitanya Moregaonkar**, and **Aishwarya Patange** have completed the project report on the topic **Blind Person's Assistant** satisfactorily in partial fulfilment of the requirements for the bachelor's degree of Engineering in Electronics and Telecommunication under the guidance of **Mr. Mrugendra Vasmatkar** during the year **2020-2021** as prescribed by University of Mumbai.

Supervisor	**Head of Department**
Mr. Mrugendra Vasmatkar	**Mrs. Shoba Krishnan**

Principal

Dr. Jayalakshmi Nair

Examiner 1	**Examiner 2**

Declaration

We declare that this written submission represents our ideas in our words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misinterpreted or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

____

(ANKITA MANDAL D19B 27)

____

(CHAITANYA MOREGAONKAR D19B 34)

____

(AISHWARYA PATANGE D19B 44)

Date: ____________________

Acknowledgement

This acknowledgment is a token of gratitude, we feel towards the people who have helped us to make this report a rich experience. We would like to take this opportunity to express our sincere thanks with deep sense of gratitude to our project supervisor **Mr. Mrugendra Vasmatkar**, for his valuable support, co-operation and guidance which helped us in the time of creation of this report. We sincerely thank our H.O.D **Prof. Mrs. Shoba Krishnan** and all our department staff for giving us the valuable suggestions and appraisal.

We are delighted to offer our sincere thanks to our respected **Principal Dr. (Mrs.) J.M Nair** and the teaching and non-teaching staff of **Vivekanand Education Society's Institute of Technology** for permitting to carry out this work in this prestigious institute.

Last but not the least, we are thankful and indebted to all those who helped us directly or indirectly in completion of this project report.

Ankita Mandal

Chaitanya Moregaonkar

Aishwarya Patange

Abstract

People suffering from vision impairment or vision loss face a lot of problems in their day-to-day life. In this project, we come up with a solution that will lessen their problems and make them more self-reliant. Blind Person’s Assistance is a wearable device that will identify faces in front of the user and output the names of the recognized faces in the form of an audio segment. We achieve this by implementing a module that will detect faces (using Multi-task Cascaded Convolutional Networks), a module which will recognize faces (using CNNs), a module that will enable to add new faces to the existing database, and finally a module that will output the audio segment. We study different face detection and recognition algorithms and compare them to understand the tradeoffs between them so that we choose the most efficient algorithm of them all. The results of this study are highlighted in this report.

Keywords: Face Detection, Face Recognition, Multi-task Cascaded Convolutional Networks, Convolutional Neural Networks.

Table of Contents

Table of Figures

Chapter 1

Introduction

Globally the number of people of all ages visually impaired is estimated to be 285 million, of whom 39 million are blind. India, alone, is home to a third of the world's blind population. The country has about 12 million individuals with visual impairment. according to a report published by the National Program for Control of Blindness (NPCB).
The World Health Organization (WHO) says some of the major causes of blindness are refractive errors (43%) and cataract (33%); the first cause of blindness is cataract (51%).

Blind people confront several visual challenges everyday – from reading the menu at a restaurant to navigating to their destination. Along with the visual challenges, blind people are often subjected to discriminations and prejudices. There has been a history of discrimination against blind people in terms of getting quality education or securing employment.

There have been few developments in assistive technology for people with visual impairments. They include GPS enabled walking sticks, radar canes, OCR, Braille keyboards, etc. Most of this equipment is bulky and/or require the user to be quite tech-savvy in order to be able to use them properly.

Chapter 2

Review of Literature

There has been extensive research in the field of face detection. Various image processing as well as machine learning algorithms have been developed for the same. Here, we go through a few major face/object detection algorithms.

Paul Viola and Michael Jones in [1] described a machine learning approach for visual object detection which can detect extremely rapidly and achieve high detection rates. They proposed a new image representation technique called "integral image" which allows the features used by the detector to be computed very quickly. They further proposed a learning algorithm based on AdaBoost which can select a small number of critical features from a large set of features. Finally, they proposed a technique which could combine more complex classifiers in cascade which allows background regions to be discarded quickly so that more computations can be spend on object-like features.

Ahonen et al in [2] study the use of Local Binary Patterns (LBP) to detect and recognize faces. The face area is first divided into small regions from which LBP histograms are extracted and concatenated into a single, spatially enhanced feature histogram efficiently representing the face image. This work considers both shape and texture information to represent faces.

Dalal and Triggs in [3] make the use of Histogram of Oriented Gradients (HOG) to extract features from an image and use the combined feature vector in a conventional SVM based window classifier for human detection. Their approach gives near perfect separation on the original MIT pedestrian database. Although, the original paper used the technique for human detection, it can also be used for face detection tasks.

Hoffman et al in [4] introduce a system for face detection using Discrete Gabor Jets transform. The main idea behind DGJ based detection is to first detect fiducial points (eye corners and mouth corners) and then to detect faces by verifying the distance between fiducial points using a reference graph. This paper also employs a skin detection method to reduce the false positives.

Zhang et al in [5] proposed a framework which adopts a cascaded with three stages of carefully designed deep convolutional networks that predict face and landmark location in a coarse-to-fine manner. In addition, in the learning process, they proposed a new online hard sample mining strategy that can improve the performance automatically without manual sample selection. Their framework essentially uses three different neural networks to detect faces and facial landmark positions. In addition to face detection, they also address the problem of face alignment.

With recent advances in computer vision and related technologies, there has been a need to develop state of the art face recognition systems. Extensive papers have been published on the same.

In [6], *Cao et al* introduce a new large-scale face dataset named VGGFace2. The dataset contains 3.31 million images of 9131 subjects, with an average of 362.6 images for each subject. VGGFace2 consists of a large number of identities and also a large number of images for each identity, large range of poses, ages, and ethnicities. This dataset also minimizes the label noises. To assess face recognition performance using the new dataset, they trained ResNet-50 Convolutional Neural Networks on VGGFace2, on MS-Celeb-1M, and on their union. Training on VGGFace2 lead to improved recognition performance over pose and age.

In [7] *Schroff et al* present a system called FaceNet that directly learns a mapping from face images to a compact Euclidean space where distances correspond to face similarity. Their method uses a deep convolutional network trained to directly optimize the embedding itself, rather than an intermediate bottleneck layer as in other deep learning approaches. They have used triplets of roughly aligned matching / non-matching face patches generated using a novel online triplet mining method. Their system achieved an accuracy of 99.63 on the LFW dataset.

Taigman et al in [8] modify the alignment and representation step involved in face recognition by employing explicit 3D face modelling in order to apply a piecewise affine transformation and derive a face representation from a nine-layer deep neural network. This deep network involves more than 120 million parameters using several locally connected layers without weight sharing rather than the standard convolutional layers.

Wu et al in [9] present a novel deep hashing framework with Convolutional Neural Network for fast person re-identification. It simultaneously learns both CNN features and hash functions/codes to get robust yet discriminative features and similarity-preserving hash codes. A structured loss function defined over positive pairs and hard negatives is proposed to formulate a novel optimization problem so that fast convergence and more stable optimized solution can be obtained.

The other important aspect we needed to consider was the type of processor to use. *A. Alling, N. Powers and T. Soyata* in [10] gave us an in – depth analysis of the amount of processing power needed to preform individual tasks like detection and recognition. It often takes from a few hundred MFLOPs (Floating Point Operations) to a few GFLOPs to process one frame depending on various factors. This amount of processing on processors like the Raspberry Pi might not yield real time results.

Cloutier et al in [11] talk about the how we could use Raspberry Pi clusters to improve the processing capabilities of the board by using two or more of them to perform a single task. They investigated various power-related metrics for seventeen different embedded ARM development boards in order to judge the appropriateness of using them in a computing cluster.

Chapter 3

Block Diagram and Architecture

This chapter covers the block diagram and architecture of our system. The block diagram shows the overall flow of the system. The architecture illustrates the different modules deployed in the system and their respective functions.

3.1 Block Diagram

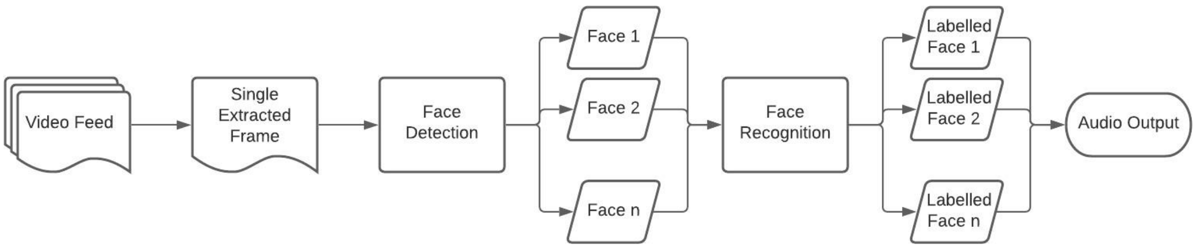


Figure 3. 1 Simplified Block Diagram of the Project

To get a clear understanding of how we plan to tackle the problem mentioned above we shall look at Figure 1. The user will be asked to wear a device which contains the processor, a mobile power supply for the same, a camera and a mono – headset.

The camera will record the capture the video of the everything in front of the user. This video feed will then be fed to a frame extractor which will extract individual frames. The extracted frame will then be passed through a Face Detection algorithm which will detect and localize faces. There can be one or more faces present in the same frame. All these faces detected will then be sent to the Face Recognition module.

The Face Recognition module has the task of comparing the faces sent by the Face Detection module with the pre-existing faces in the database. After the comparison task is done, it will either output the label (or the name of the person whose face is already present in the database) or will output UNKNOWN FACE if the database does not contain the face processed.

In the end, the list of the labelled faces present in the frame will be received by the user in an audio format through the mono-headset.

3.2 Architecture of the System

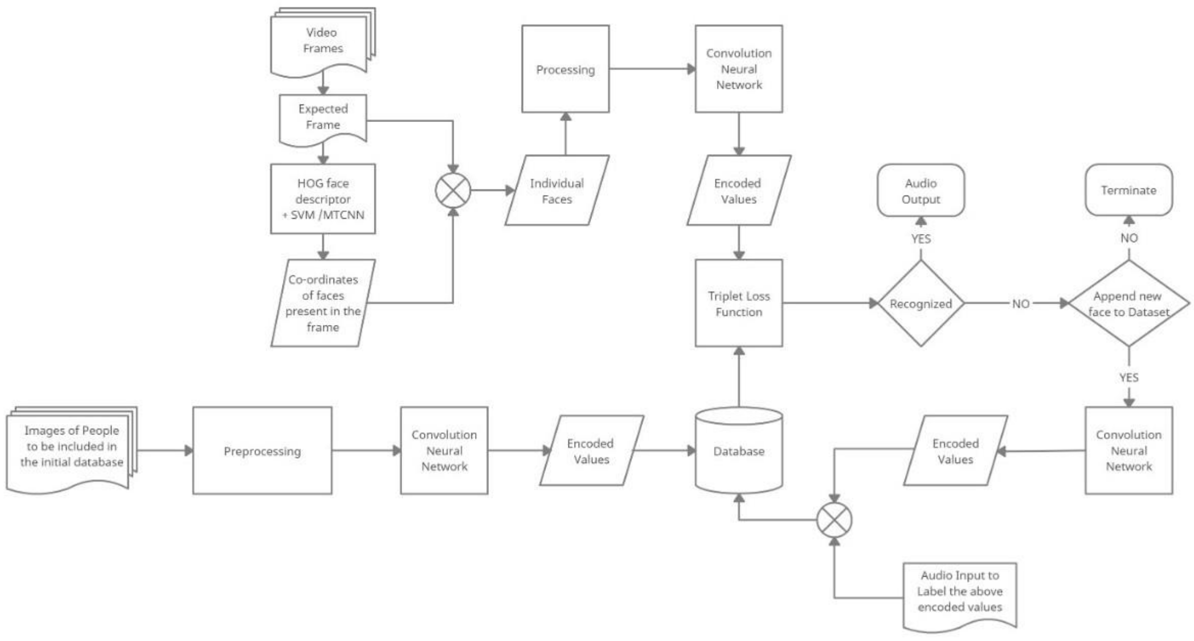

Video Frames
Expected Frame
HOG face descriptor + SVM /MTCNN
Co-ordinates of faces present in the frame
Individual Faces
Processing
Convolution Neural Network
Encoded Values
Triplet Loss Function
Recognized
YES
Audio Output
NO
Append new face to Dataset
NO
Terminate
YES
Convolution Neural Network
Encoded Values
Audio Input to Label the above encoded values
Database
Images of People to be included in the initial database
Preprocessing
Convolution Neural Network
Encoded Values

Figure 3. 2 Architecture of the Project

The system can be divided into three integral modules. The first module deals with detecting and localizing faces in the extracted video frame. This module is loosely based on the system shown in [3] where Histogram of Oriented Gradients for the whole frame are found. A sliding window is used on the bins created and then these bins are fed to a Support Vector Machine which classifies whether the local area fed into it is a face or not based on a threshold. We can eliminate the threshold parameter by simply retaining only the top or highest values obtained from (W'*X + b). This localizes and gives us the co-ordinates of all the faces present in the frame.

An alternative to the above-mentioned detection technique would be to use the MTCNN framework. MTCNN consists of three stages of cascaded neural network, where each stage produces a more refined result than the previous one. Along with detection, this technique also tackles the problem of face alignment. Face detection coupled with face alignment gives rise to a more robust system.

The second module of the system appertains to face recognition. The output from module one activates the second module which consists of a convolutional neural network based on [6] and [7]. Treating this neural network as a black box, the most important part of our approach lies in the end-to-end learning of the whole system. On the input side, we receive localized faces from the first module. While, on the output side, we employ the triplet loss function that directly reflects what we want to achieve in face verification and recognition. We endeavor for an embedding vector f(x), from a face x into a feature space R^d, such that the squared distance between all faces, independent of external conditions of the same identity is small, while different identity is huge. The embedding vector is matched against the pre-existing and labelled embedding vectors present in the database using the manner mentioned above. The convolution neural network mentioned in the black box will be trained on a pre-existing databased consisting of labelled faces which outputs this embedding vector.

The third and the final module of the system concerns the output of the entire system and appending new faces to the existing database. The biggest obstacle this system faces is that it would be used by blind people. The conventional methods to display labels and bounding boxes on the video feed is useless in this case. Hence, we have devised a proprietary output system which yields the output of module two in an audio format. The names which are to be displayed on the screen would be received by the uses through a mono-headset in a FIFO order. To take care of the faces coming in and out of the frame, we have decided to introduce a time threshold. If a new face is detected which is not included in the database, the user would be asked whether he would like to save the new face in the dataset and would also be asked to speak out the name in the dataset so that the face can be labelled.

Chapter 4

Implementation and Hardware

This chapter describes the implementation of the system. The implementation section describes in detail the various algorithms used to carry out the face detection and recognition task. This chapter also covers the hardware requirements of the system.

4.1 Implementation

The implementation is carried out in two parts i.e., face detection and face recognition. Details about the same are highlighted below.

4.1.1 Face Detection

Face detection is the first step for face recognition. Face detection algorithms provide means to detect human faces in a photo or a video.

4.1.1.1 Comparison of Face Detection Algorithms

To get a better understanding of the performance of different face detection algorithms, a comparative study of three different face detection algorithms is carried out. The four algorithms which we used are as following:

- Haar Cascade
- Local Binary Patterns
- Histogram of Oriented Gradients
- Multi-task Cascaded Convolutional Network

4.1.1.1.1 Haar Cascade

In this algorithm Haar-like features are extracted. To speed up the process of extraction the integral image representation has been used. Integral image representation enables us to compute Haar-like features in constant time. Within any image sub window, the total number of Haar-like features is exceptionally large, far larger than the number of pixels. To ensure fast classification, the learning process must exclude a large majority of the available features and focus on a small set of critical features. This is achieved through a simple modification of the AdaBoost procedure. Each stage of the boosting process, which selects a new weak classifier, can be viewed as a feature selection process. We then combine more complex classifiers in a cascade structure which dramatically increases the speed of the detector by focusing attention on promising regions of the image. More complex processing is reserved only for these promising regions. The key behind such an approach is to reduce the false positive rates.

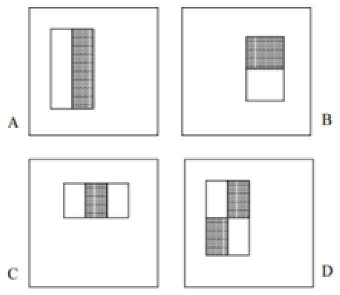


Figure 4. 1 Haar-like Features

4.1.1.1.2 Local Binary Patterns

The original LBP operator was introduced to be used for texture detection. The operator labels the pixels of an image by thresholding the 3x3-neighbourhood of each pixel with the center value and considering the result as a binary number. The binary number of each pixel in the 3x3 neighborhood is concatenated and the resultant binary number is converted into decimal. This decimal value then replaces the original center value. In this histogram, we effectively have a description of the face on three different levels of locality: the labels for the histogram contain information about the patterns on a pixel-level, the labels are summed over a small region to produce information on a regional level and the regional histograms are concatenated to build a global description of the face.

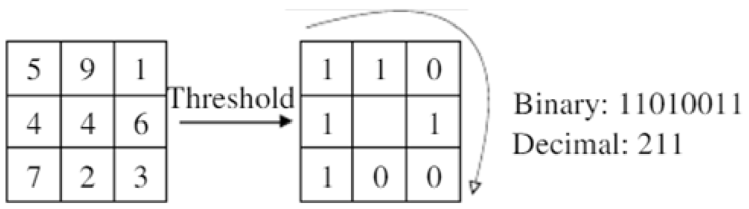


Figure 4. 2 Basic LBP Operator

4.1.1.1.3 Histogram of Oriented Gradients

The method is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. The basic idea is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions. s. In practice this is implemented by dividing the image window into small spatial regions ("cells"), for each cell accumulating a local 1-D histogram of gradient directions or edge orientations over the pixels of the cell. The combined histogram entries from the representation. For better invariance to illumination, shadowing, etc., it is also useful to contrast-normalize the local responses before using them. The combined vectors are fed to a linear SVM for object/non-object classification. The results of these implementations are highlighted and illustrated in Chapter 5. Based on our findings we chose HOG.

4.1.1.1.4 Multi-task Cascaded Convolutional Network

The unique feature of this method is that it considers the inherent correlation between face detection and face alignment. MTCNN is a framework that integrates these two tasks using unified cascaded CNNs by multi-task learning. The CNNs consist of three stages. The first stage consists of a shallow CNN that can produce candidate windows very quickly. The next stage consists of a complex CNN that can refine the candidate windows produced by the first stage by rejecting many false positives. At the final stage, it uses a more powerful CNN that further refines the results and outputs facial landmark positions. In the original paper the authors also propose an online hard sample mining method that is significantly less tedious than the traditional hard sample mining. The authors have carried out extensive experiments on various standard and challenging benchmarks and this method has shown to be more efficient than other state of the art methods in both face detection and alignment tasks.

4.1.1.2 Procedure for Face Detection

The steps involved in two of the four detection algorithms i.e., HOG and MTCNN are discussed below.

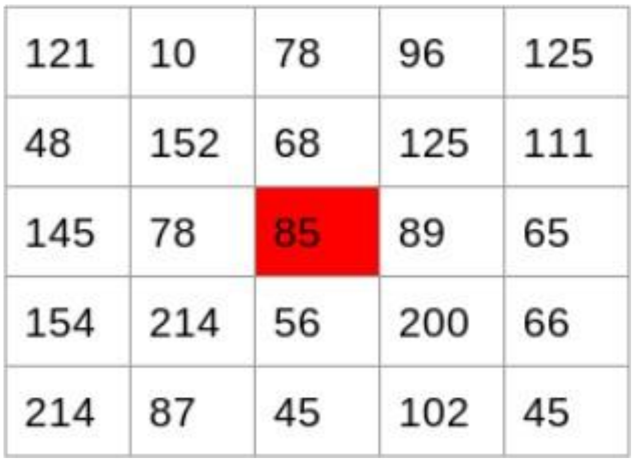
4.1.1.2.1 Histogram of Oriented Gradients

I. **Pre-process the Data**:

We need to preprocess the data to bring the width to height ratio down to 1:1 (i.e., a square image). We, then, apply gamma correction to the downscaled image. This improves performance at low False Positives Per Window (FPPW). We also pad the image so that the information along the edges and corners are not lost.

II. **Calculate the Gradients**:

The next step is to calculate the gradients. Let us assume that an image has the following pixel values:



121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

Figure 4. 3 A 5X5 Grid Considered Calculation the Gradients in HOG

Consider the highlighted pixel. To calculate the horizontal gradient about the highlighted pixel we subtract the pixel to the immediate right of the highlighted pixel from the pixel value on the immediate left:

$$\text{Horizontal gradient } (G_x) = 89 - 78 = 11$$

We do the same for vertical gradient but from top to bottom:

$$\text{Vertical Gradient } (G_y) = 68 - 56 = 8$$

By performing this operation on every pixel, we get two matrices, one for storing vertical gradients and other for horizontal.

III. **Calculate the Resultant Magnitude and Orientation:**

We now calculate the resultant magnitude, G:

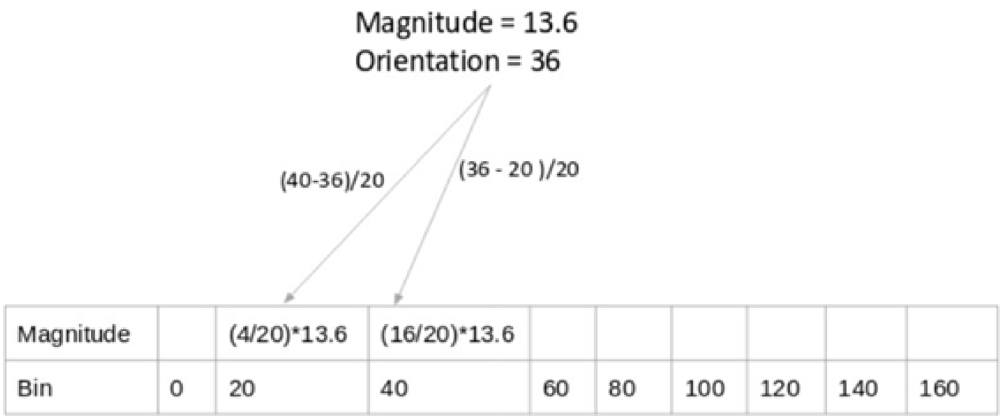
$$G = \sqrt{Gx^2 + Gy^2} = \sqrt{11^2 + 8^2} = 13.6$$

The resultant orientation is calculated as:

$$\Phi = arctan(Gy/Gx) = 36^{\circ}$$

IV. Compute the Histogram:

To compute the histogram, we first choose a bin size and then we fill the bins according to the example shown below:



Magnitude		(4/20)*13.6	(16/20)*13.6						
Bin	0	20	40	60	80	100	120	140	160

We add the contribution of a pixel’s gradient to the bins on either side of the pixel gradient. The higher contribution should be to the bin value which is closer to the orientation.

We divide the whole image into 8X8 patches and compute the histogram of each patch according to the above method. Since, there are 9 bins each 8X8 image patch will have 9X1 vectors associated with it.

V. Normalize the Gradients in 16X16 cells:

We construct 16X16 cells by combining four 8X8 cells together. This step is performed to reduce the sensitivity to light variations. We have seen that each 8X8 patch has a 9x1 vector associated with it. So, a 16x16 cell would have a 36x1 vector associated with it.

To normalize this matrix, we will divide each of these values by the square root of the sum of squares of the values. Mathematically, for a given vector V:

$$V = [a1, a2, a3, \ldots. a36]$$

We calculate the root of the sum of squares:

$$k = \sqrt{a1^2 + a1^2 + a3^2 + \ldots.. + a36^2}$$

And divide all the values in the vector V with this value k:

$$Normalised\ vector = (\frac{a1}{k}, \frac{a2}{k}, \frac{a3}{k}, \ldots. \frac{a36}{k})$$

The resultant would be a normalized vector of size 36×1.

VI. Concatenate the feature vector:

We have computed the features for 16x16 patches of image. We now combine all these features to get the features for the complete image. Finally, the total number of features would be:

Total number of features = 7x7x36 = 1,764

VII. Feed the Feature Vector into a Linear SVM for Detection:

The combined feature vectors are then fed to a Linear SVM for face/non-face classification. Bounding boxes are drawn around the detected faces. If there are more than one bounding box for a single face, non-maximum suppression is applied to merge the overlapping boxes.

4.1.1.2.2 Multi-task Cascaded Convolutional Network

The pipeline of the cascaded framework is illustrated below:

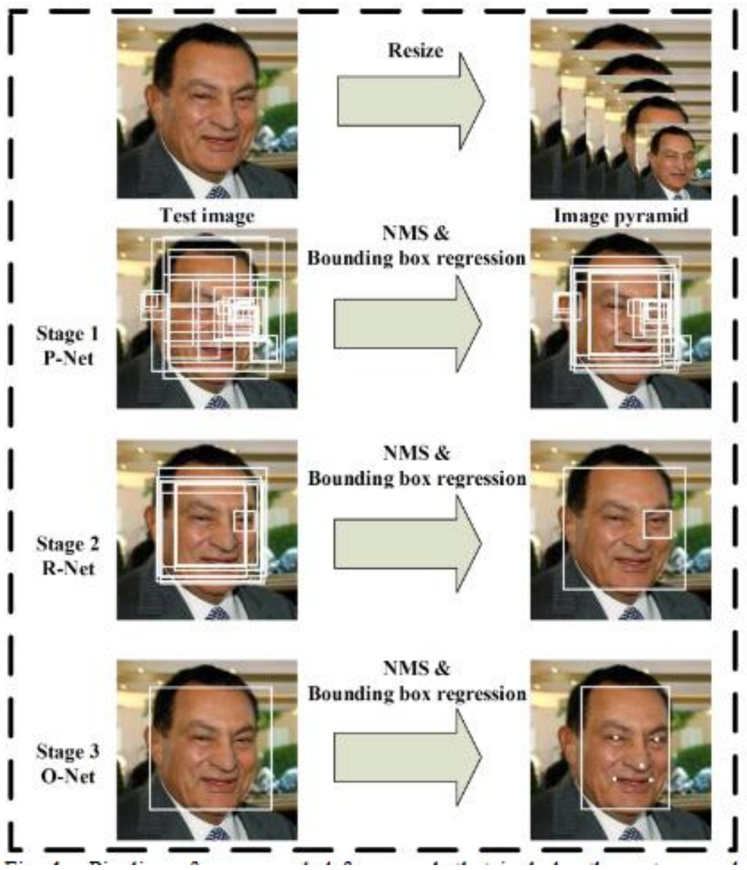

Resize
Test image
Image pyramid
NMS & Bounding box regression
Stage 1 P-Net
NMS & Bounding box regression
Stage 2 R-Net
NMS & Bounding box regression
Stage 3 O-Net

Figure 4. 4 Pipeline of the Cascaded Framework of MTCNN

At first, the input image is resized to different scales to build an image pyramid which is then fed to the input of the three-stage cascaded network.

Stage 1: A fully convolutional network called the Proposal Net (P-net) is used to obtain candidate windows and their bounding box regression vector. The estimated bounding box regression vectors are then used to calibrate the candidates. Non max suppression is employed to merge highly overlapping boxes.

Stage 2: The candidates from stage 1 are fed to another CNN called Refine Net (R-net) which further rejects many false candidates, performs calibration with bounding box regression, and NMS candidate merge.

Stage 3: This stage is the same as the second stage but aims to describe the face in more detail. The network outputs five facial landmarks’ positions.

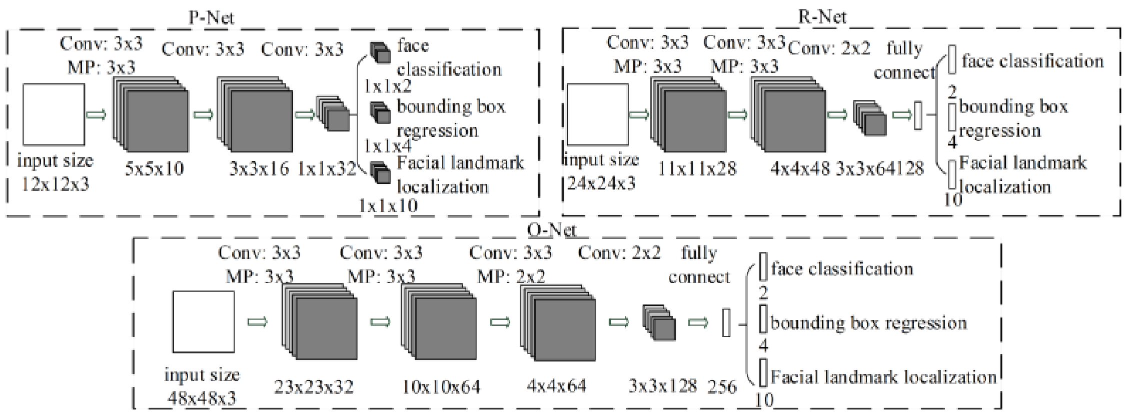


Figure 4. 5 Architecture of P-Net, R-Net, and O-Net of MTCNN

Online hard sample mining: Online hard sample mining is performed which is made adaptive to the training process in contrast to the traditional hard sample mining which is employed after the classifier has been trained. For each mini batch, the loss computed is first sorted and the top 70% from them is selected as hard samples. In the back propagation phase only the gradient from the hard samples is computed. The easy samples are ignored because they are less helpful in strengthening the detector. This technique improves the performance of the detector.

4.1.2 Face Recognition

Face Recognition is the task of recognizing human faces in a digital picture by matching the image with images present in an existing database. If a match is found, the person’s name will be outputted.

4.1.2.1 Comparison of Face Recognition Algorithms

Several face recognition algorithms have been proposed over the years. Two of the most promising ones are discussed below.

4.1.2.1.1 DeepFace

The conventional face recognition pipeline consists of four stages: detect ⇒ align ⇒ represent ⇒ classify. DeepFace attempts to modify both the alignment step and the representation step by employing explicit 3D face modeling in order to apply a piecewise affine transformation and derive a face representation from a nine-layer deep neural network. This deep network involves more than 120 million parameters using several locally connected layers without weight sharing, rather than the standard convolutional layers. It is trained on an identity labeled dataset of four million facial images belonging to more than 4,000 identities. The learned representations coupling the accurate model-based alignment with the large facial database generalize remarkably well to faces in unconstrained environments, even with a simple classifier. This method reaches an accuracy of 97.35% on the Labeled Faces in the Wild (LFW) dataset, reducing the error of the current state of the art by more than 27%, closely approaching human-level performance.

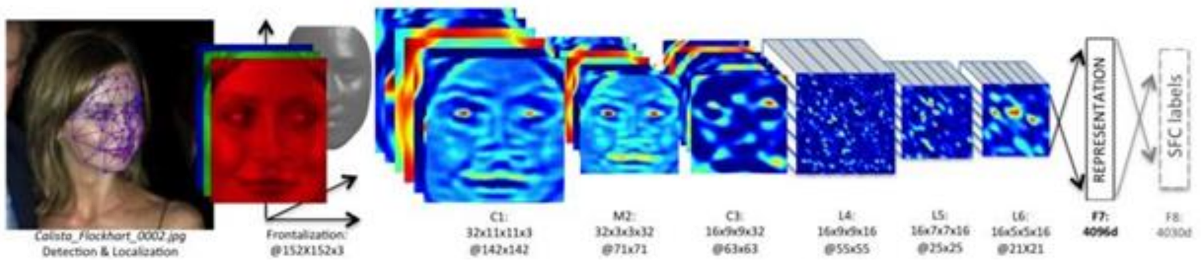


Figure 4. 6 Outline of the DeepFace Architecture

4.1.2.1.2 FaceNet

This method uses a deep convolutional network trained to directly optimize the embedding itself, rather than an intermediate bottleneck layer as in previous deep learning approaches. To train, triplets of roughly aligned matching / non-matching face patches generated using a novel online triplet mining method are used. The benefit of this approach is much greater representational efficiency. State-of-the-art face recognition performance is achieved using only 128-bytes per face. On the widely used Labeled Faces in the Wild (LFW) dataset, our system achieves a new record accuracy of 99.63%.
LFW is used as the common evaluation metric for both architectures. We can see clearly that FaceNet outperforms DeepFace by 2.28%.

4.1.2.2 Procedure of Face Recognition

The model structure, working, triplet loss mechanism, and the architecture of the network have been explained and illustrated below.

4.1.2.2.1 FaceNet

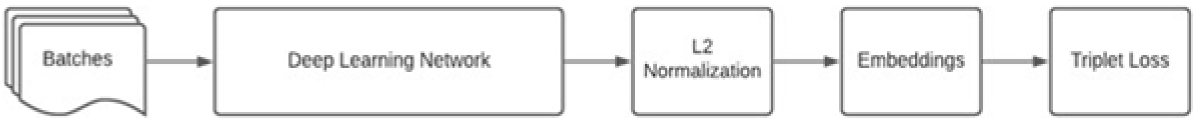


Figure 4. 7 Model Structure for Face Recognition Using FaceNet

FaceNet is a deep neural network used for extracting features from an image of a person's face which is useful for face recognition. FaceNet is based on learning a Euclidean embedding per image using a deep convolutional network. The network is trained such that the squared L2 distances in the embedding space directly correspond to face similarity: small distances imply the face is of the same person and large distance imply that the face is of a different person.
Once this embedding has been produced, then the tasks become straight-forward: face verification simply involves thresholding the distance between the two embeddings; recognition becomes a k-NN classification problem; and clustering can be achieved using k-

means or agglomerative clustering.

Previous face recognition approaches based on deep networks use a classification layer trained over a set of known face identities and then take an intermediate bottleneck layer as a representation used to generalize recognition beyond the set of identities used in training. The downsides of this approach are its indirectness and its inefficiency: one must hope that the bottleneck representation generalizes well to new faces. The representation size per face is usually exceptionally large (1000s of dimensions) when we are using a bottleneck layer.

In contrast to these approaches, FaceNet takes an image of the person's face as input and outputs a vector of 128 numbers which represent the most important features of a face using a triplet-based loss function.

Working:

The main aspect of approach in FaceNet lies in the end-to-end learning. To this end FaceNet employs the triplet loss that directly reflects what we want to achieve in face verification, recognition, and clustering. Namely, we strive for an embedding f(x), from an image x into a feature space R^d, such that the squared distance between all faces, independent of imaging conditions, of the same identity is small, whereas the squared distance between a pair of face images from different identities is large. The triplet loss tries to enforce a margin between each pair of faces from one person to all other faces. This allows the faces for one identity to live on a manifold, while still enforcing the distance and thus discriminability to other identities.

Triplet Loss:

The embedding is represented by $f(x) \in R^d$. It embeds an image x into a d-dimensional Euclidean space. Additionally, we constrain this embedding to live on the d-dimensional hypersphere, i.e., $||f(x)||_2 = 1$. This loss is motivated in the context of nearest-neighbor classification. Here we want to ensure that an image x^a_i (anchor) of a specific person is closer to all other images x^p_i (positive) of the same person than it is to any image x^n_i (negative) of any other persons.

Thus, we want:

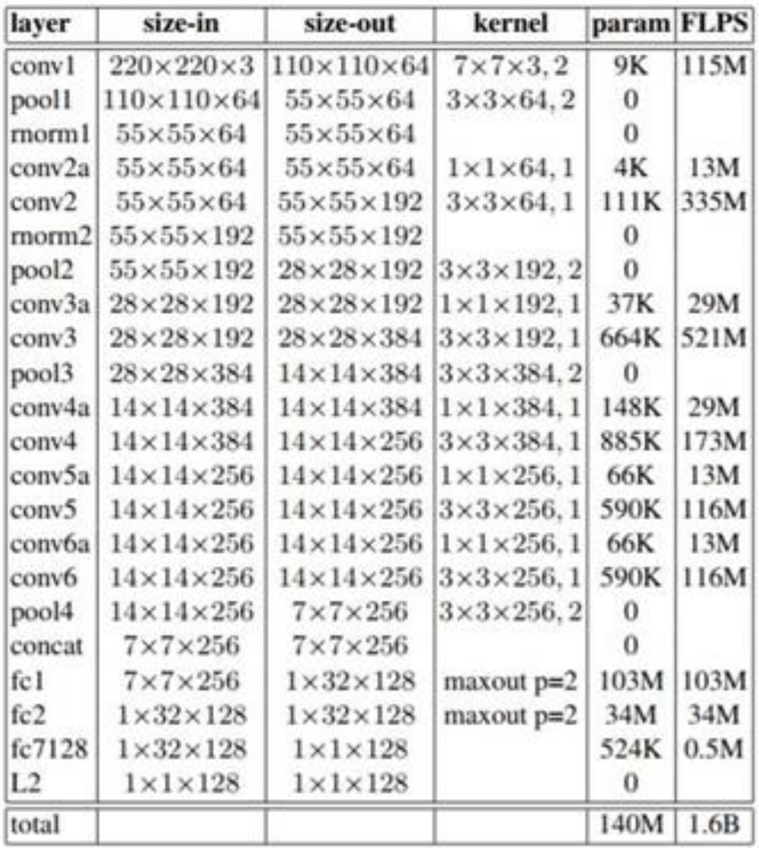
$$\|x_i^a - x_i^p\|_2^2 + \alpha < \|x_i^a - x_i^n\|_2^2, \ \forall \left(x_i^a, x_i^p, x_i^n\right) \in \mathcal{T} \ .$$

where α is a margin that is enforced between positive and negative pairs. T is the set of all possible triplets in the training set and has cardinality N. The loss that is being minimized is then,

L=

$$\sum_{i}^{N} \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+$$

We have discussed trade-offs of two types of architectures which are Zeiler & Fergus and Inception. Their practical differences lie in the difference of parameters and FLOPS. The best model may be different depending on the application. E.g., a model running in a data center can have many parameters and require many FLOPS, whereas a model running on a mobile phone needs to have few parameters, so that it can fit into memory.



layer	size-in	size-out	kernel	param	FLPS
conv1	220×220×3	110×110×64	7×7×3, 2	9K	115M
pool1	110×110×64	55×55×64	3×3×64, 2	0	
rnorm1	55×55×64	55×55×64		0	
conv2a	55×55×64	55×55×64	1×1×64, 1	4K	13M
conv2	55×55×64	55×55×192	3×3×64, 1	111K	335M
rnorm2	55×55×192	55×55×192		0	
pool2	55×55×192	28×28×192	3×3×192, 2	0	
conv3a	28×28×192	28×28×192	1×1×192, 1	37K	29M
conv3	28×28×192	28×28×384	3×3×192, 1	664K	521M
pool3	28×28×384	14×14×384	3×3×384, 2	0	
conv4a	14×14×384	14×14×384	1×1×384, 1	148K	29M
conv4	14×14×384	14×14×256	3×3×384, 1	885K	173M
conv5a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv5	14×14×256	14×14×256	3×3×256, 1	590K	116M
conv6a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv6	14×14×256	14×14×256	3×3×256, 1	590K	116M
pool4	14×14×256	7×7×256	3×3×256, 2	0	
concat	7×7×256	7×7×256		0	
fc1	7×7×256	1×32×128	maxout p=2	103M	103M
fc2	1×32×128	1×32×128	maxout p=2	34M	34M
fc7128	1×32×128	1×1×128		524K	0.5M
L2	1×1×128	1×1×128		0	
total				140M	1.6B

Figure 4. 8 Table of Architecture of FaceNet – NN1

The above table (1) shows, 1×1×d convolutional layers, between the standard convolutional layers of the Zeiler & Fergus architecture and results in a model 22 layers deep. It has a total of 140 million parameters and requires around 1.6 billion FLOPS per image.

4.2 Hardware Specifications

The required hardware, their specifications and comparisons have been highlighted below.

4.2.1 Processor

Selection of the right processor is of utmost importance. The chosen processor should have appropriate memory, speed as well as should be compatible with our software requirements. The specifications of two shortlisted processors are listed below.

4.2.1.1 Raspberry Pi 4 Model B

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 2GB, 4GB or 8GB LPDDR4-3200 SDRAM (depending on model)
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
- 4-pole stereo audio and composite video port
- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
- OpenGL ES 3.0 graphics
- Micro-SD card slot for loading operating system and data storage

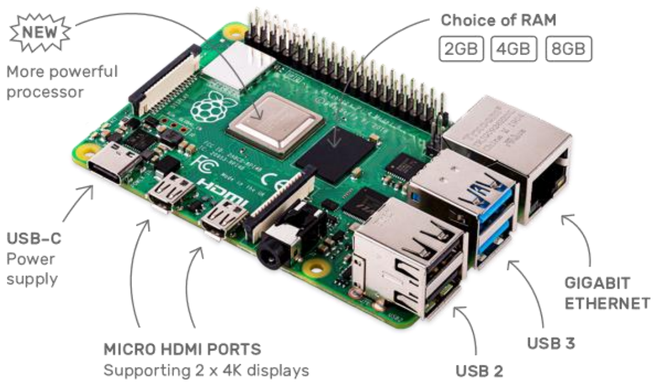


Figure 4. 9 Raspberry Pi 4B

4.2.1.2 Nvidia Jetson Nano

- NVIDIA Maxwell architecture with 128 NVIDIA CUDA® cores GPU
- Quad-core ARM Cortex-A57 MPCore processor
- 4 GB 64-bit LPDDR4 RAM
- 16 GB eMMC 5.1 Storage

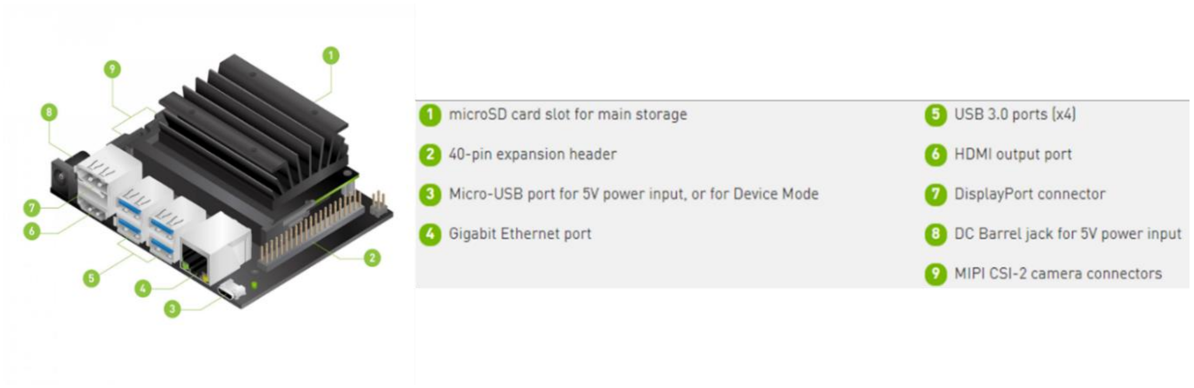

1 microSD card slot for main storage
2 40-pin expansion header
3 Micro-USB port for 5V power input, or for Device Mode
4 Gigabit Ethernet port
5 USB 3.0 ports (x4)
6 HDMI output port
7 DisplayPort connector
8 DC Barrel jack for 5V power input
9 MIPI CSI-2 camera connectors

Figure 4. 10 Nvidia Jetson Nano

4.2.2 Camera

An ideal camera should provide appropriate resolution which can satisfy the requirements of the detection and recognition network. Since, our device would be wearable, the camera should be light weight. The camera module which we have shortlisted is highlighted below.

4.2.2.1 Raspberry Pi HQ Camera

It offers higher resolution (12 megapixels, compared to 8 megapixels), and sensitivity (approximately 50% greater area per pixel for improved low-light performance) than the existing Camera Module v2.

Specifications:

- Sony IMX477R stacked, back-illuminated sensor, 12.3 megapixels, 7.9 mm sensor diagonal, 1.55 μm × 1.55 μm pixel size.
- Back focus: Adjustable (12.5 mm–22.4 mm).
- Lens standards: C-mount, CS-mount (C-CS adapter included).
- IR cut filter: Integrated.

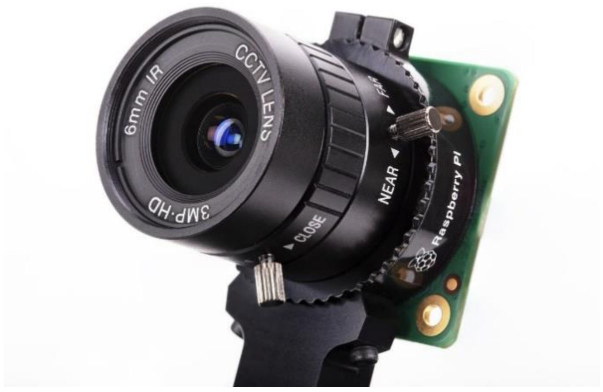


Figure 4. 11 Raspberry Pi HQ Camera

4.2.2.2 6mm Wide Angle Lens

This lens is suitable for basic photography. It can also be used for macro photography because it can focus on objects at noticeably short distances.

Chapter 5

Results and Timeline

This chapter highlights the timeline and the results achieved. The timeline describes in detail all the work that we have carried out over the course of our academic year. In the results section we have highlighted the performance of different detection algorithms and the performance of HOG and MTCNN when paired with the recognition module.

5.1 Results

These tests were run on a Personal Computer with the following specifications:

- AMD Ryzen 3500 Processor
- Nvidia GTX 1650 4GB
- 16 GB 3000 MHz RAM

5.1.1 Face Detection

To finalize our Face Detection algorithm, we ran a few pre-existing modules which deal with them.The four algorithms we chose for testing was Haar Cascading [1], Local Binary Pattern Cascading [2], Histogram of Oriented Gradients [3], and Multi-Task Cascaded Convolutional Networks [5].

The processors which we are going to deploy in our project would be considerably slower. Hence,we had to pick the algorithm with the highest frame rate. This is because frame rate here is inversely proportional to the amount of processing power required to process a frame for detectinga face.

We found out that HOG Face Detection was the most accurate and had the highest frame rate averaging between 39 – 44 frames per second. This was followed closely by the LBP Cascade Detector which had frame rates comparable to the HOG Algorithm, but it would falsely detect other objects as faces once every 500 frames. Haar – Cascading proved to be the least efficient onewith frame rates not even reaching 30 frames per second.

The major drawback of the HOG Face Detection Algorithm was the fact that it was not able to detect faces titled at one angle or would detect cartoon faces are human faces at certain times. To overcome this, we decided to use the MTCNN Face Detection Algorithm.
MTCNN seemed to be the most reliant face detection algorithm at the expense of its slightly higher processing cost. It was able to detect tilted faces thanks to its landmark detection as well as differentiate between actual human and cartoon faces. We were getting a steady framerate of 2-3 frames per second for the MTCNN face detection algorithm.
Hence, we decided to go with two of the four algorithms – Histogram of Oriented Gradients and Multi-Task Cascaded Convolution Network.
The results of the four algorithms are given below along with their frame rates:

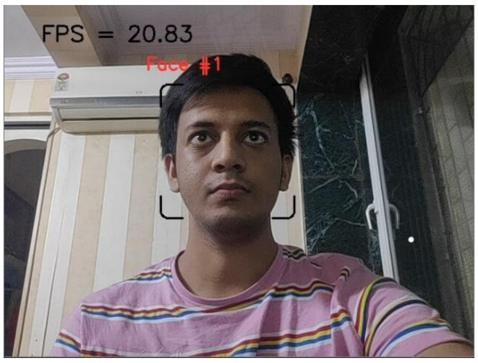


Figure 5. 1 Face Detection Using Haar Cascade

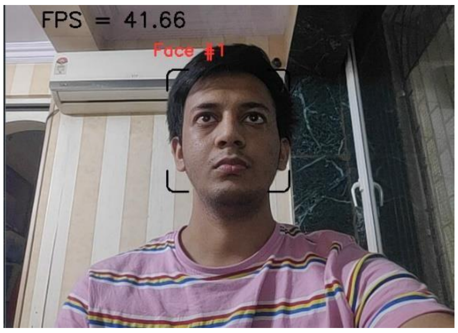


Figure 5. 2 Face Detection Using LBP

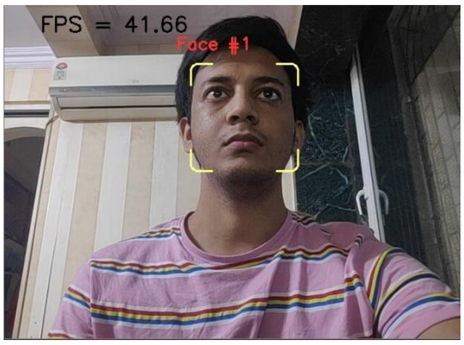


Figure 5. 3 Face Detection Using HOG

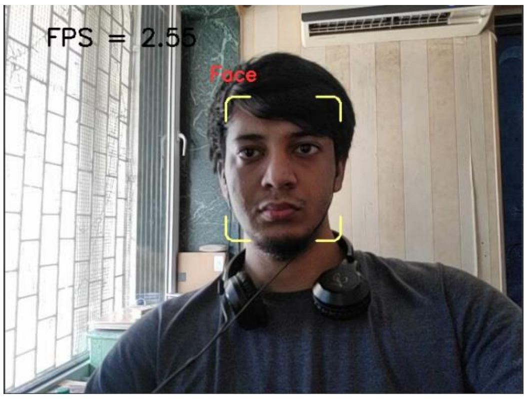
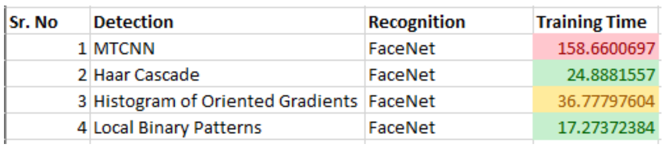


Figure 5. 4 Face Detection Using MTCNN

5.1.2 Training FaceNet

The next part of our implementation included interfacing the face detection algorithms with the face recognition algorithm. The recognition algorithm we went ahead with was the FaceNet algorithm. It stored embeddings of the provided face in a 128-dimensional space and then used Cosine Distance to differentiate between the faces present in the database.
We trained the FaceNet on our custom dataset containing 33 faces using the four face detection algorithms. The training time for the different algorithms is given below.



Sr. No	Detection	Recognition	Training Time
1	MTCNN	FaceNet	158.6600697
2	Haar Cascade	FaceNet	24.8881557
3	Histogram of Oriented Gradients	FaceNet	36.77797604
4	Local Binary Patterns	FaceNet	17.27372384

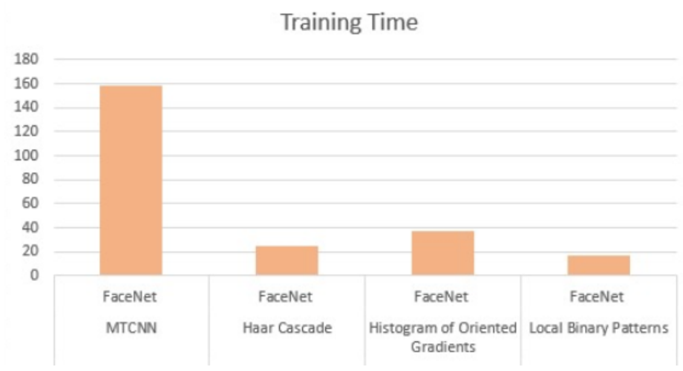
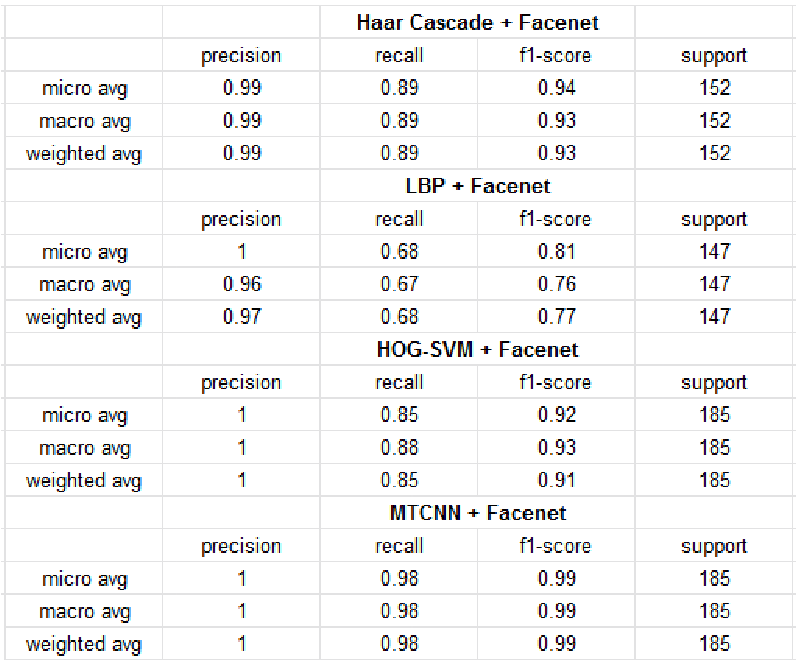


Figure 5. 5 Training Time Requirements for Face Detection Coupled with FaceNet

Here, we found out that the training time required was almost directly proportional to the merit of the face detection algorithm.

5.1.3 Comparison between the Face Detection Algorithms

The precision, recall, and the F1-scores of the four algorithms on their training data set is given below:



		Haar Cascade + Facenet		
	precision	recall	f1-score	support
micro avg	0.99	0.89	0.94	152
macro avg	0.99	0.89	0.93	152
weighted avg	0.99	0.89	0.93	152
		LBP + Facenet		
	precision	recall	f1-score	support
micro avg	1	0.68	0.81	147
macro avg	0.96	0.67	0.76	147
weighted avg	0.97	0.68	0.77	147
		HOG-SVM + Facenet		
	precision	recall	f1-score	support
micro avg	1	0.85	0.92	185
macro avg	1	0.88	0.93	185
weighted avg	1	0.85	0.91	185
		MTCNN + Facenet		
	precision	recall	f1-score	support
micro avg	1	0.98	0.99	185
macro avg	1	0.98	0.99	185
weighted avg	1	0.98	0.99	185

Figure 5. 6 Evaluation Metrics on the Training Data

As we can see, LBP was able to detect the least number of faces from the frames provided for it to train followed closely by the Haar Cascade algorithm. These two were the worst in detecting the faces amongst the four algorithms.
The HOG as well as the MTCNN face detectors were able to detect all the 185 faces, but the MTCNN enjoyed a greater recall and f1-score. Although the face provided by the HOG face detector were usable, MTCNN gave us more accurate faces which could be sent to the FaceNet for recognition purposes.
Moreover, the main metric which concerned us was the precision. Precision is a good measure to determine when the costs of False Positive is high. In our project, a false positive means that a face which is not present in the dataset (actual negative) has been identified as face present in the dataset. Having a higher cost associated with false negatives meant it was better for a model to not detect faces rather that assigning a false label to a face not present in the database.

5.1.4 Face Recognition

5.1.4.1 Frame Rates

The framerates of the models after the faces have been recognized was solely dependent upon the framerates of the detection algorithm thanks to the fast FaceNet algorithm. Comparing the embeddings was far faster than comparing individual faces by feeding them in neural networks. Given below is the difference between the HOG + FaceNet as well as MTCNN + FaceNet in real time detection:

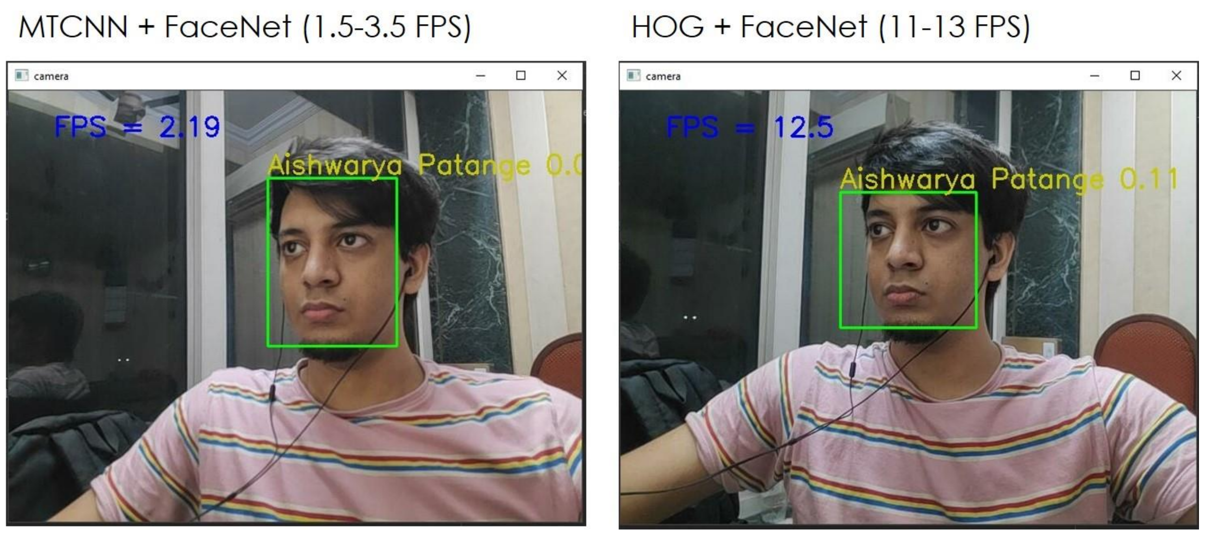


Figure 5. 7 Framerate Comparison between HOG + FaceNet vs MTCNN + FaceNet

As we can see, the HOG + FaceNet provided a faster recognition speed while the MTCNN + FaceNet was much slower bottlenecked by the time required to detect a face using the two detection algorithms.

5.1.4.2 Performance on Test Data

We created a custom test dataset consisting of 50 different frames and 95 faces present in those images to detect and recognize. We have given a few examples of the images present along with the predictions made by the two models to give you a better understanding about the differences between MTCNN and HOG detectors.

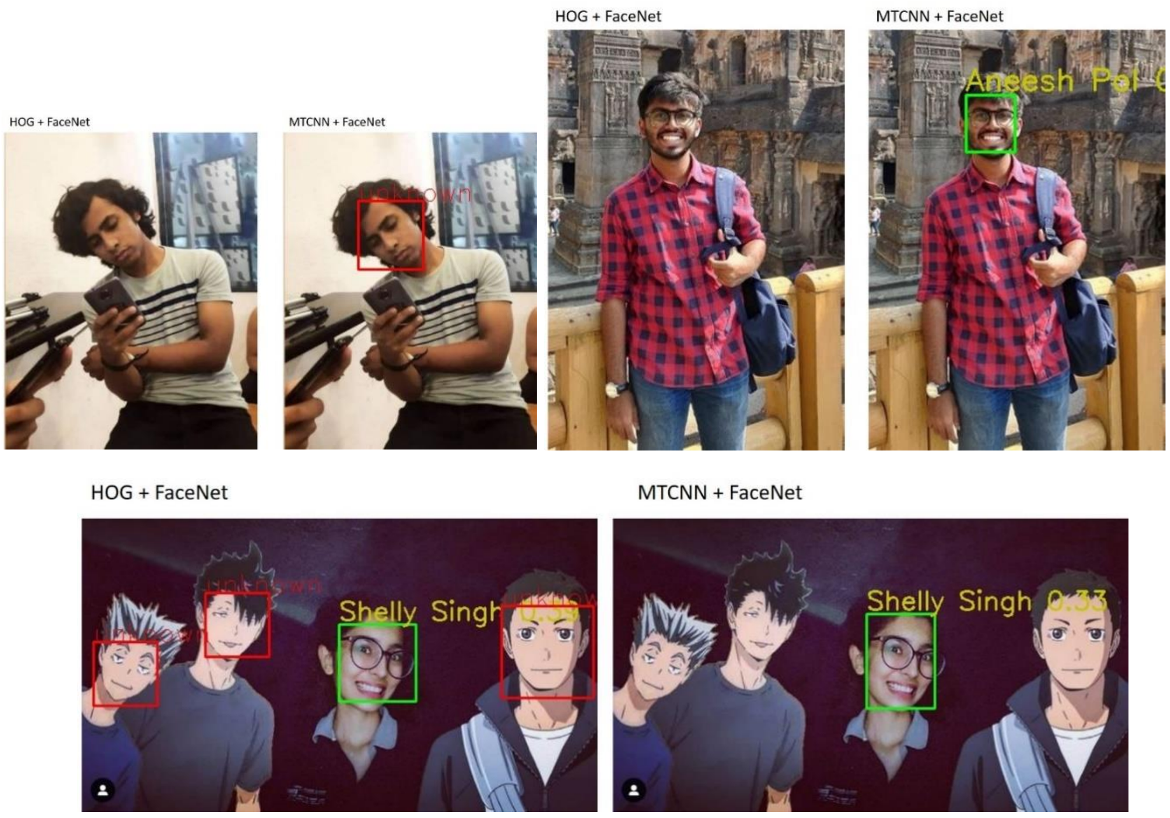


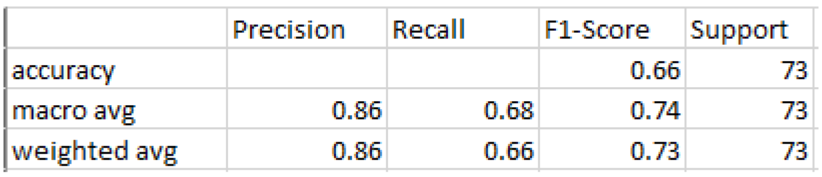
Figure 5. 8 Performance of HOG + FaceNet vs MTCNN + FaceNet on Test Data

Although HOG was much faster than MTCNN, MTCNN was much more accurate at detecting tilted faces, differentiating between actual faces and cartoon faces and proved to be much more detecting faces which did not have a stark contrast between their complexion and background color (mostly because HOG detector uses a single channel image compared to a three-channel image used by the MTCNN, and it is harder to detect a face in a single channel image).

5.1.4.3 Evaluation Metrics and Confusion Matrices on Test Data

The metrics and the confusion matrices for the two algorithms are given below:

5.1.4.3.1 HOG + FaceNet



	Precision	Recall	F1-Score	Support
accuracy			0.66	73
macro avg	0.86	0.68	0.74	73
weighted avg	0.86	0.66	0.73	73

Figure 5. 9 Evaluation Metrics of HOG + FaceNet on Test Data

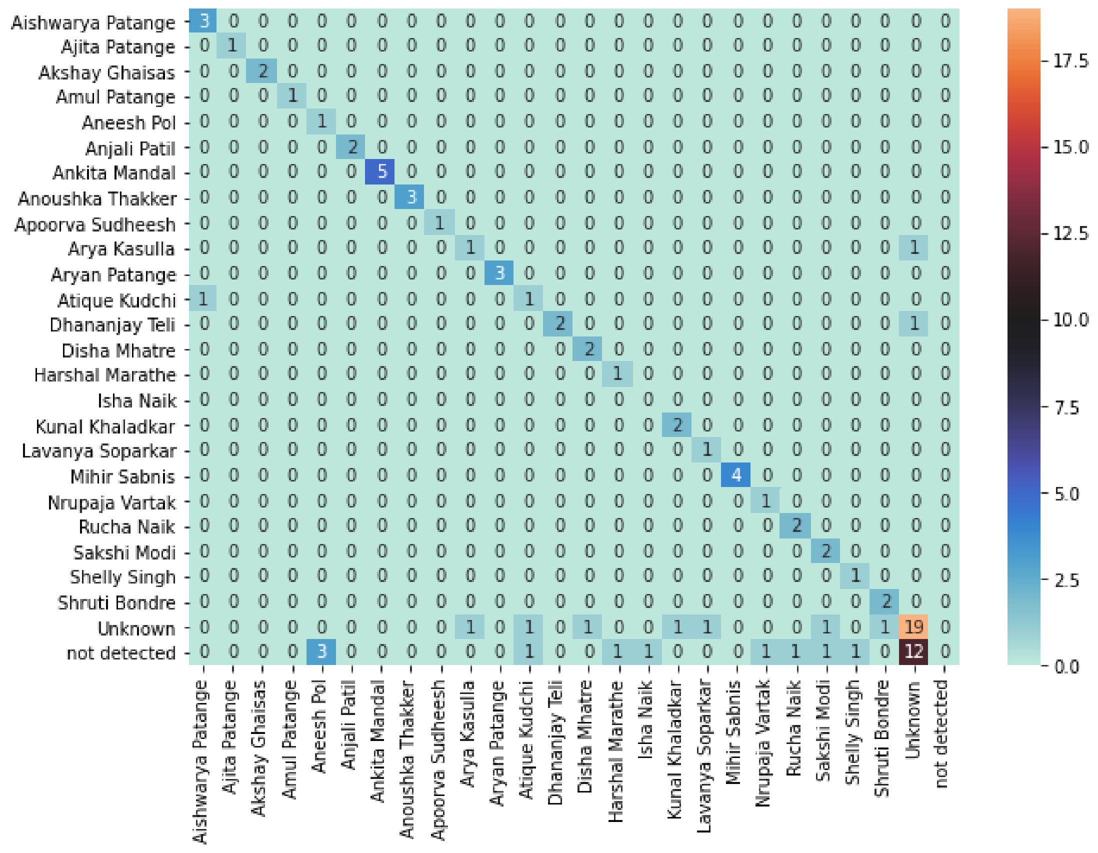


Figure 5. 10 Confusion Matrix of HOG + FaceNet on Test Data

The HOG + FaceNet method was unable to detect 22 faces present in the test frames. Although the detected faces were good enough for the FaceNet to recognize the person, the sheer lack of robustness is a big issue with the HOG face detection method.

5.1.4.3.2 MTCNN + FaceNet

	Precision	Recall	F1-Score	Support
accuracy	0.88	0.88	0.88	95
macro avg	0.89	0.79	0.82	95
weighted avg	0.9	0.88	0.87	95

Figure 5. 11 Evaluation Metrics of MTCNN + FaceNet on Test Data

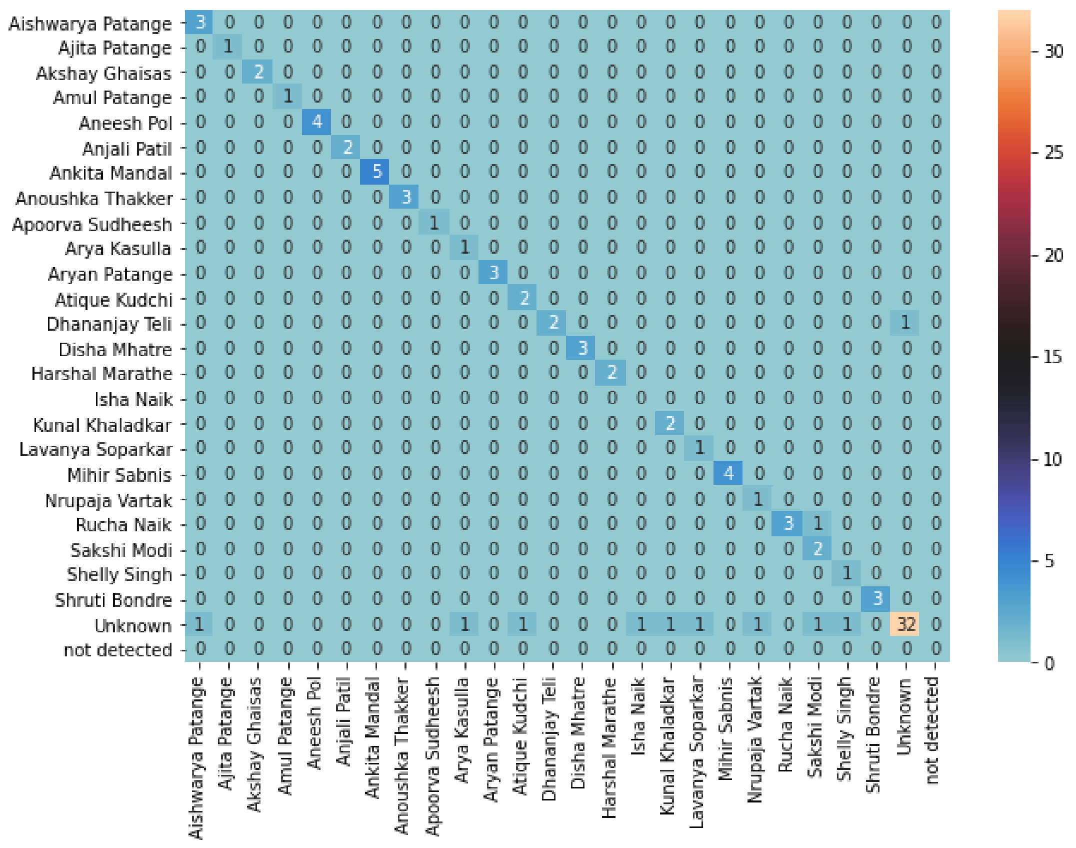


Figure 5. 12 Confusion Matrix of MTCNN + FaceNet on Test Data

Although the MTCNN + FaceNet method is more accurate than the previous method, the sheer amount of processing it takes to detect faces in a frame is a lot more than the previous method. Hence, there is a robustness vs processing power tradeoff between the two methods when it comes to recognizing faces. One should first assess which one of the two parameters is more important when it comes to use these methods in their applications.

5.2 Timeline

- In **December 2019**, we defined the problem statement, and we went through different technical papers on face detection.
- From January 2020 to April 2020, we conducted an exhaustive survey of all the different facial detection and recognition algorithms.
- From **April 2020 to June 2020**, we acquired the required skills such as the working of neural networks especially CNNs. We also gained sufficient knowledge about various image processing and machine learning libraries.

- From **July 2020 to September 2020**, we started implementing four shortlisted detection algorithms namely Haar Cascade, LBPs, HOG and MTCNN and compared their performance side by side. This was done by using already trained classifiers available on the internet. We also finalised HOG and MTCNN for detection based on their overall efficiency and performance. Selection of processor and camera and other peripheral devices were also done.
- From **October 2020 to February 2021**, we prepared our own HOG classifiers. We also started working on SVM and got the accuracy of 98.82% on the LFW dataset. Compilation of the images was started to make our own dataset.
- In the months of **February and March 2021**, we updated the custom dataset with wider range of negative patches. We worked on a Sliding Window Algorithm to feed patches into SVM. Then, we tuned the SVM on the custom dataset and implemented negative hard mining to reduce false positives. Simultaneously, we started working on CNNs for facial recognition on available datasets.
- In the month of **April 2021**, we studied and implemented a new face detection algorithm i.e., MTCNN. We implemented the combination of the four detection algorithms along with the recognition module. We came up with a comparative report of the same.
- The network could be optimised according to the needs and specifications of the processor. The device could be made portable as well as wearable. More features such as object detection could be added.

Chapter 6

Conclusion and Future Scope

We have summarized and concluded our observations in this chapter. This chapter also states the future scope of the project. We have highlighted different techniques that could further improve the performance of the system. We have also suggested a few features that could make the system more versatile.

6.1 Conclusion

- In this project we designed a device which helps a visually impaired person to recognize an individual in front of him. This device will not only make the user more self-reliant but also will safeguard him against fraud.
- While going through the various face detection and face verification algorithms and interfacing them with each other, we made a few observations which helped us determine the best possible couple amongst them.
- The face verification part was the easiest to finalize. FaceNet creates a 128-dimensional embedding from images and inserts them into a feature space, in such a way, that the squared distance between all faces, regardless of the imaging conditions, of the same identity, is small, whereas the cosine distance between a pair of face images from distinct characters is large. It always provided reliable predictions.
- The part which harder to determine was the face detection algorithm. The Haar Cascade and the Local Binary Patterns were an obsolete and less reliable as compared to Histogram of Oriented Gradients and Multi-task Cascaded Convolutional Network, and hence they were discarded off at an early stage. The usage of HOG vs MTCNN was met with a tradeoff between robustness v/s processing power required to detect an individual face.

6.1.1 Advantages of MTCNN over HOG Face Detection

The MTCNN proved to be much more robust than the HOG face detection method in the following ways:

- It was able to detect faces in frames with much lower resolution.
- It was to detected faces that were tilted at an angle.
- It was able to detect faces when the complexion and the background had a similar color palette.
- It was able to distinguish between human faces and cartoon faces.

6.1.2 Advantages of HOG Face Detection over MTCNN

The advantages of MTCNN over the HOG face detector came at a high cost of processing power. These are few of the main reasons:

- Ten times faster in terms of frame rate as compared to MTCNN + FaceNet.
- Since it operates on localized cells, the method maintains the invariance to geometric and photometric transformations.

Hence, we concluded that the usage of either one should be based on a thorough assessment of which one of the two parameters is more important and only then finalize the face detecting algorithm.

6.1.3 Shortcomings of the Project

Although our model is deployable for real world usage, there are certain shortcomings which it still must overcome, them being:

- Although the model was able to detect tilted faces after using MTCNN, it still proved to be a hard task for the verification algorithm to work on it.
- The device will only work in a well-lit area as infrared face detection has not been integrated in the model.

6.2 Future Scope

Below are mentioned a few ways in which we can upgrade our device in the future. The main issue faced at this stage to integrate the following ideas is the lack of processing power in a compact processor at a feasible price.

6.2.1 Alignment of the Face Before Verification

In the current stage of our project, the model still struggles at verifying faces tilted at an angle. We could add a face alignment module after detection of the faces in a frame so that the verification neural network always receives faces aligned at the right angle. Alignment of the faces can be inferred from the landmarks detected by the MTCNN algorithm and this data can be used to rotate the face to a much more suitable angle.

6.2.2 Usage of YOLO for Object Detection

We could also help the user to identify object in front of them by using the YOLO (You Only Look Once) algorithm. This can be used to find missing objects as well after integrating it to a suitable tagging algorithm tagging algorithm.

6.2.3 Usage of a Grid to determine the Relative Position of the Object / Person

We could assign a grid system to the frames which the camera inputs and hence determine where exactly the object / person is with respect to the user. This will take some getting used to by the user, but in turn will increase the usability of the device exponentially.

6.2.4 Obstacle Avoidance

The YOLO algorithm can be also used to detect if there are any obstacles present in the path of the user. If a feedback mechanism is integrated to this device, it can possibly work as a virtual walking stick and find the most optimum path to walk on for the user.

References

[1] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Kauai, HI, USA, 2001, pp. I-I, doi: 10.1109/CVPR.2001.990517.

[2] T. Ahonen, A. Hadid and M. Pietikainen, "Face Description with Local Binary Patterns: Application to Face Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 12, pp. 2037-2041, Dec. 2006, doi: 10.1109/TPAMI.2006.244.

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005, pp. 886-893 vol. 1, doi: 10.1109/CVPR.2005.177.

[4] Hoffmann, Ulrich & Naruniec, Jacek & Yazdani, Ashkan & Ebrahimi, Touradj. (2008). Face Detection using Discrete Gabor Jets and Color Information. SIGMAP 2008 - Proceedings of the International Conference on Signal Processing and Multimedia Applications. 76-83.

[5] Zhang, K., Zhang, Z., Li, Z., and Qiao, Y., "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks", IEEE Signal Processing Letters, vol. 23, no. 10, pp. 1499–1503, 2016. doi:10.1109/LSP.2016.2603342.

[6] Q. Cao, L. Shen, W. Xie, O. M. Parkhi and A. Zisserman, "VGGFace2: A Dataset for Recognizing Faces across Pose and Age," 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), Xi'an, 2018, pp. 67-74, doi: 10.1109/FG.2018.00020.

[7] F. Schroff, D. Kalenichenko and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 815-823, doi: 10.1109/CVPR.2015.7298682.

[8] Y. Taigman, M. Yang, M. Ranzato and L. Wolf, "DeepFace: Closing the Gap to Human-

Level Performance in Face Verification," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, 2014, pp. 1701-1708, doi: 10.1109/CVPR.2014.220.

[9] Wu, Lin & Wang, Yang. (2017). Structured Deep Hashing with Convolutional Neural Networks for Fast Person Re-identification.

[10] A. Alling, N. Powers and T. Soyata. “Face Recognition: A Tutorial on Computational Aspects”. Emerging Research Surrounding Power Consumption and Performance Issues in Utility Computing. IGI Global. October 2015. pp. 405-425

[11] Cloutier, Michael & Paradis, Chad & Weaver, Vincent. (2016). “A Raspberry Pi Cluster Instrumented for Fine-Grained Power Measurement”. 2014 First International Workshop on Hardware-Software Co-Design for High Performance Computing Electronics. 5. 61. 10.3390/electronics5040061.