REAL TIME FACE RECOGNITION AND TRACKING SYSTEM USING DEEP LEARNING

A PROJECT REPORT

Submitted by

SHASHANK S. CHANDEL [Reg No: RA1511008010182]
ABHISHEK PATHAK [Reg No: RA1511008010274]
ABHI RAJ [Reg No: RA1511008010290]
PAWAN GAJRAJ [Reg No: RA1511008010670]

Under the guidance of

Mr. J. PRABAKARAN

(Asst.Professor, Department of Information Technology)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

MAY 2019

SRM INSTITUTE OF SCIENCE TECHNOLOGY

(Deemed to be University u/s 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled " **REAL TIME FACE RECOGNITION AND TRACKING SYSTEM USING DEEP LEARNING** " is the bonafide work of " **SHASHANK S. CHANDEL [Reg No: RA1511008010182], ABHISHEK PATHAK [Reg No: RA1511008010274], ABHI RAJ [Reg No: RA1511008010290], PAWAN GAJRAJ [Reg No: RA1511008010670],** ", who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Mr. J. PRABAKARAN **GUIDE** Asst.Professor Dept. of Information Technology	Dr. G. VADIVU **HEAD OF THE DEPARTMENT** Dept. of Information Technology
Signature of the Internal Examiner	Signature of the External Examiner

ABSTRACT

Nowadays the biometric system is one of the popular topics for the research community. Biometric technology provides us with various facilities like identification, authentication and tracking of individuals. Biometric system can store physical properties of people electronically and they are really helpful because people do not have to remember details like passwords for authentication or identification purposes instead their physical properties captured by the biometric system can be used. Biometric system relies on identification through physical properties which is unique and cannot be altered. There are various biometric techniques such as fingerprint, iris, retina and voice recognition. Biometric systems are used for security purpose because of their real-time applications. Face Recognition systems are one of the popular and significant subjects in biometric systems. Face Recognition based biometric system are being researched, tested and applied in security systems of many countries. Therefore we propose a web application to perform real-time modern facial recognition and tracking tasks. This application provides users with services like creating a personal account with the option of uploading images and creating their own dataset, capturing the digital image of people in real time through a well-framed camera, generating facial features and comparing against trained facial features of uploaded images present in the dataset of the user. This application also provides the option of training a classifier on their dataset to make a better prediction of faces detected in real time and using this application user can generate a detailed report in PDF format containing details like list of individuals identified along with probability estimates and face distance measurements. This application provides a simple GUI for the users to quickly and easily set up their own modern facial recognition based security system.This application can also be used to set up facial recognition based security system in public places like shopping malls,stores etc.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my guide, Mr.J.PRABAKARAN his valuable guidance, consistent encouragement, personal caring, timely help and providing me with an excellent atmosphere for doing research. All through the work, in spite of his busy schedule, he has extended cheerful and cordial support to me for completing this research work.

Author

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

MTCNN	Multi Cascaded Convolutional Neural Network
GUI	Graphic User Interface
ANN	Artificial Neural Network
ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Network
BP	Backpropagation Algorithm
MLP	Multi Perceptron
RELU	Rectified Linear Unit
KNN	K Nearest Neighbour
LMNN	Large Margin Nearest Neighbour
LFW	Labeled Faces in the Wild
API	Application Programming Interface
SQL	Sequential Query Language
HOG	Histogram of Oriented Gradients
SVM	Support Vector Machine
LBPH	Local Binary Patterns Histograms
GPU	Graphic Processing Unit
PCA	Principal Component Analysis

LDA	Linear Discriminant Analysis
SGD	Stochastic Gradient Descent
MVT	Model View Template
ResNet	Residual Neural Network

LIST OF SYMBOLS

CHAPTER 1

INTRODUCTION

In today's digital world security is very important and it is essential for everybody. Nowadays people have to remember many passwords and credit/debit cards pin etc. They have to carry more cards with themselves. Such methods are becoming less practical and also less secure. These problems are acting as a catalyst for increasing interests in the adoption of approaches involving biometric-based security systems. Biometric systems involve identification through biological characteristics of human which is unique for any individual and cannot be altered. There are various biometric techniques like fingerprint, voice, iris, retina and face recognition. These techniques are being continuously researched, tested and also developed and applied in many countries. Face recognition based biometric system is one of the popular and significant area in biometric systems. Nowadays due to extensive research in deep learning and discoveries and obtaining of high accuracy rate, face recognition has been extensively used in security systems for identification and authentication purposes. Face recognition process involves various sub processes like face detection through a well-framed camera or from a static image, extraction of facial features and then comparing against facial features present in the database to obtain an accurate match. In this project, we propose a web application that provides a simple Graphic User Interface (GUI) to set up their own modern facial recognition based security system in no time.

1.1 Motivation and Problem statement

Nowadays facial recognition systems are very popular because they have many practical applications in real life. Nowadays security is a very important aspect for any individual in this digital world. People have to remember many passwords and also have to carry many credit/debit cards. These methods are becoming less practical and also less secure. Biometric security systems based on facial recognition plays a vital role in solving

this problem where people are identified and authenticated based on their unique facial features. A popular example of face recognition technology is the tagging feature in Facebook which uses face recognition technology for tagging friends. It runs facial recognition on the uploaded image and automatically tags friends because Facebook has access to large dataset consists of millions of faces. Because of the recent discoveries and achievements in the deep learning field has made this field quite interesting and opened doors to many new and exciting opportunities. Due to the availability of various machine learning algorithms and face recognition based open source libraries, the task of performing modern facial recognition and running on the mid range personal machine is possible. Recent works also depict high accuracy results from pre-trained models and Transfer learning based approach. All these factors have motivated us to develop an application for performing facial recognition and tracking tasks.
For performing facial recognition on the digital image or the images taken from a camera we have to perform many sub-processes which includes face localization, face alignment.face representation and face comparison. For facial recognition first, we require some known faces in our database so that our system can extract facial features from it and based on that predict matched faces in real time.Convolutional Neural Network (CNN) based approaches achieve the best result in terms of accuracy for facial recognition but other methods also exist which have achieved good results. There are different CNN based architectures and classification methods are present for performing facial feature extraction and face comparison.

1.2 Overview of Facial Recognition Systems

The facial Recognition system has the capabilities like identification or authentication of individuals from a digital image or from a video or live images captured from a camera. There are various techniques used in facial recognition systems but all systems follow the same process which includes face detection, facial features extraction and comparison of selected facial features against faces present in our database. Facial Recognition system can also be described as the biometric-based security system capable of identifying physical properties of any individual which is unique for any human and cannot be altered. Although a biometric system based on fingerprint or iris recognition have high accuracy compared to facial recognition. It is widely used because the facial recognition process does not require any cooperation from individual i.e there is no contact between the subject and the system.Deep learning based facial recognition system have achieved high performance and accuracy.

1.3 Deep Learning

Deep Learning is a machine learning field which operates with neural networks that have more than two layers. These neural networks are known as deep networks. Deep Learning (DL) techniques have been applied in multiple fields like natural language processing, object localization and recognition and also in face recognition. Deep learning provides the most promising methods for face recognition. They use a type of neural networks for the extraction of facial features that can be used for accurate classification and for identities detection.

1.4 Neural Networks

Neural Networks are also known as Artificial Neural Network (ANN) as shown in Figure.1.1. A neural network is heavily inspired by the human being's nervous system like the human brain. These neural networks are known as a deep neural network when we introduce hidden layers. Neural networks in general constructed by stacking neuron layers. A neural network is divided into three types of layers, first is an input layer. the input layer is the entry point in the neural network where training data is fed.second are hidden layers which are intermediate layers. These layers help the neural network learn complicated relationships present in the training data.last is the output layer which gives the final result. For instance, in the case of classification problem if there are 4 classes present in the training data then the output layer will have 4 neurons for each class. Feedforward neural network has fully connected neurons in the adjacent layers. These layers are known as fully connected layers.
Neural network looks like a graph where neurons act like vertices and connections between the neurons act like weighted edges. These weights are then applied to the input

data which gets manipulated across the multiple layers. It is done through matrix multiplication operations. In addition to that, each neuron also applies an activation function which makes the neural network very flexible and it also gets the capability to estimate non-linear complex relationship present in the data. There are many activation functions for different applications like Sigmoid, Rectified Linear Unit (RELU) etc. A Bias unit is also added to the input of the activation function which is a constant value. This whole process of application of weights followed by the addition of bias value is known as feedforward pass. During Neural network training there is first forward pass followed by the backpropagation algorithm [33].Backpropagation Algorithm (BP) is responsible for minimizing the error at the output by propagating it back into the neural network. It works by updating the weights and resultant output from each neuron improves.This whole process of feedforward pass and BP accounts for one full training cycle known as an epoch.

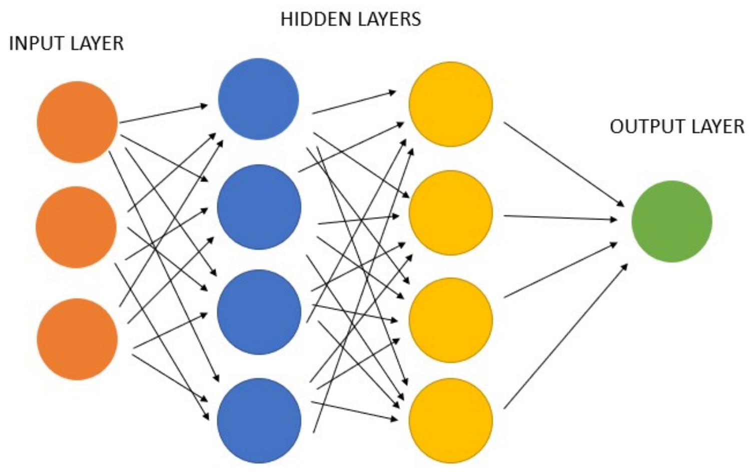


Figure 1.1: ARTIFICIAL NEURAL NETWORK

1.5 Convolutional Neural Networks

The convolutional neural network is a type of deep feedforward neural network.CNN consists of an input layer, multiple hidden layers and the final output layer. The hidden layers consist of a convolutional layer, pooling layer and fully connected layers as shown in Figure.1.2. The convolutional layer is responsible for generating 2D/3D activation maps. It contains a set of filters that extract specific features from the image without losing the spatial arrangement of pixels in the image.CNN has successfully been applied in computer vision [37] and image processing. When multiple filters are used the generated activation maps are stacked together to generate the final feature map. Here the number of channels of output is same as the number of filters used in CNN and the typical spatial dimension of a filter is 3 * 3 * n or 5 * 5 * n (where n is equal to the number of input channel). These filters run across the image so that all pixel values are visited once. These filters must follow a certain step size while moving across the image that step size is known as stride. filters can navigate horizontally and vertically and should be within the image. As the size of stride value increases the size of the image also reduces. For controlling the spatial size of the image, padding is applied which involves adding zeros around the image matrix to retain the size of the input image. In CNN there is also pooling layer which is an optional layer and mainly used to reduce the number of trainable parameters and to reduce the spatial size of the image. Pooling operation is applied after successive convolutional layer.Dropout [40] is also applied to prevent overfitting in neural networks.Dropout refers to the process of avoiding the output of random neurons so that the network does not give output with high variation in case it does not receives certain input features.

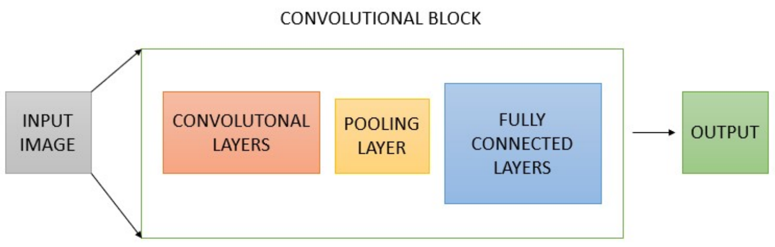


Figure 1.2: CONVOLUTIONAL NEURAL NETWORK

1.6 Deep Metric Learning

Deep Metric Learning is a technique which is mainly used in the face recognition process. In general training of model involves giving the network an input image and the model has to output label for that image. But deep metric learning [8] based approach follows a different method for training the model. In this network is provided with three images where two of the image belong to the same person and the third image is of the random person. In deep metric learning [8], the model gives the output in the form of feature vector instead of giving label for that image. During training, our model manipulates the facial measurements such that the distance between the measurements generated for the same person is lesser and the distance between the measurements of different person is far. Basically what it does is making sure that the same labels should have small distance as compared to objects with different labels.Large Margin Nearest Neighbour (LMNN) metric is one of the popular metric learning technique.

1.7 Transfer Learning

Transfer learning based approach involves first training the neural network from scratch using large dataset. During training, the neural network gains useful knowledge and all this learning is compiled in the form of weights. Now, these precomputed weights now can be transferred to another neural network so it saves time in training a network from scratch. A trained neural network can also be used as a feature extractor by stripping the last layer i.e output layer of the network. Now, this model can be used to generate face embedding without training a model from scratch on large data sets which takes a lot of time, computational power and also availability of large uniform dataset which is not feasible. Transfer learning based approach is a very useful approach for instance if you want to build deep learning based intelligent car we can directly take Inception model which is a pre-trained model by Google for this purpose without training model from scratch for this purpose and wasting a lot of time and computational resources. We can adopt different methods to fine tune the model for our own application like using a pre-trained model as a feature extractor or using the architecture of the pretrained model or training a few layers and freezing other layers.

1.8 Image Processing

Image Processing is an important part of the facial recognition process. Image Processing deals with varieties of problems like object detection, pattern recognition and feature extraction. During facial recognition, we cannot directly compare raw pixels instead, facial features needed to be extracted from the image for face comparison purpose. There are mainly two types of features. Low-level features are features extracted in the form of colour histograms, gradients etc. High-level features are those features which our human eyes sense for identification and detection purpose.

1.9 Face Recognition Process

For performing facial recognition on the digital image or the images taken from a camera we have to perform many sub-processes which includes face localization, face alignment.face representation and face comparison.CNN based approaches achieve the best result in terms of accuracy for facial recognition but other methods also exist which have achieved good results.

1.9.1 Face Localization

Face localization process involves detecting the size and location of faces in the image. The output of this face localization process is the face coordinates where the face is detected in the input image. Many approaches exist for face detection like Face detector by Viola Jones, Histogram of Oriented Gradients (HOG) [26] and CNN based approaches like Multi Cascaded Convolutional Neural Network (MTCNN) [19]. Deep learning based approaches have achieved extraordinary results. They use CNN for face localization in real time.

1.9.2 Face Alignment

Face alignment is the method which involves processes like the centring of faces in the image so that they lie in the centre as much as possible and also scaling of faces present so that all faces present in the dataset are of the same size and making sure that all required facial landmarks on the face can be projected. There are various methods proposed for face alignment purposes like alignment using an ensemble of regressive boost trees [20], 2D/3D face alignment using deep learning and MTCNN [19].MTCNN

[19] uses a cascade of CNNs for face alignment and face landmark estimation where regressive boost trees method works on the pixel intensities and detects high-quality facial landmarks in real time.

1.9.3 Face Representation

Face representation refers to the process of capturing important facial features in the form of a feature vector so that different faces can be distinguished.the computer must represent pixels of the image in the form of a feature vector. Nowadays Deep learning based approaches are state of the art methods. These approaches generate face embedding. Here deep neural network is first trained on a very large dataset so that it can learn to generate accurate face embedding or feature map. Now after training is completed, the last layer i.e. output layer of the network is removed in order to obtain face embedding for any input face. The weights which are generated by training the model helps in the generation of face embedding. There have been various methods proposed in this field like DeepFace [39] by Facebook, Facenet [34, 10] developed by Google etc. All these proposed face recognition models are focussing on improving the accuracy of the model but work is still needed in the areas like improving the efficiency of the model in terms of speed and size. However, some of the recently proposed models have shown extraordinary results in improving the efficiency of the network.

1.9.4 Face Comparison

Face comparison refers to the process of comparing the facial features in order to obtain the correct match. Face comparison is the last step in the face recognition pipeline. Face comparison is of mainly two types, distance based and classification based. In the face comparison process, the face encodings which are generated for each face in the face representation process is used. In the distance-based method, the comparison is done by calculating the Euclidean distance between the target face and all the faces present in our dataset. A threshold value or tolerance value is learned by the system by utilizing the nearest neighbour search [21] method. If the distance is less than tolerance threshold value which means faces match. Otherwise, faces do not match. Classification based method involves training a classifier, For instance, Linear SVM classifier other classifiers also exist, which takes in the face encodings for training and gives probability estimate for the test image and also predict the closest match. Training a classifier requires more than one image per class and every time a new image is added we need to retrain the classifier model. So the classifier-based approach is time-consuming.

1.10 Research Objectives

This project aims to develop a real-time facial recognition and tracking application by detecting human faces using deep neural network and performing face comparison with faces present in our database.This research also studies different algorithms and approaches present for facial recognition process and also learning their effectiveness. Our main goal is to develop an application for facial recognition which can run on a mid-range personal machine without the requirement of GPU but also give high performance in terms of accuracy and speed in the real-time scenario. For this application, we are using a pre-trained deep neural network to generate facial features and applying distance based or classification based methods for face comparison. For our application, we are adopting the Transfer learning based approach which involves using a pre-trained deep neural network model because the state of the art approaches takes a lot of data and high computational power. Training a deep neural network from scratch is very time consuming, it can take days which is not feasible. Recent work in face recognition using transfer learning and open source machine learning libraries have achieved very good results in terms of accuracy. This application provides a simple GUI to perform all modern real-time facial recognition and tracking tasks and using this application, user can also set up their own facial recognition based security system in no time.

1.11 Organization of the Thesis

In this project, we are proposing a facial recognition and tracking application. The organization of this project report is as follows: Chapter one gives the overall theory behind the face recognition technology. It gives a brief description of all the underlying processes involved in the face recognition process. this chapter also gives detail of deep learning, neural networks, CNNs, deep metric learning [8], image processing and at the end of this chapter research objectives were proposed. Chapter two focuses only on the literature survey on face recognition technology. It gives detail of all the work that has been done by other researchers in face recognition, deep learning and computer vision field. Chapter three mainly analyses the system proposed for facial recognition. It highlights all the existing system and its issues. This chapter also gives detail of the proposed system and its advantages and also mentions the hardware and software requirements.

Chapter four focuses on the system design and gives details like algorithms and methods used for face detection, alignment, representation and comparison. It gives details of various pre-trained models present and description of the model used for our application.This chapter gives detail of various standard datasets used for training models. After that, this chapter provides complete detail of procedure and implementation, tools and libraries used, data flow, system architecture diagram, the framework used, different modules and their functionalities, database design and testing details of our proposed application.

Chapter five provides results obtained while performing the facial recognition process in terms of accuracy.chapter six and seven provides a conclusion and future enhancement respectively. chapter eight provides publication status.chapter nine provides detail of all the references taken. appendix provides the code snippets of our application.

CHAPTER 2

LITERATURE SURVEY

Computer vision is the scientific field which deals with how computer systems can be made to understand digital images and digital videos and extract useful information from it. Evolution in this field is very impressive. Deep learning based approaches have shown extraordinary results in terms of performance. It has undergone tremendous changes over the years. Image processing techniques are also evolving and improved techniques are coming into the picture. Deep learning is one of the hottest topics for researchers nowadays and it also has a very active community. Because of the advancement in the computer hardware and easy availability of information this field is evolving at a faster rate. It is even surpassing human performance. Face recognition is a process of determining whether two images belong to the same person or not. Performance of human being in face detection and identification is around 98%. In 2015, Facenet [34, 10], a model by Google has performed better than the human being in the face recognition task. Recently a model proposed by the Chinese researchers has even performed better than Facenet [34, 10] but unfortunately, that work is not disclosed in the public. However several open source models exist which have shown extraordinary results.

For performing facial recognition on the digital image or the images taken from a camera we have to perform many sub-processes which includes face localization, face alignment.face representation and face comparison.CNN based approaches achieve the best result in terms of accuracy for facial recognition but other methods also exist which have achieved good results.CNNs are useful for extracting specific and unique features from the image which simple neural network cannot extract easily.CNN based models are built by stacking multiple CNN blocks containing convolutional-pooling layers which are followed by the fully connected layer [14]. However, work is being done to build a single system [42] that can perform all the sub-processes that happen during the facial recognition process. Now it is clear that CNN based system gives the best performance in the face recognition task. Still, research is going on and new CNN based

architectures are being proposed by the researchers.
CNN architecture has evolved a lot starting from constructing very deep and complex models like VGG Net, ResNet [28] and Inception [5] to developing models which are simple and can also run on smaller or less powerful devices like mobile devices. Nowadays researchers are working on developing models which take less computational resources and have small size but can give performance similar to deeper models. Many such models have been proposed by the researchers. For instance, XceptionNet [4], MobileNet [2, 35] and SqueezeNet [11].In 2015, when Google proposed Inception architecture [5] then researchers started working on architecture in which connections between layers are different and layers are not simply stacked as shown in Figure.2.1. This approach of connecting layers also gave very good results. But all such models consume a lot of memory and computational power. New models have been developed and proposed by researchers that can run on mobile devices [36].
Face localization process involves detecting the size and location of faces in the image. First popular face detector was Viola-Jones detector [29] and then many other popular face detectors have been proposed like HOG [26] and SVM based detector. Recently CNN based face detectors have also been proposed. For instance, MTCNN [19].First, popular face detector was Viola-Jones detector [29] which has given very good results in real time scenarios and then many other popular face detectors have been proposed like HOG [26] and SVM based detector. In this method, the first image is represented by HOG [26] and then SVM performs the binary classification which indicates whether a detected face is of a person is or not. Viola-Jones method used Adaboost classifier and HaaR feature representation. Recently CNN based face detectors have also been proposed. For instance, R-CNN [31, 13] and MTCNN [19].MTCNN [19] model uses three CNNs which are P-net, R-net and O-net.MTCNN [19] can perform both face detection and face alignment. During training, MTCNN [19] model uses an image pyramid. It is trained on CelebA and Wider face data.Among these available methods MTCNN [19] gives best results in face detection and face alignment. However, MTCNN [19] approach requires GPU to run because it uses CNN whereas HOG [26] based method does not require GPU and can easily run on CPU.
Next process in the face recognition pipeline is face alignment. Face alignment is the method which involves processes like the rotation of faces in the image so that they

lie in the centre as much as possible and also scaling of faces present so that all faces present in the dataset are of the same size and making sure that all required facial landmarks on the face can be projected. Face alignment is the essential step in projecting facial landmarks which in turn used in the network for the generation of facial measurements. Various useful and effective methods have been proposed by many researchers in this area. Some of the notable and popular methods include face alignment based on an ensemble of regressive boost trees[20], this approach has the capability to estimate 194 facial landmarks. This method uses Gradient boosting and decision tree both are machine learning techniques.MTCNN [19] based approach which utilizes a cascade of CNN for face alignment and detection purpose.MTCNN [19] utilizes three CNNs P-net, R-net and O-net for face detection and facial landmark estimation.MTCNN [19] estimate five landmark points which include eyes centre, the tip of the nose and both mouth corners. Here alignment is done through affine transformation technique. Another method is 2d/3d deep neural network based face alignment approach [3].
After face detection and alignment process comes the face representation process in the face recognition pipeline. Face representation refers to the process of capturing important facial features in the form of a feature vector so that different faces can be distinguished.the computer must represent pixels of the image in the form of a feature vector. Nowadays Deep learning based approaches are state of the art methods. These approaches generate face embedding. Here weights of the pre-trained model are used for the generation of face encodings. Many models have been proposed by the researcher in this area. For instance, In 2015, Facenet [34, 10] proposed by Google research team has performed better than a human because the performance of human in face recognition is around 98% but Facenet [34, 10] achieved results greater than 98%. Recently in 2018, a model proposed by the Tencent You Lab [41] (a group of Chinese researchers) has shown results better than Facenet [34, 10] but unfortunately, model architecture details have not been disclosed in public. However several model's architecture, training details and pre-trained models are available as open source which is state-of-the-art and gives very good results for face recognition. Some popular works in this field are DeepFace [39] by Facebook, Facenet [34, 10] by Google based on Triplet loss and LMNN, ArcFace [17] by DeepInsight etc.
Face comparison refers to the process of comparing the facial features in order to ob-

tain the correct match. Face comparison is the last step in the face recognition pipeline. Face comparison is of mainly two types, distance based and classification based. In the distance-based method Euclidean distance is calculated between face measurements of a different person. However euclidean distance doesn't give the correct or linear scale for comparison but it does give the meaning that distance between the face encodings of the same person is lesser in comparison of facial measurements of a different person. Facenet [34, 10] model by Google uses squared L2 distance. Both Euclidean and non-euclidean distance measures exist. Some non-euclidean distances like correlation, histogram intersection etc. Research has also shown that the direction of the feature vector gives more sense than the magnitude.Here model is trained using deep metric learning [8] technique, which manipulates the facial measurements so the distance between the measurements generated for the same person is lesser and the distance between the measurements of different person is far. Basically what it does is making sure that the same labels should have small distance as compared to objects with different labels.LMNN metric is one of the popular metric learning technique.Facenet [34, 10] by Google is also trained using this technique.

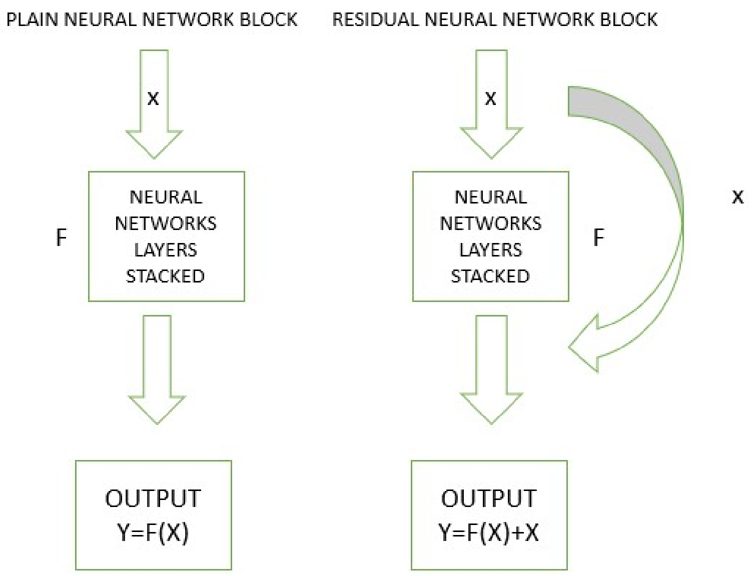

PLAIN NEURAL NETWORK BLOCK
RESIDUAL NEURAL NETWORK BLOCK
X
X
X
F
NEURAL NETWORKS LAYERS STACKED
NEURAL NETWORKS LAYERS STACKED
F
OUTPUT Y=F(X)
OUTPUT Y=F(X)+X

Figure 2.1: RESIDUAL NEURAL NETWORK(ResNet)

CHAPTER 3

SYSTEM ANALYSIS

3.1 Existing System

There are different methods exists for performing facial recognition. Some of the existing methods are :
Principal Component Analysis (PCA) based on EigenFaces.This is an unsupervised based Machine Learning (ML) technique that works by removing unnecessary data and using necessary data to detect facial patterns accurately. It is basically involved in the reduction of the dimensionality of the data. Here Training data images act as Eigenfaces which are extracted from the large datasets using PCA technique and checked against the target image for face verification.
Linear Discriminant Analysis (LDA) is a supervised ML learning technique which performs pattern recognition on the facial features. Other ML techniques like Viola-Jones [29] Haar cascade classifier also used for facial recognition task. OpenCV [16] is the python [38] module which is used in computer vision related works like facial recognition. There are several methods and algorithms provided in the OpenCV [16] for performing facial recognition like Local Binary Patterns Histograms (LBPH), Eigen and Fisher face.Face recognition can be performed both on digital image and also on video captured by the camera in real time.
Earlier face recognition was done only on still images. Different methods exist for face recognition but it is mainly categorised into three types: feature based, holistic and hybrid-based. The holistic-based approach works on global feature i.e. it takes into consideration the whole face region as input data for face recognition. Eigenfaces,Fisher faces, SVM,NFL method are examples of a holistic-based approach. They all follow the PCA technique for the dimensionality reduction but also retaining the useful data. Next approach is Feature-based approach which focuses only on the specific facial features and it does not take the whole face region for the input. It is also known as the

local feature based approach. Here facial features such as nose and eyes are separated and that acts as input data for the classifier. Examples of feature-based approach are pure geometry based, Hidden Markov model-based methods. Now the last approach is the hybrid approach which takes both global and local feature into account for face recognition. It works on the idea of the human visual system. It also takes into account which feature is necessary and which is not. The best performing approaches for facial recognition are neural network based approaches. The neural network requires training on the example image to perform facial recognition. They use certain ML techniques to extract useful facial features from the image. However many neural network based approaches exist but Multi Perceptron (MLP) with backpropagation algorithm has been the most used technique.BP is essential part while training neural network because it is responsible for minimizing the error which is generated.the final output is made from the output of all previous layers in the network so error gets propagated in the network and in order to reduce that error the loss or error in the output is sent back into the system so that weights of network get updated.BP supports many optimization algorithms [33] like Stochastic Gradient Descent (SGD), RMS-prop, Adam [7] etc.

3.2 Issues in Existing Methodology

For performing facial recognition on the digital image or the images taken from a camera we have to perform many sub-processes which includes face localization, face alignment.face representation and face comparison.CNN based approaches achieve the best result in terms of accuracy for facial recognition but other methods also exist which have achieved good results. Image Processing is an important part of the facial recognition process. Image Processing deals with varieties of problems like object detection, pattern recognition and feature extraction. During facial recognition, we cannot directly compare raw pixels instead, facial features needed to be extracted from the image for face comparison purpose. There are mainly two types of features. Low-level features are features extracted in the form of colour histograms, gradients etc. High-level features are those features which our human eyes sense for identification and detection purpose.MTCNN [19] based approach works on high-level feature and it requires Graphic Processing Unit (GPU) to run whereas HOG [26] based method works on the low-level features and it doesn't require GPU and can easily run on CPU.MTCNN [19] based method gives the best result but HOG [26] based approach also gives the comparative good results considering it does not require GPU. Here low-level features are divided into two types first is general and other is domain specific. Visual or high-level features include shape, texture , colour etc. So all those feature-based and holistic methods does not give good results and accuracy if the input image is not up to the standard like these methods perform worse if faces in the images are not properly aligned, if lighting condition is not good and if it has bad posture so all these factors heavily affect the face recognition task and these approaches perform badly in such condition or give wrong results or predictions. The pose and the illumination problem are the two most important problems which should be taken into consideration for performing facial recognition task with high accuracy rate.Due to these problems, the same face appears different due to a change in lighting conditions. So such methods must be adopted which overcome such problems and they are independent of the lighting conditions.For solving pose, different methods are adopted like aligning faces, estimating facial landmarks, bounding boxes etc. Face alignment is done so that all the facial landmarks remains at the same position irrespective of rotation of the face and can be easily projected. For solving the illumination problem, MTCNN [19] is one of the popular methods which uses a cascade of CNN for face alignment and detection and it takes image pyramid as input and outputs the bounding box coordinates surrounding the face along with five landmarks. Problems also exist in the face comparison process because we cannot directly compare the raw pixels. For face comparison task we need feature vectors for each face which can be used to compare faces. For generating the accurate facial measurements or encodings for any face we need a pretrained model which can act as a feature extractor. We can solve this problem by adopting a transfer learning based approach. For measurement purpose, both Euclidean and non-euclidean method exist.

3.3 Proposed System

In this project, we are proposing a facial recognition and tracking application using deep learning. For performing facial recognition on the digital image or the images taken from a camera we have to perform many sub-processes which includes face localization, face alignment.face representation and face comparison.CNN based approaches achieve the best result in terms of accuracy for facial recognition but other methods also exist which have achieved good results. For face detection, we are using HOG [26] based approach which does not require GPU and also gives very good result.HOG [26] based method involves first making the image black and white and then it looks at every single pixel and also the surrounding pixel in that image.it determines how dark is current pixel as compared to all those surrounding pixels and it marks arrow in that direction where a pixel is getting darker. So in HOG [26] based method lightness and darkness in the image of the same person does not change the result and do not give the wrong result because it considers the direction where pixel brightness changes. So both dark and light images will give the same representation. For face alignment purpose we are using Dlib [23] aligner and for facial landmark estimation, we are using Dlib [23] 68 point landmark estimator which is based on an ensemble of regressive boost trees approach which can estimate 194 facial landmarks[20, 32].

In this application, we are also giving the option of using MTCNN [19] based approach for face detection and alignment. The only condition in using this method is that it requires an appropriate GPU to be installed in the user's computer machine.MTCNN [19] performs better in real time for face detection and face alignment tasks. For computing facial measurements we are using ResNet [28] architecture based model which is already trained and it contains 29 convolutional layers. For performing all face recognition tasks we are using open source Face_Recognition API which is a python [38] implementation of Dlib [23].we are using Dlib [23] ResNet [28] model because it takes less memory and it has quick load time as compared to very deep and complex models like Inception [5] model by Google and Inception-ResNet [28] model used in Openface [1] library for facial recognition written in Lua script which has achieved the highest performance in the face recognition task. However Dlib [23] based model does not surpass the results of the opeface library but it gives very close and competitive results without using a lot of computational time and resources.

Dlib [23] ResNet [28] model generates 128 face encoding for each face and then for face comparison approach it computes the Euclidean distance and also set some tolerance threshold value which depicts that if face encoding of two faces is less than the threshold value then it depicts that face belongs to the same person. Our application also provides the option of training a Linear SVM classifier on top of embeddings generated for each face so that a trained classifier model can predict known faces in real time and can also give probability estimates. However, training a classifier is a time-consuming task and it requires more than one image per class to give good results and if you add a new image in the database you need to retrain the classifier for it to predict.whereas face distance-based method can work on single image per class this approach is known as one-shot learning.Here we are using Transfer learning based approach by using a pretrained model weights for generating facial measuremnents.

3.4 Advantages of Proposed System

For facial recognition process, we have to perform many sub-processes like face detection, face alignment, face landmark estimation, face description generation and face comparison.There are several advantages of our proposed approach. First, in face detection we are using HOG [26] method which looks for the direction of pixel brightness which ensures that both dark and light images of the same person give correct results.we are also giving the option of using MTCNN [19] based method for face alignment and face detection which utilizes CNN and requires GPU to run. So Users who have GPU installed in their computer machine can utilize MTCNN [19] based method.MTCNN [19] performs better than HOG [26] method in real time and also in case of images having pose and illumination problem. For generating facial landmarks we are using Dlib [23] based 68 points facial landmark estimator which is based on already proposed 194 point face landmark estimator based on regressive boost trees [20]. Another advantage of our proposed application is that we are adopting the best state of the art techniques for solving problems which arise in the face recognition process but also keeping in mind that application can be accessed by all users with or without GPU with minor differences in the performance.

We have also adopted the transfer learning approach by using pre-trained model weights for generating facial measurements. This reduces a lot of time which is required in training a deep neural network from scratch on a large dataset. By this approach, we do not require large dataset for training purpose and we can directly work on our own dataset.

Our application also provides the option of training a Linear SVM classifier on top of embeddings generated for each face so that a trained classifier model can predict known faces in real time and can also give probability estimates. However, training a classifier is a time-consuming task and it requires more than one image per class to give good results and if you add a new image in the database you need to retrain the classifier for it to predict.whereas face distance-based method can work on single image per class this approach is known as one-shot learning.

3.5 Requirements

3.5.1 Hardware Requirements

Processor : INTEL I5/I7

Main Memory : 8/16 GB RAM

Processing Speed : 2.47 GHz

Hard Disk Drive : 1 TB

GPU : 4 GB(not mandatory)

Keyboard : 104 Keys

Webcam : HD

3.5.2 Software Requirements

Operating System : Win 10/Mac OS/Linux

Coding language : python [38] 3.6, HTML5, CSS, JS

Framework : Django 2.1.7

IDE : Visual studio code

Database : Sqlite

CHAPTER 4

SYSTEM DESIGN

4.1 Algorithms and Techniques used

For performing facial recognition on the digital image or the images taken from a camera we have to perform many sub-processes which includes face localization, face alignment.face representation and face comparison.CNN based approaches achieve the best result in terms of accuracy for facial recognition but other methods also exist which have achieved good results. For solving each subproblem that arises in the face recognition process we are utilizing the effective algorithms and techniques available to achieve high performance in terms of accuracy, speed and size. We are also making sure that our application can be used for facial recognition by all kinds of users i.e user with a mid-range personal machine without GPU can also benefit from this application. For solving this problem we are utilizing both kinds of algorithms and techniques that can run on a system with or without GPU and if a user has a high-end system then the user can also achieve the highest performance in performing face recognition tasks. Some of the techniques and algorithms used in our facial recognition and tracking application are:

For solving the problem of **face detection** we have used two techniques in our application first is HOG [26] technique and second is MTCNN [19] technique.

4.1.1 HOG (Histogram of Oriented Gradients)

This technique was invented in the year 2005. The technique analyses the low-level features in the image for face detection. This technique does not require high-end GPU enabled system to run. The process involves first converting our image into black and white i.e we remove the colour data from the image. Then we take into consideration every pixel in the image and also its surrounding pixel i.e pixels at top, bottom, left and right of that pixel. Now, this technique checks in which direction brightness is increasing in the image and marks that direction. These steps are repeated for every pixel in the image.The direction which is marked by this technique is called as gradients which depict in which direction flow of light changing and every pixel in the image is replaced by this gradients. This technique of taking into consideration the direction of brightness change is very helpful because both dark and light images of the same person will appear the same and not different. That is why HOG [26] is an efficient technique for face detection purpose. However, using gradients for each pixel gives us too much detail and makes face detection less efficient. To solve this problem, this technique breaks the image into square blocks containing a set of pixels and identifies the strongest gradient in that square and store only that detail and rest of the extra data are removed. This makes the whole process of face detection more efficient and now we can easily compare the face pattern generated from this approach using several images to detect faces in the target image. There are many libraries available but in our application, we have used Dlib [22] library for implementing HOG [26].

4.1.2 MTCNN (Multi Cascaded Convolutional Neural Network)

Another technique used for face detection is MTCNN [19]. This technique requires the system to be GPU enabled to run.MTCNN [19] utilizes a cascade of CNNs for face detection purpose. It gives better performance than the HOG [26] based face detector.MTCNN [19] takes into consideration high-level features present in the image for face detection. It consists mainly of three CNN layers:P-net,R-net and O-net. It gives better accuracy in face detection task in real time. It also solves the problem of pose and illumination which arises mostly in images captured in real time by the camera. It can extract useful and specific information from the image because it uses CNN for face detection. This technique has better performance in face detection accuracy.MTCNN [19] process cannot run efficiently on CPU it requires GPU to run in a fast and efficient manner. This technique takes image pyramid as input data for face detection purpose and uses PRelu activation function for every CNN layer.IImage pyramid refers to a set of images with different sizes. Using MTCNN [19] we can perform both face detection and alignment tasks. It output the bounding boxes where a face is detected in the image and uses NMS approach to reduce the number of redundant bounding boxes. This technique also returns the five facial landmark points for face alignment purpose. So we do not need separate face aligner for performing facial alignment in our system.MTCNN [19] model is trained on Wider Face and Celebi dataset and pre-trained model is available for public use.

4.1.3 Face Landmark Estimator(Dlib_68 face aligner)

For solving face alignment problem we are using two main techniques first is MTCNN [19] which is CNN based approach and outputs five landmark points which we discussed already. Another technique used for face alignment purpose is Dlib [23] face aligner. Face alignment is an essential step in the facial recognition pipeline because it solves the problem of aligning faces present in the image so that our network can easily recognize faces in the image. Because our network cannot recognize images of the same person with different pose and face rotation. It will see both images as a completely different individual. Dlib [23] based Face aligner does not require GPU enabled system to run, unlike MTCNN [19] technique. For face alignment in the image Dlib [23] face aligner estimates total 68 facial landmark points like eye centre, the tip of the nose, top of the chin etc. as shown in Figure.4.1. A pre-trained model for face alignment and landmark estimation is provided in Dlib [23] library.Dlib [23]_68 face aligner uses regressive boost tree technique [20] for facial landmark estimations.Dlib [23] face aligner only performs the face landmark estimation, unlike MTCNN [19] technique which performs both face detection and face alignment.MTCNN [19] outputs the 5 landmark points whereas Dlib [23] face aligner output 68 landmark points.

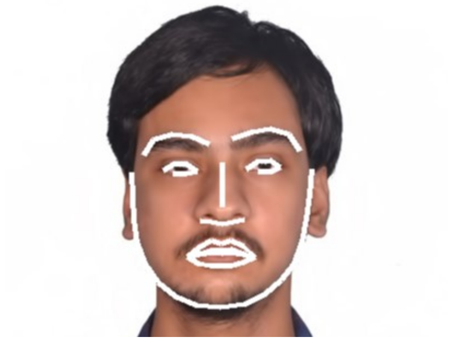


Figure 4.1: FACIAL FEATURES

4.1.4 Transfer Learning Approach

For solving face description problem in the face recognition pipeline we need a deep neural network which can generate facial measurements that can be finally used for face comparison. For a neural network to generate accurate and appropriate feature maps we have to train the model from scratch using a very large uniform dataset so that our model learns to generate facial measurements. Training a deep neural network from scratch on large dataset takes a lot of time and computer power which is not a feasible option for everyone and availability of such large uniform dataset is also another problem because if the dataset will not be proper then the network will not converge properly and will not give correct facial measurements. To overcome all these problems we are using Transfer learning based approach for our facial recognition application. Training learning based approach involves first training the neural network from scratch using large dataset.

During training, the neural network gains useful knowledge and all this learning is compiled in the form of weights. Now, these precomputed weights now can be transferred to another neural network so it saves time in training a network from scratch. A trained neural network can also be used as a feature extractor by stripping the last layer i.e output layer of the network. Now, this model can be used to generate face embedding without training a model from scratch on large data sets which takes a lot of time.

In our application, we are using a pre-trained model provided by Dlib [23] library which gives 128 facial measurements for the input image. This model is based on the ResNet [28] architecture with 29 convolutional layers. We have also tested another state of the art model Facenet [34, 10] which is based on the Inception-ResNet [28] architecture proposed by Google.this model generates 512 facial measurements. Facenet [34, 10] by Google performs better than Dlib [23] based model. But Dlib [23] based model is faster than facet because facet uses a very deep and complex neural network and also uses MTCNN [19] which uses CNN for face detection and alignment.In our application, we are using Dlib [23] based model for a generation of facial measurement because of its speed and simplicity and it also gives very close and competitive results on the various dataset.For developing our application on Django framework we have used Face_Recognition API [12] for executing all Dlib [23] face recognition modules.

This API provides python [38] implementation of Dlib [23] face recognition modules because Dlib [23] library is originally written in C++ language.

4.1.5 Face Comparison Techniques

After facial measurements are generated using the pre-trained model. Now the final step in the facial recognition pipeline is face comparison process. For face comparison, we cannot directly compare the raw pixels that is why we have generated facial measurement for each face using the pre-trained model. Face comparison refers to the process of comparing the facial features in order to obtain the correct match. Face comparison is the last step in the face recognition pipeline. Face comparison is of mainly two types, distance based and classification based. In the face comparison process, the face encodings which are generated for each face in the face representation process is used. In the distance-based method, the comparison is done by calculating the Euclidean distance between the target face and all the faces present in our dataset. A threshold value or tolerance value is learned by the system by utilizing the nearest neighbour search [21] method. If the distance is less than tolerance threshold value which means faces match. Otherwise, faces do not match.In classification based approach we can use different classification algorithms like K Nearest Neighbour (KNN),Support Vector Machine (SVM) etc.To include classification based face comparison module in our face recognition application we have used scikit learn [9] python [38] library. This library provides easy implementation of many classification and regression-based ML algorithms.In our application, we are first training a classifier model using the images present in our database. Then we are using that trained classifier model for prediction of faces detected in real time and also give probabilty value for faces recognized.

4.1.6 KNN Algorithm

K-Nearest Neighbour algorithm classifies new cases based on the available cases. This algorithm can be used for both classification and regression purpose. Here we are utilizing the KNN algorithm for classification purpose. We are using a voting technique to identify faces in real time. In our application, we are using the counter value which keeps track of faces detected and corresponding label output which is obtained using distance threshold technique and increment its value. Now label value which has the highest vote value is given as output. For storing votes for each label we are using python [38] dictionary.

4.1.7 SVM algorithm

SVM stands for support vector machine which is another classification algorithm used in our application for face comparison purpose. In our application, we have used Linear SVM for classification purpose. First, we have trained our classifier model based on the SVM algorithm and used that pre-trained classifier model for prediction and probability estimation. This whole classification module in our application is implemented using scikit learn [9] and Pickle python [38] library.SVM is a type of supervised ML algorithm used for both classification and regression purpose. This algorithm works by drawing a hyperplane for dividing features into different classes.It plots all features on N-dimensional space and then performs the classification task.

CHAPTER 5

PROCEDURE

The goal of this final year project is to develop a facial recognition and tracking web application that gives high face verification performance. But as it was presented throughout state-of-the-art methods use enormous amounts of data and require immense computational power. It has been shown that such methods take a long time to train, from several days to even months, making it infeasible for one to train a model from scratch in the context of a final year project that must be developed in approximately six months. This way, the proposed work will use Transfer Learning Approach, like pre-trained models and pre-computed weights. In this project we are using latest state of art pre-trained deep net models (Dlib [23] ResNet [28] model by Davis King and face_recognition API [12]) provided by open source community and training our classifier on top of it using transfer learning approach to perform face recognition on our own dataset. In this project, we are developing a web application for performing facial recognition and tracking. This web application is built using Django framework and will provide users with many functionalities like which includes GUI for performing complex facial recognition and tracking tasks creating their own dataset by uploading images, identifying individuals in real time .calculating face distance and probability and getting personalized tracking report.

5.1 Dataset

During the development of the facial recognition and tracking application, we also researched various publically available standard dataset. These datasets are basically used for training model. Dataset is a very important part of facial recognition and ML tasks. Basically, the performance of the model is heavily dependent on the datasets which are used for training purpose apart from the model architecture. In our facial recognition and tracking application, we have used the transfer learning approach by taking a pre-trained model for facial measurements generation. During our research of available open source state of the art pre-trained model, we also learned about various datasets and its details on which those models were trained. Some of the popular standard dataset used for training face recognition based models are LFW [15], CASIA-WebFace, VG-GFace 1 and 2 [30], face scrub etc.LFW [15] stands for labelled faces in the wild .this is the standard dataset for measuring the accuracy of different trained models. This dataset consists of face images basically used for performing non-restricted face recognition tasks. There are totally 13233 face images present in this dataset with 5749 different face identities. For this dataset Viola-Jones [29] based face detector was used for face detection purpose. Another popular dataset is VGGFace [30] which consists of 2.6 million images of celebrities. This dataset was created by researchers so that the open source community have access to large dataset which is generally required for face recognition tasks. CASIA-Web face is another public dataset which consists of around 500k face images. Facenet [34, 10] based model was trained on both VGGface and Casia webface dataset whereas Dlib [23] based model was trained on VGGFace and Face scrub datasets.

5.2 Model

For Performing face recognition we require face measurements or feature vector for face comparison. In order to generate face encoding/face measurements, we need a deep neural model that can generate face encodings for input face image. We cannot directly compare the image pixels for face comparison and recognition process we need facial features for that. In our application, we are mainly using a pre-trained model for facial feature generation purpose. We have used the pretrained model provided by Dlib [23] library. The author of Dlib [23] has already trained this model from scratch on a very large dataset consisting of around three million faces. This model is based on ResNet [28] architecture which is state of the art architecture. Original ResNet [28] has 34 CNN layers but Dlib [23] based model has only 29 CNN layers and filters are manipulated and reduced to half in Dlib [23] model as compared to the original ResNet [28] model.ResNet [28] model follows a different approach in connecting several CNN layers it does not follow the traditional method of stacking layers. This Dlib [23] pre-trained model gives 128-dimensional face measurements for each input face.Here model will generate roughly the same face encodings for faces of the same person. This Dlib [23] based Encoder was trained on the VGGFace [30] and face-scrub dataset. The loss function used in this model basically project the faces on the sphere [24] of radius 0.6 which is non-overlapping in nature. Dlib [23] based model has achieved accuracy of 99.38% on Labeled Faces in the Wild (LFW) [15] dataset. This model was presented in the year 2017 and it gives competitive results as compared to other state of the art models.This model also have quick loading time and it has small model size and it is easy to use. For using pre-trained Dlib [23] model we have used Face_Recogntion Application Programming Interface (API), which abstracts all the complex Dlib [23] face recognition modules and also provides python [38] implementation because Dlib [23] is written in C++ language. During the development phase of our facial recognition and tracking application, we have also used another state of the art pretrained model based on Facenet [34, 10] by Google. This gives better performance than Dlib [23] based model. Open face, which is an open source library based on Facenet [34, 10] paper. Another library with Tensorflow implementation[25] is also available in the open source community. This model is based on Inception-ResNet [28] architecture. which is very deep and complex. This model has been trained using both dataset CASIA-WebFace and VGGFace2 [30] and has achieved an accuracy of 99.05% and 99.65% on LFW [15] dataset benchmark respectively. This model is very deep and heavy and it takes time to load and it performs efficiently if GPU is available.But our goal was to develop an application which can perform efficient and fast facial recognition without any constraint of GPU so that it can run on all kinds of midrange personal machine.

CHAPTER 6

IMPLEMENTATION

This application is built using Django python [38] based web framework and It uses open source API which is built on top of Dlib [23] library by Davis king which uses ResNet [28] model which contains 29 convolutional layers. This model is trained by Davis king and we are using that pre-trained ResNet [28] model and weights to perform face recognition tasks. However face recognition tasks involves performing several subtasks like face detection and alignment, face landmark estimation, computing face measurements and face comparison. All these sub-functionalities are provided by face recognition API [12]. Here for face detection and alignment, we are using algorithms like HOG [26], Dlib [23] pose estimation, OpenCV [16] affine transformation. However, at present best approach for face detection and alignment is based on CNN one such example is MTCNN [19] but it requires GPU to execute so here we are using HOG [26] as default approach which runs on CPU (gives comparatively good results) for our web application. But user can switch from the HOG [26] to MTCNN [19] approach provided they satisfy GPU requirements. By this more people can access this application even if they don't have GPU in their system.For face landmark estimation approach we are using Dlib [23] 68pt landmark estimator and for computing 128 face embeddings we are using pre-trained Dlib [23] ResNet [28] model.

6.1 Tools and Libraries used

For performing face recognition process different technologies and libraries are being used. Our application uses python [38] as the main coding language for writing face recognition scripts. We are using python [38] version 3.6.8. python [38] is high-level language and it is very powerful, fast and portable in nature. python [38] is widely used for machine learning and deep learning based task.it is widely used because of the availability of wide varieties of modules for performing different tasks. python [38] is

actively used in the data mining and ML community. Using python [38] we can perform efficient data manipulation and most of the machine learning algorithm libraries are present in python [38] language. In the past several practical works on face recognition used python [38] for coding and achieved state of the art results. This application also uses NumPy [27] library which is used for dealing with the feature vector and storing image detail in the form of a matrix containing pixel values during feature extraction.NumPy [27] is a library used for scientific computing purposes. It provides an N-dimensional array object.
Open CV is another important library which is basically used for image processing and in the computer vision field. It provides various in-built methods to quickly capture and handle web camera and take picture and record video and also can change framerate to capture and detect faces in the image. Dlib [23]-ML is another library by Davis king which is in C++ for solving complex and real-world ML-based problems. It provides the implementation of many ML and optimization based algorithms.Face_Recognition API [12] is the python [38] implementation of Dlib [23] face recognition module. scikit learn [9] is another useful ML library available in python [38] which provides many machine learning models like various classifier models like KNN, SVM etc.
Tensor flow is a powerful library by Google for performing graph-based heavy computation.it can work both on CPU or GPU. It can be used for performing deep neural based face recognition process.Another popular library is Keras which is just a high-level implementation of Tensorflow because it abstracts many complex features of Tensorflow and makes it quick and easy to create a deep neural network-based model. In this face recognition and tracking web application, we have used Django which is a python [38] based web framework for creating the whole GUI for performing complex modern facial recognition and tracking tasks without any extra effort.

Table 6.1: **Libraries Used.**

NumPy [27]==1.16.1	Django==2.1.7
OpenCV [16]-python [38]==4.0.0.21	reportlab==3.5.13
Pillow==5.4.1	django-crispy-forms==1.7.2
Click==7.0	requests==2.21.0
matplotlib==3.0.3	urllib3==1.24.1
scikit-learn[6]==0.20.3	pip version==19.0.3
scipy==1.2.1	Virtualenv
Dlib [23]==19.8.1	
cmake==3.13.3	
face-recognition==1.2.2	
face-recognition-models==0.3.0	
Facenet [34, 10]==1.0.3	
tensorflow==1.13.1	
Keras-Applications==1.0.7	

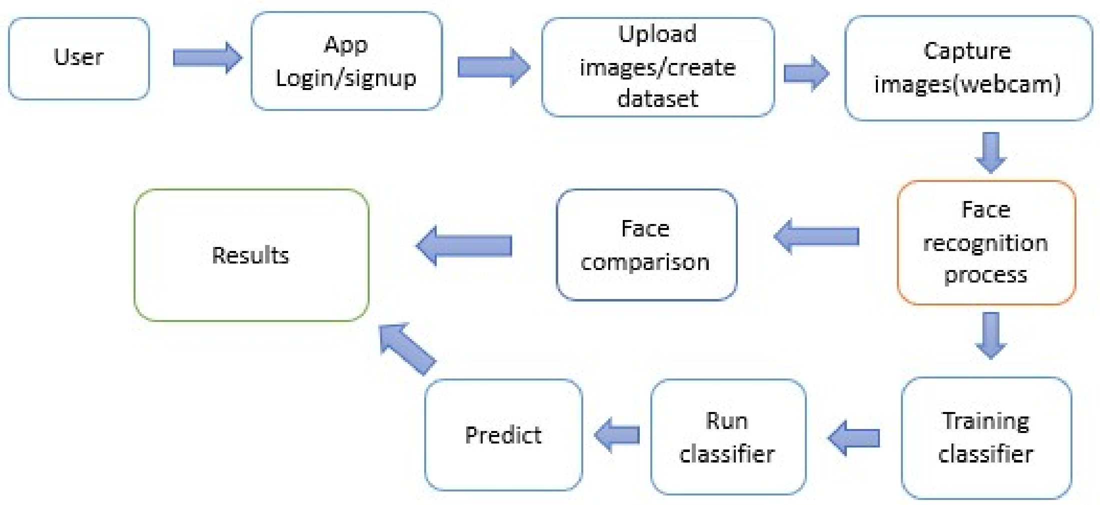


Figure 6.1: SYSTEM ARCHITECTURE

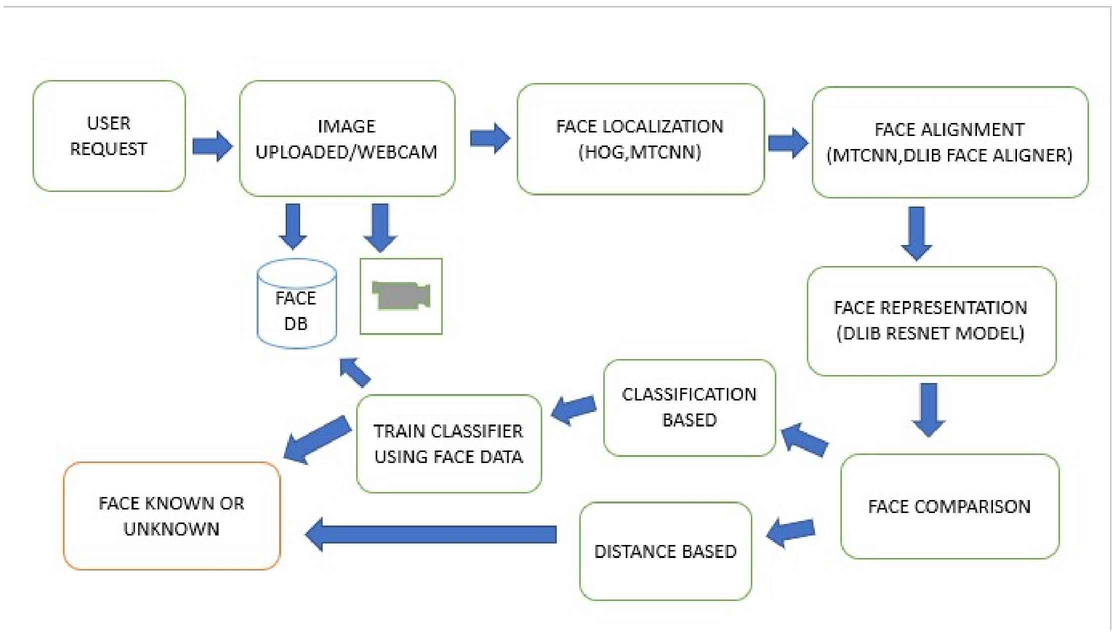

USER REQUEST
IMAGE UPLOADED/WEBCAM
FACE LOCALIZATION (HOG,MTCNN)
FACE ALIGNMENT (MTCNN,DLIB FACE ALIGNER)
FACE DB
FACE REPRESENTATION (DLIB RESNET MODEL)
CLASSIFICATION BASED
TRAIN CLASSIFIER USING FACE DATA
FACE COMPARISON
FACE KNOWN OR UNKNOWN
DISTANCE BASED

Figure 6.2: DATA FLOW: FACE RECOGNITION

CHAPTER 7

MODULES AND ITS FUNCTIONALITIES

In this project, we are developing a Web-based application to perform real-time modern facial recognition and tracking tasks. This application provides users with services like creating a personal account with the option of uploading images and creating their own dataset, capturing the digital image of people in real time through a well-framed camera, generating facial features and comparing against trained facial features of uploaded images present in the dataset of the user. This application also provides the option of training a classifier on their dataset to make a better prediction of faces detected in real time and using this application user can generate a detailed report in a PDF format containing details like list of individuals identified along with probability estimates and face distance measurements. This application provides a simple GUI for the users to quickly and easily set up their own modern facial recognition based security system.

Modules of our facial recognition and tracking application are:

Dataset creation and updation :

This component of our application is responsible for the creation of the personalized dataset. This component allows users to upload images containing faces. Here user can create a directory for each class or can directly upload images of different persons on whom user want to perform facial recognition and tracking tasks. In our application, there is no constraint that the user has to upload more than one image per class. For our application to perform facial recognition in real time a single good quality image will work fine. This is possible because our application performs face comparison using nearest neighbour and distance threshold technique. However to improve the accuracy and probability of face comparison our application also provides the option of training a classifier on the dataset created by the user by uploading images.For classifier to give meaningful results, users have to upload more than one image per class.Our application also provides very important and useful functionality of dataset updation where user can easily update their own dataset created by uploading images. Using our application user can easily delete images which user do not require. Our application gives complete details of images uploaded in the dataset in the form of table and quick option of delete to get rid of unwanted images in the dataset. Our application provides a complete view of images present in the dataset which includes original image thumbnail, upload time, details added by the user during the upload etc. User can easily view all these details on their homepage. This option really helps the user to quickly update their dataset.

Real-time Face Recognition using distance threshold technique:
This component of our application basically deals with performing facial recognition using tolerance value. This component performs real-time facial recognition using only a single good quality image per person. In this facial recognition technique, face comparison is done using the distance-based method. In the face comparison process, the face encodings which are generated for each face in the face representation process is used. In the distance-based method, the comparison is done by calculating the Euclidean distance between the target face and all the faces present in our dataset. A threshold value or tolerance value is learned by the system by utilizing the nearest neighbour search [21] method. If the distance is less than tolerance threshold value which means faces match. Otherwise, faces do not match. This type of learning technique is known as a one-shot learning technique. This type of technique for facial comparison in the facial recognition pipeline reduces time in training a classifier model and user can quickly perform real-time facial recognition and tracking tasks after uploading at least one image in the dataset. For using this component user has to simply choose the option of distance based facial recognition provided on the home page of the application. After selecting that option user is redirected to that componentś page where a user has to simply click start to begin the process.
This page also gives complete details like techniques used for all subprocesses in the face recognition, model details etc. After clicking the start button a new window will pop up on top of our application running the default camera module installed in the userćomputer machine. Now if anyone comes in in front of that camera and face is detected, our application starts predicting that person whether he/she is a known person or unknown person. All these processes happen in real time. This component of our application is very useful for those users who want to quickly set up their facial recog-

nition and tracking system in their security system.

Face Recognition using Classification method:

This component of our application basically deals with performing facial recognition using classification technique. In classification based approach we can use different classification algorithms like KNN, SVM etc. To include classification based face comparison module in our face recognition application we have used scikit learn [9] python [38] library. This library provides easy implementation of many classifications and regression-based ML algorithms. In our application, we are first training a classifier model using the images present in our database. Then we are using that trained classifier model for prediction of faces detected in real time and it also gives probability value for faces recognized. For using this component user has to simply select the option of face recognition using the classification method. In this type of face recognition process first, the user has to train a classifier model on the images which user want to recognize and then using that trained classifier model to perform real-time face prediction. The classifier also gives additional details like probability value for the matched faces. After selecting the classification option present on the home page of our application it redirects the user to that respective page. On this page, the user is provided with two options to select. The first option is training a classifier model and the second option is running the classifier.Training a classifier model once is mandatory provided dataset is not updated. For training a classifier properly so that it gives meaningful results during prediction there should be more than one image per class uploaded and stored in the database of that user. Once the user has trained the classifier model, then our application stores the trained classifier model with the help of python [38] pickle library. By this user do not have to train the classifier model everytime .Now user can simply select the option of run classifier and pop up window appears running the camera module installed in the userśystem and it starts predicting the faces detected by the camera in real time. The advantages of training a classifier are many like training a classifier on our own dataset gives more accurate results in real time and it also provides additional details like probability value for faces matched. However, training a classifier model is time-consuming on a large dataset.

Tracking System:

This component of our application is very useful for tracking purposes. This component is responsible for storing all the details captured during real-time face recognition. This tracking system of our application comes into the picture after the facial recognition process is completed. It provides the user with several features which include personalized tracking report containing all the necessary details like the total number of individuals detected, the total number of individuals matched with the faces stored in the useŕdataset, complete details of recognized individuals. Tracking component of our application also provides the user with useful details like computed euclidean face distance value between the faces matched depending on a preset distance threshold value, probability values etc. Using all these details user can take appropriate and accurate decisions. User is provided with two option of viewing these details first option is by user directly going to the result section of our application where user can access all these details in a tabular format. Another option is downloading a PDF report containing all these details. This option is useful if the user wants to quickly generate results in PDF format. This option in our application saves userś time and the user can also take away the results in PDF format and can print it easily.

User authentication:

This component of our application is necessary for accessing our application in order to perform face recognition and tracking tasks. This is the entry point of our application. This component basically deals with the creation of the user profile so that different users can access this application easily. It also provides users with a secure login page where users have to enter their credentials in order to access this application. For this first user has to get registered with our application.Then the user is provided with login credentials which user can use to securely access the application. After logging into the application user is redirected to userś dashboard page where user can safely upload images and can perform tracking and facial recognition tasks. Our application securely stores all these details in the applicationś database and the user can anytime access these details by logging into the application using its login credentials. For registration, users have to provide email id and password. So this component of our application allows all the users to securely use this application for their own purpose.

7.1 Django Web Framework

Our facial recognition and tracking application is built using Django which is a python [38] based web framework. Django provides all kinds of functionalities to quickly build web applications like content management system, social networking application, an application which can perform scientific computing and also an application which can provide a platform for doing machine learning and deep learning tasks. In our application, we are using this framework to build facial recognition and tracking application. There are several reasons for using Django for building our application which includes fast development, highly secure, scalable and reusable and it also provides dozens of plugins which helps in the rapid development of our application. The whole GUI of our facial recognition and tracking application is built with the help of the Django framework. Django framework obeys the Model View Template (MVT) pattern in the development. Django provides an inbuilt authentication system and administrative interface which user can directly use in its own application. Django also provides its own application routing service which makes it easy to develop an application without relying on any external routing service.

Django also provides relational database ORM for SQLite database and one can use any relational database for our application using this ORM.ORM stands for object-relational mapper which help in quick and easy creation of database and tables in our database without writing any complex queries. It uses data models for performing operations on the database. These data models are implemented using python [38] classes. Django provides functionalities for handling HTTP requests and also helps in rendering HTML web pages. Django adopts many OOPS techniques like inheritance for templates. Django also provides a platform for performing unit testing for our application which is another very useful feature.

Django also has pre-installed security-based applications which protect our application from various hacking attacks like CSRF, Sequential Query Language (SQL) injections, cracking of password etc. Django Software Foundation maintains and updates the Django framework. Django is a highly popular python [38] based web framework and its open source community is highly active and supportive. Because of all these reasons we have used Django for building our facial recognition and tracking application. Our application uses the SQLite database which is provided by Django during the development process for developing our application which later can be easily changed during the deployment stage depending on the application usage.

7.2 Database design

In our facial recognition and tracking application, we have used SQLite database provided by the Django framework. Django provides the ORM for performing operations on our database like database and table creation.Django uses python [38] classes for this purpose.these are referred to as Data model in our application. For our facial recognition and tracking application, our main requirement was to first create a database for users to store their images and they can create their own dataset for the facial recognition process. We also needed a database for storing details of the registered users and also details of the images uploaded by the user. We needed a database for storing all tracking details which are generated after the face recognition process. In our applicationś database we mainly used three tables : Users detail table having three columns for email id, password and for unique user id which acts a primary key for this table.Next table is for storing details of uploaded images which contain four columns: description column for storing details related to that particular image like name, purpose etc.,image column for storing the image file URL to the userś database. upload time column which stores the upload timestamp and profiles user column which stores the unique ID of that user. This Id acts as a foreign key in this table and application can now easily recognize all the details and uploaded data of any user. The last table is for storing tracking details it contains two columns first is unique user ID acting as a foreign key and other columns for storing all tracking details generated for that user.

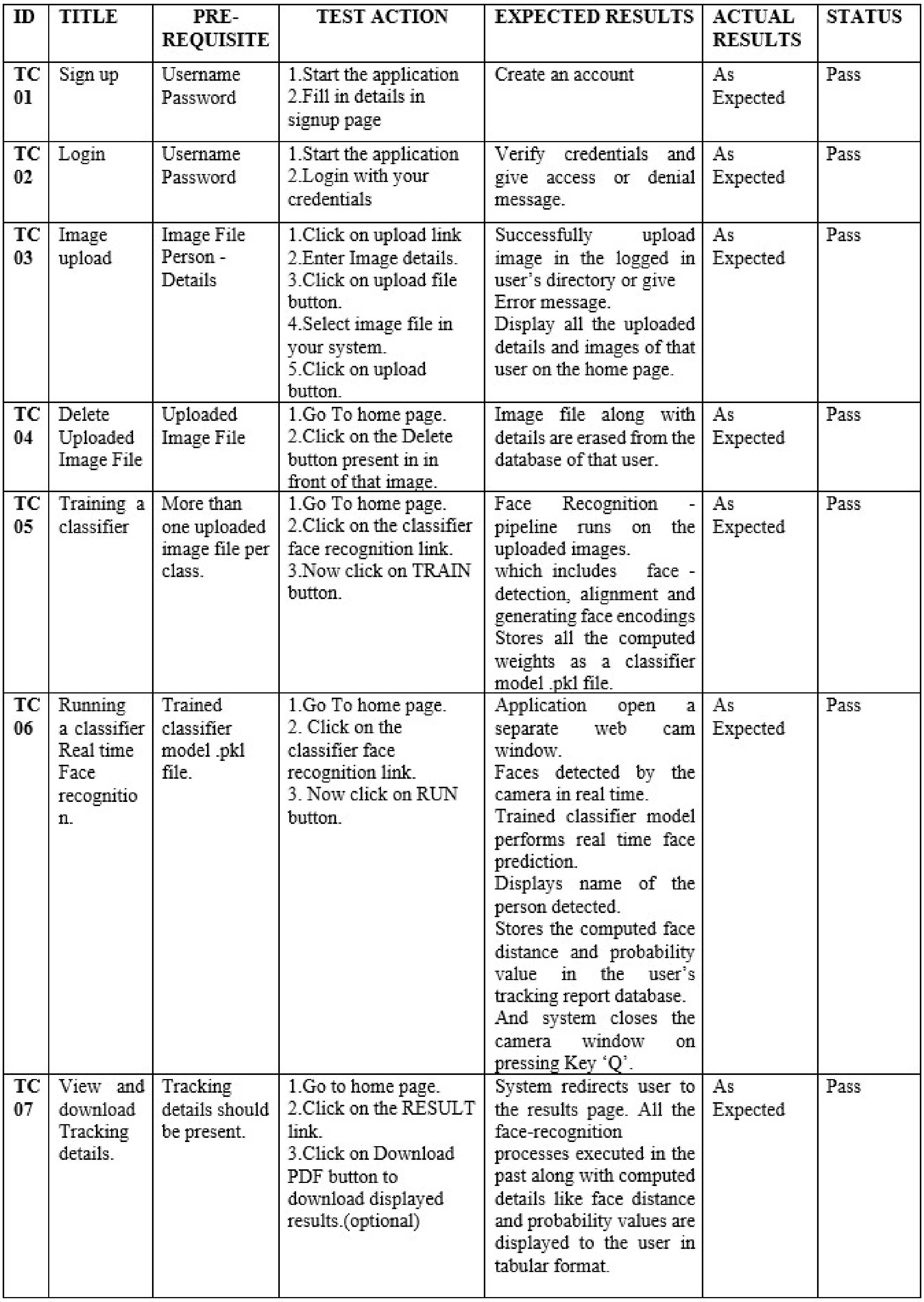
CHAPTER 8

SYSTEM TESTING

During the development as well as the completion phase of our facial recognition and tracking application we performed various kinds of testing on our application so that our application behaves appropriately and performs all the facial recognition and tracking tasks correctly and there should be no errors in our application and if in case any unexpected error case comes then our application should handle all those errors appropriately. Testing is very important for developing an application which is efficient and performs precisely in all situations. Testing is a beneficial part of developing an efficient application which does not have any kind of fault and weakness. Testing help in identifying all the problems which exist in the functionalities provided by our application. Testing helps in finding all kinds of coding bugs present in our application which ultimately leads to unexpected behaviour and incorrect results. Testing also checks all the functionalities present in our application. It checks whether all the components of our application are working as required or not. It also checks whether proper interaction is happening between different components of our application. Testing basically ensures that our application performs smoothly and appropriately and gives the correct output. It also ensures that our application gives the user a smooth and error-free experience while using all the functionalities provided by our application. There are various types of testing techniques available for testing different areas of the application depending on the application requirements. Some of the popular testing techniques are Unit testing, System testing, Integration testing, black and white box testing, functional testing etc. In our application, we have done unit testing using the inbuilt unit testing framework provided by the Django framework on which our facial recognition and tracking application is built. Django provides a framework for writing unit testing scripts for testing different components of our application. Unit testing technique deals with testing of individual components of our application. It checks whether each individual component of our application is getting required inputs and it is giving correct results. After completion of each of the component of our application, we ran the unit testing script written for that particular component and checked whether all the test cases written in the testing script for that component passes. If any test case failed then we corrected that part in our application. We repeated this process for all the individual components of our application until all the test cases passed.

For our application, we have also performed many testing manually like integration, system and functional testing. For performing integration testing for our application we have manually checked all the interaction that is happening between different components of our application. All these testing was done after integrating all the components of our facial recognition and tracking application. We checked whether all the components interacting uniformly or not and also check whether the request and response of each component are appropriate and correct. We also added print statements at the different critical section of our application and kept track of all the error message if any generated in the console. Integration testing is very important testing because it ensures all the components of our application performs appropriately and gives expected output after the integration. This testing is manually done after unit testing is completed successfully. We have set different testing objectives for our application and performed many testing manually like testing all the options and buttons in our application and their behaviour whether they are directing users to the correct page or not. We also checked whether all the input fields are working properly and proper validation is done on the input data and there is no data duplication. We also checked whether all the uploaded images are stored properly at the specified location or not. We also ensured that all the response are generated without any delay and gives the user a smooth experience. We also ensured that any unexpected error is handled properly by our application. We checked our application by creating many test users within our applicationś database and given various inputs to our application to ensure that the application gives correct results in all situations.

Our application is also tested by other users and feedback and reviews were taken from them and important changes were done. So performing all these testing on our application ensured that it performs facial recognition and tracking tasks efficiently and meets the desired objectives set for our application.All the test cases are shown in Figure.8.1.



ID	TITLE	PRE-REQUISITE	TEST ACTION	EXPECTED RESULTS	ACTUAL RESULTS	STATUS
TC 01	Sign up	Username Password	1.Start the application 2.Fill in details in signup page	Create an account	As Expected	Pass
TC 02	Login	Username Password	1.Start the application 2.Login with your credentials	Verify credentials and give access or denial message.	As Expected	Pass
TC 03	Image upload	Image File Person - Details	1.Click on upload link 2.Enter Image details. 3.Click on upload file button. 4.Select image file in your system. 5.Click on upload button.	Successfully upload image in the logged in user's directory or give Error message. Display all the uploaded details and images of that user on the home page.	As Expected	Pass
TC 04	Delete Uploaded Image File	Uploaded Image File	1.Go To home page. 2.Click on the Delete button present in in front of that image.	Image file along with details are erased from the database of that user.	As Expected	Pass
TC 05	Training a classifier	More than one uploaded image file per class.	1.Go To home page. 2.Click on the classifier face recognition link. 3.Now click on TRAIN button.	Face Recognition - pipeline runs on the uploaded images. which includes face - detection, alignment and generating face encodings Stores all the computed weights as a classifier model .pkl file.	As Expected	Pass
TC 06	Running a classifier Real time Face recognitio n.	Trained classifier model .pkl file.	1.Go To home page. 2. Click on the classifier face recognition link. 3. Now click on RUN button.	Application open a separate web cam window. Faces detected by the camera in real time. Trained classifier model performs real time face prediction. Displays name of the person detected. Stores the computed face distance and probability value in the user's tracking report database. And system closes the camera window on pressing Key 'Q'.	As Expected	Pass
TC 07	View and download Tracking details.	Tracking details should be present.	1.Go to home page. 2.Click on the RESULT link. 3.Click on Download PDF button to download displayed results.(optional)	System redirects user to the results page. All the face-recognition processes executed in the past along with computed details like face distance and probability values are displayed to the user in tabular format.	As Expected	Pass

Figure 8.1: TEST CASES

CHAPTER 9

RESULTS

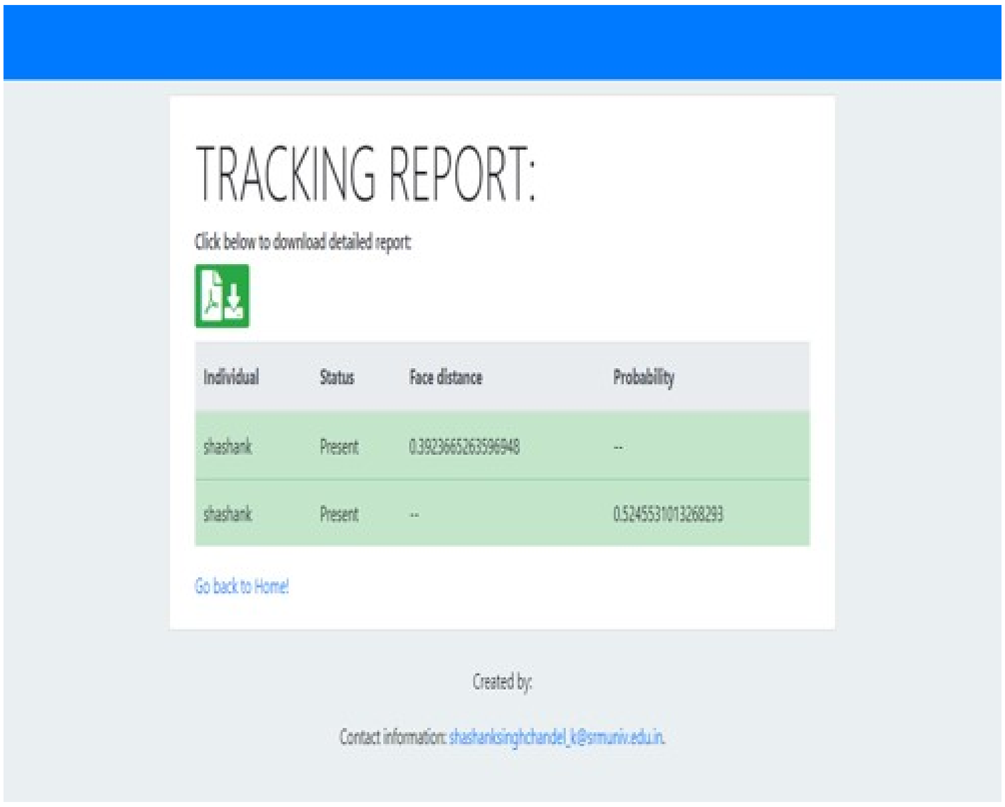
After completing the full testing of our application, We finally developed a web application that can perform real-time modern facial recognition and tracking tasks as shown in Figure.9.8. This application provides users with services like creating a personal account as shown in Figure.9.3 with the option of uploading images as shown in Figure.9.4 and creating their own dataset, capturing the digital image of people in real time through a well-framed camera, generating facial features and comparing against trained facial features of uploaded images present in the dataset of the user. This application also provided the option of training a classifier on their dataset to make a better prediction of faces detected in real time as shown in Figure.9.7 and using this application user can generate a detailed report in a PDF format containing details like list of individuals identified along with probability estimates and face distance measurements as shown in Figure.9.2.

Now we checked the results obtained in performing the facial recognition and tracking tasks because we want facial recognition process in our application to give high accuracy in face verification task in real time. After running our facial recognition process several times in real time on different face inputs with different face position and different lighting conditions, we achieved accuracy of around 52% in face matching in real time shown in Figure.9.1. We also computed Euclidean distance between the face encodings of the matched individuals and minimum value we achieved was around 0.32 and maximum face distance value was below 0.5. These values were verified after testing our facial recognition on several face inputs in real time conditions. Therefore we set a tolerance distance threshold for face comparison equal to 0.5. The author of the Dlib [23] library who trained the Dlib [23] Residual Neural Network (ResNet) [28, 18] model used tolerance threshold value equal to 0.6 but for our input dataset this tolerance distance threshold value was not appropriate and it was giving many wrong predictions in real time. The probability value of 52% was achieved after training a classifier on our dataset. These results are more than satisfactorily considering the whole face recognition process was executed without GPU.

All the techniques and algorithms we considered for the face recognition process do not require GPU to run. This shows that we can perform the facial recognition process with high accuracy on our mid-range personal machines.

In our application for solving different sub-problems like for face detection we used HOG [26] method, for face alignment we used Dlib [23] face aligner which gives 68 facial landmark points, for face representation we used pre-trained Dlib [23] ResNet [28, 18] model with 29 CNN layers. for face comparison, we used both the distance threshold technique as well as classification based method. However, we cannot deny the fact that face recognition techniques which utilize GPU performs better and gives better accuracy rate in recognizing faces in real time. Techniques like MTCNN [19] for face detection and alignment which consists of a cascade of CNN layers gives better results in real time as compared to HOG [26] and Dlib [23] face aligner. But MTCNN [19] requires GPU in the system to run. For face representation, various state of the art pre-trained model exists which performs better than Dlib [23] models like Facenet [34, 10] models which are based on very deep and complex architecture and they cannot run efficiently on CPU and smaller devices like mobile devices.However recently many models were proposed which are not heavy and can run on low-end devices like mobile devices and also give high performance.

Our application provided a simple GUI for the users to quickly and easily set up their own modern facial recognition based security system without any need of high-end GPU enabled computer machine. This application can also be used to set up facial recognition based security system in public places like shopping malls, stores etc.



TRACKING REPORT:

Click below to download detailed report:

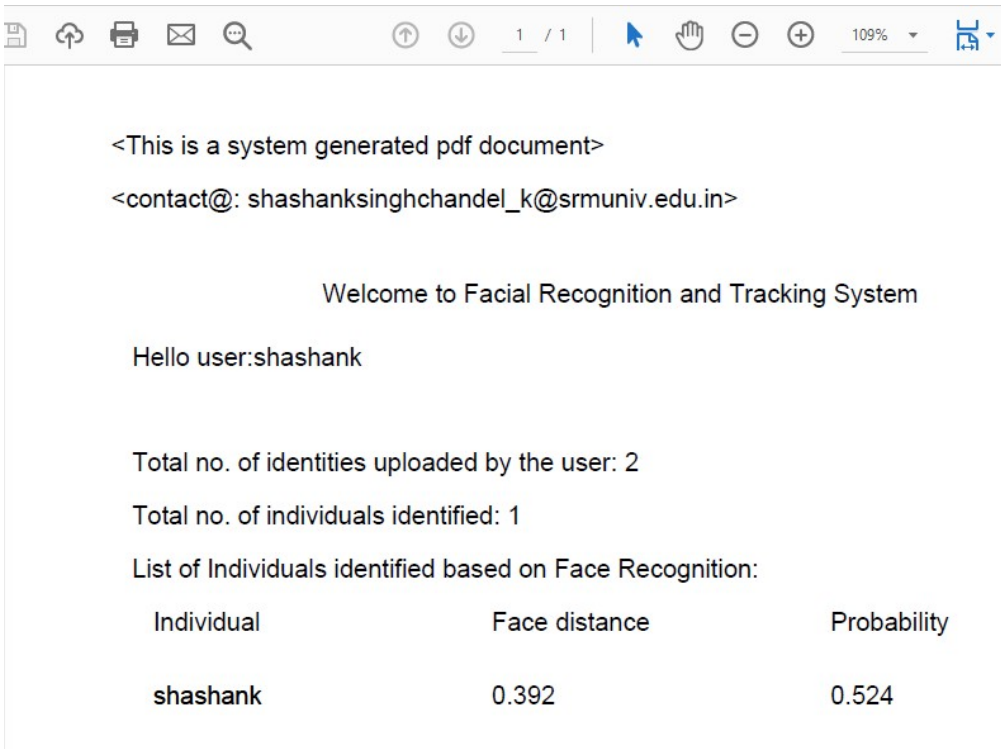
Individual	Status	Face distance	Probability
shashank	Present	0.3923665263596948	--
shashank	Present	--	0.5245531013268293

Go back to Home!

Created by:

Contact information: shashanksinghchandel_k@srmuniv.edu.in.

Figure 9.1: TRACKING REPORT



<This is a system generated pdf document>

<contact@: shashanksinghchandel_k@srmuniv.edu.in>

Welcome to Facial Recognition and Tracking System

Hello user:shashank

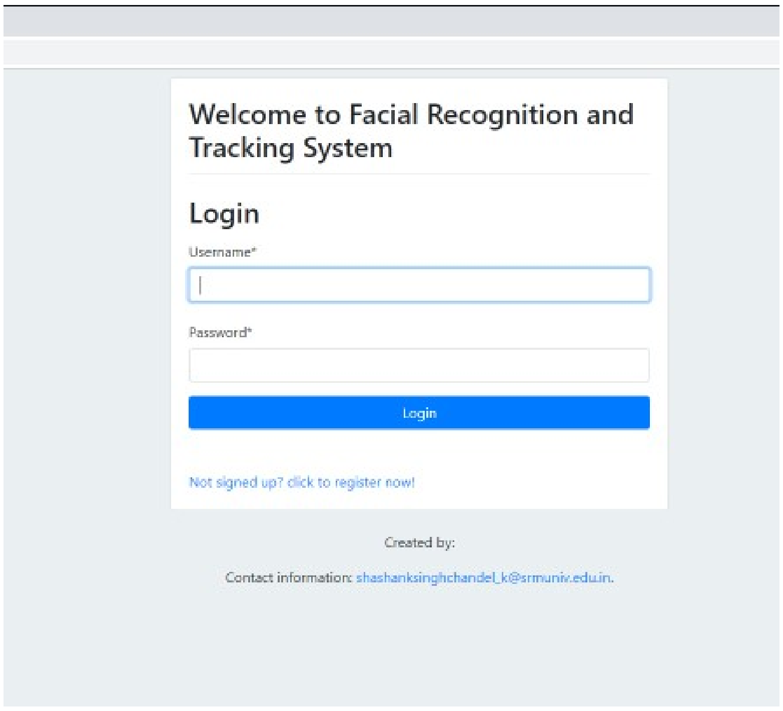
Total no. of identities uploaded by the user: 2

Total no. of individuals identified: 1

List of Individuals identified based on Face Recognition:

Individual	Face distance	Probability
shashank	0.392	0.524

Figure 9.2: REPORT GENERATED IN PDF


Welcome to Facial Recognition and Tracking System

Login

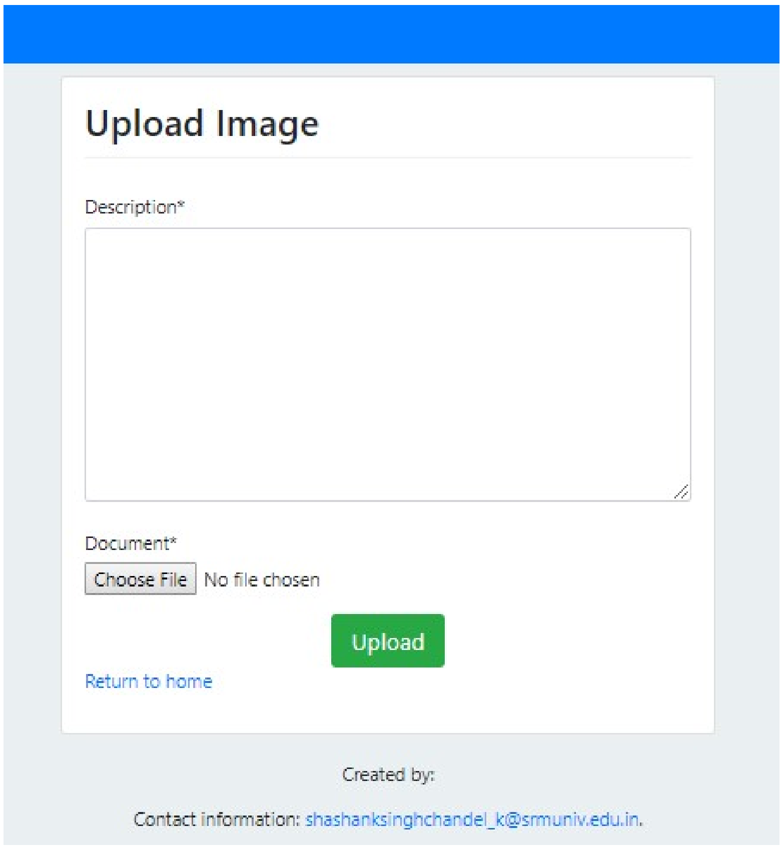
Username*

Password*

Created by:

Contact information: shashanksinghchandel_k@srmuniv.edu.in.

Figure 9.3: LOGIN PAGE


Upload Image

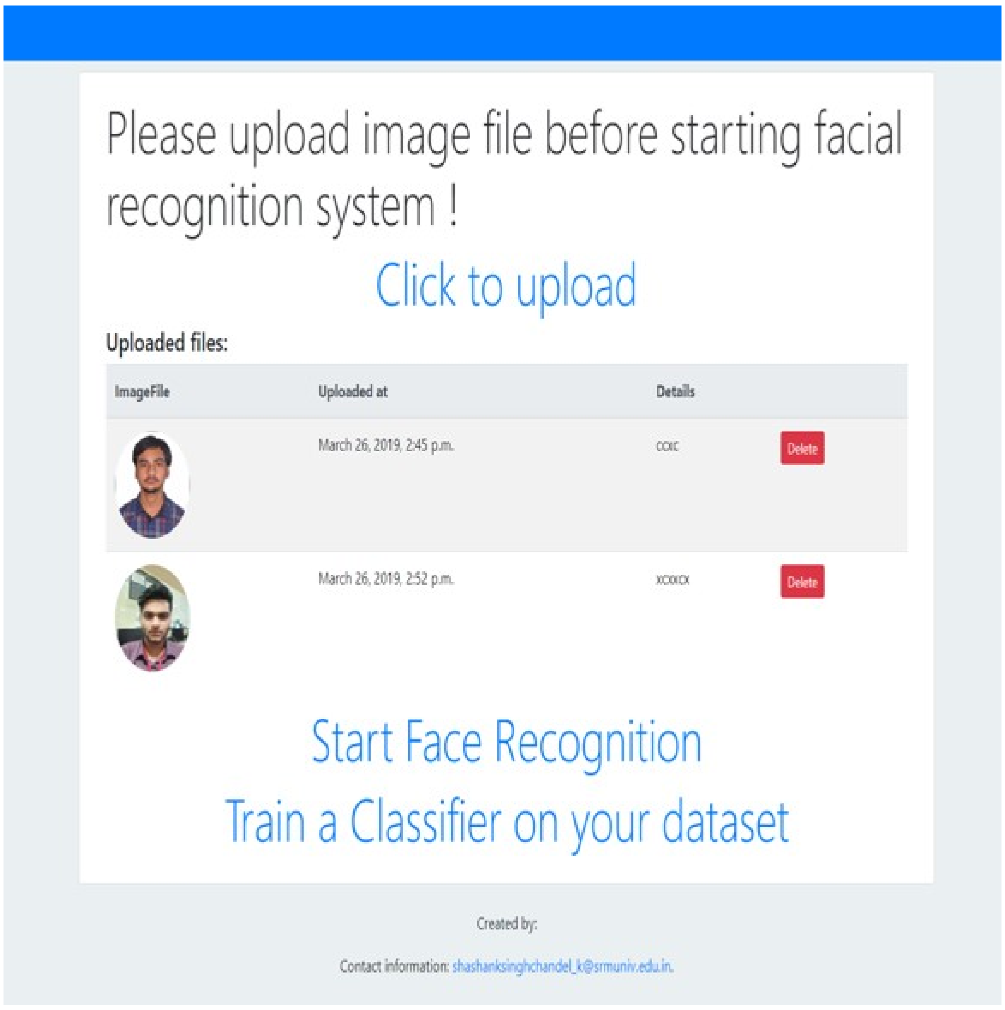
Description*

Document*

Created by:

Contact information: shashanksinghchandel_k@srmuniv.edu.in.

Figure 9.4: IMAGE UPLOAD PAGE



Please upload image file before starting facial recognition system !

Click to upload

Uploaded files:

ImageFile	Uploaded at	Details	
	March 26, 2019, 2:45 p.m.	cxxc	Delete
	March 26, 2019, 2:52 p.m.	xcxxcx	Delete

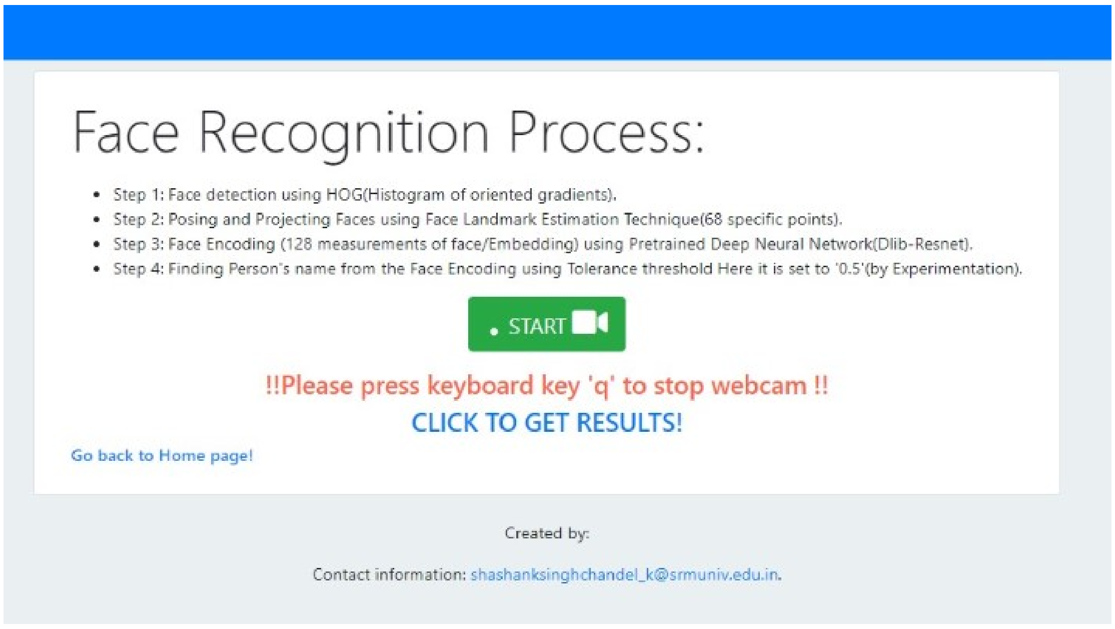
Start Face Recognition

Train a Classifier on your dataset

Created by:

Contact information: shashanksinghchandel_k@srmuniv.edu.in.

Figure 9.5: USER HOME PAGE



Face Recognition Process:

- Step 1: Face detection using HOG(Histogram of oriented gradients).
- Step 2: Posing and Projecting Faces using Face Landmark Estimation Technique(68 specific points).
- Step 3: Face Encoding (128 measurements of face/Embedding) using Pretrained Deep Neural Network(Dlib-Resnet).
- Step 4: Finding Person's name from the Face Encoding using Tolerance threshold Here it is set to '0.5'(by Experimentation).

START

!!Please press keyboard key 'q' to stop webcam !!

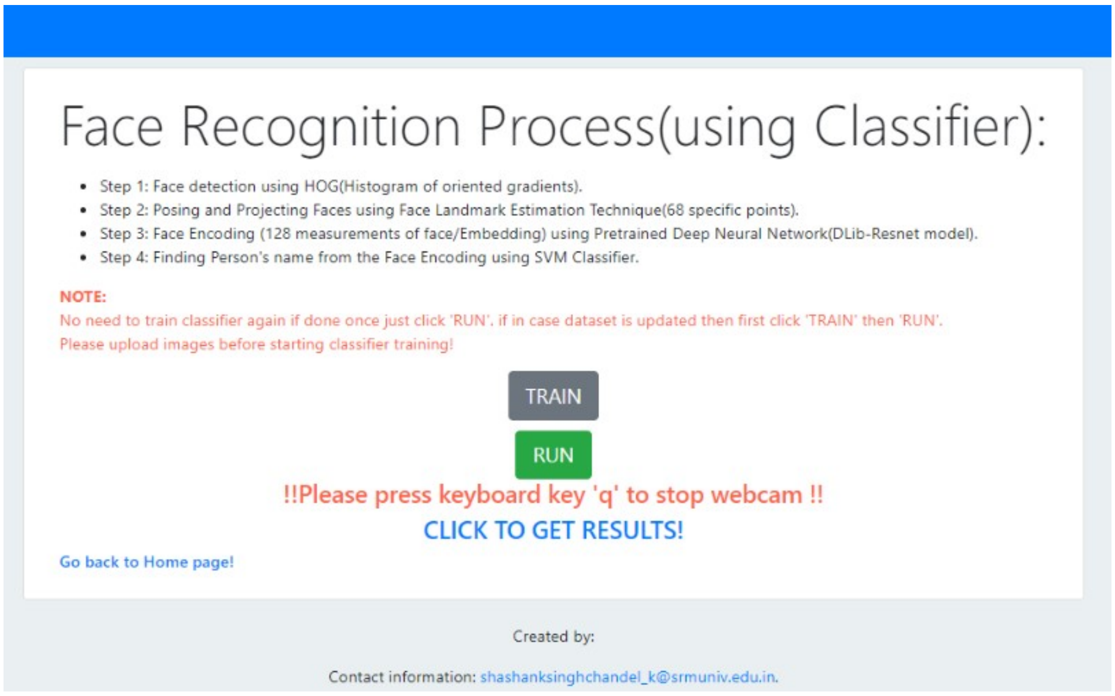
CLICK TO GET RESULTS!

Go back to Home page!

Created by:

Contact information: shashanksinghchandel_k@srmuniv.edu.in.

Figure 9.6: FACE RECOGNITION USING DISTANCE THRESHOLD



Face Recognition Process(using Classifier):

- Step 1: Face detection using HOG(Histogram of oriented gradients).
- Step 2: Posing and Projecting Faces using Face Landmark Estimation Technique(68 specific points).
- Step 3: Face Encoding (128 measurements of face/Embedding) using Pretrained Deep Neural Network(DLib-Resnet model).
- Step 4: Finding Person's name from the Face Encoding using SVM Classifier.

NOTE:
No need to train classifier again if done once just click 'RUN'. if in case dataset is updated then first click 'TRAIN' then 'RUN'.
Please upload images before starting classifier training!

TRAIN

RUN

!!Please press keyboard key 'q' to stop webcam !!

CLICK TO GET RESULTS!

Go back to Home page!

Created by:

Contact information: shashanksinghchandel_k@srmuniv.edu.in.

Figure 9.7: FACE RECOGNITION BY TRAINING A CLASSIFIER

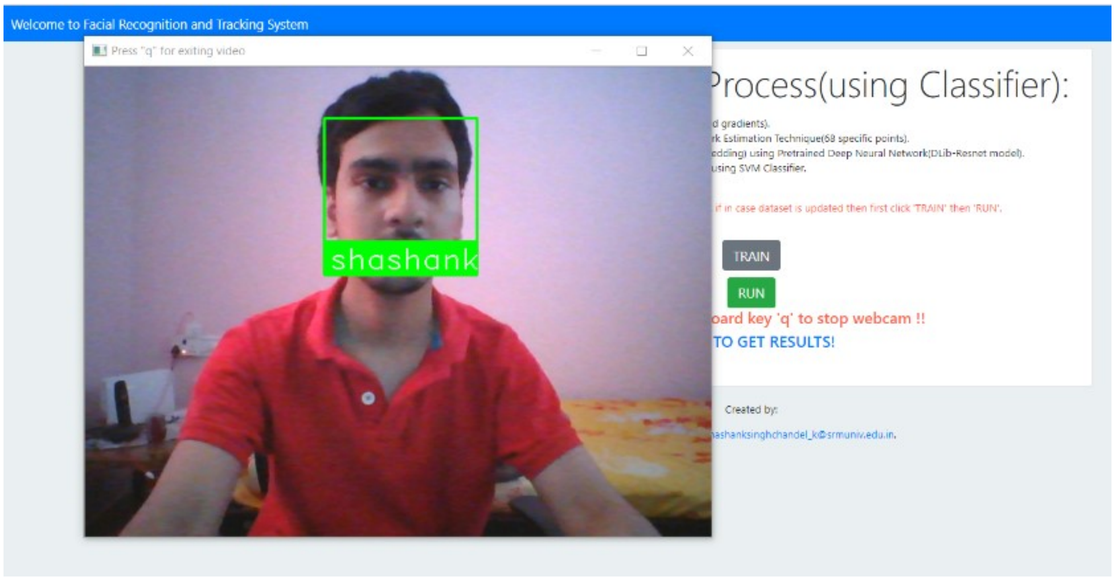

Welcome to Facial Recognition and Tracking System
Press "q" for exiting video
shashank
Process(using Classifier):
d gradients).
rk Estimation Technique(68 specific points).
edding) using Pretrained Deep Neural Network(DLib-Resnet model).
using SVM Classifier.
if in case dataset is updated then first click 'TRAIN' then 'RUN'.
TRAIN
RUN
oard key 'q' to stop webcam !!
TO GET RESULTS!
Created by:
ashanksinghchandel_k@srmuniv.edu.in.

Figure 9.8: REAL-TIME FACE PREDICTION

CHAPTER 10

CONCLUSION

In this project, we developed a web application to perform real-time modern facial recognition and tracking tasks. It provides users with services like creating a personal account with the option of uploading images, creating their own dataset, capturing the digital image of people in real time through a well-framed webcam, generating facial features and comparing against facial features present in the dataset of the user,training a classifier, generating detailed report in a PDF format containing individuals identified, probability and face distance value. This application is a significant step in making facial recognition technology easier for public use, people with no knowledge of the underlying complex processes in face recognition can also use this technology. This application provides better results in real-time face prediction. This application provides necessary details like probability, face distance value etc.for making any decisions. In this project report, we first discussed the use of biometric-based security systems in today's digital world.we discussed deep learning, a machine learning field. we discussed neural networks. In our facial recognition process, we are utilizing available deep learning based models, algorithms and techniques to achieve high performance in real time. In this report, we also discussed CNN, deep metric learning [8] and transfer learning. In face recognition pipeline following processes take place: face detection, alignment, feature extraction and comparison. We discussed existing techniques and models, pros and cons of different approaches and their requirements, deep learning based libraries used for developing face recognition for our application. Django framework was used for integration and for developing GUI.we also tested our application.For face detection,used HOG [26]as the default method and MTCNN [19],a CNN based technique for users with GPU installed,for alignment used Dlib [23]aligner which gives 68 facial landmarks [20],for measurements,used pre-trained Dlib [23] ResNet [28, 18] model. for comparison, used both the distance and classification based method. In the end, presented results obtained after using the facial recognition system of our application in real-time.

CHAPTER 11

FUTURE ENHANCEMENT

In this paper, we proposed a web application to perform real-time modern facial recognition and tracking tasks. This application provides users with services like creating a personal account with the option of uploading images and creating their own dataset, capturing the digital image of people in real time through a well-framed camera, generating facial features and comparing against trained facial features of uploaded images present in the dataset of the user. This application also provides the option of training a classifier on their dataset to make a better prediction of faces detected in real time and using this application user can generate a detailed report in a PDF format containing details like list of individuals identified along with probability estimates and face distance measurements. This application provides a simple GUI for the users to quickly and easily set up their own modern facial recognition based security system. This application can also be used to set up facial recognition based security system in public places like shopping malls, stores etc.
This application is a significant step towards making facial recognition technology easier for public use and one can use this technology through our application without the knowledge of underlying complex processes involved in the face recognition process. The facial recognition and tracking application proposed in this project have a lot of the scope in the future. The whole research work done during the development of this application in the face recognition field will help a lot in understanding all the underlying processes involved in the face recognition process and will act as a starting point in the development of more efficient face recognition and tracking application. Our application can easily be updated in the near future. Some of the future scopes for our application includes use of the better model with better underlying architecture for face representation, use of better techniques for face detection and alignment so that our system gives better accuracy in recognizing faces in real time.

CHAPTER 12

PAPER PUBLICATION STATUS

Publication process not yet started

REFERENCES

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

[2] Andrew G. Howard, M. Z. e. a. (2017). "Mobilenets: Efficient convolutional neural networks for mobile vision applications." *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[3] Bulat, A. and Tzimiropoulos, G. (2017). "How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks)." *The IEEE International Conference on Computer Vision (ICCV)*.

[4] Chollet, F. (2017). "Xception: Deep learning with depthwise separable convolutions." *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1251–1258.

[5] Christian Szegedy, S. I. e. a. (2016). "Inception-v4, inception-resnet and the impact of residual connections on learning." *The IEEE Conference on Computer Vision and Pattern Recognition*.

[6] Cournapeau, D. and et al (2018). "scikit-learn v0.20.2." *https://scikit-learn.org/stable/index.html*.

[7] Diederik P. Kingma, J. B. (2014). "Adam: A method for stochastic optimization." *International Conference for Learning Representations*.

[8] Elad Hoffer, N. A. (2015). "Deep metric learning using triplet network." *Springer International Publishing*, 84–92.

[9] et al, F. P. (2011). “Scikit-learn:machine learning in python.” *The Journal of Machine Learning Research*, 2825–2830.

[10] Florian Schroff, Dmitry Kalenichenko, J. P. (2015). “Facenet: A unified embedding for face recognition and clustering.” *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 815–823.

[11] Forrest N.Iandola, S. H. e. a. (2016). “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size.” *ICLR*.

[12] Geitgey, A. (2018). “face-recognition v1.2.3.” *https://face-recognition.readthedocs.io/en/latest/index.html*.

[13] Huaizu Jiang, E. L. M. (2017). “Face detection with the faster r-cnn.” *IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*.

[14] Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). “Densely connected convolutional networks.” *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4700–4708.

[15] Huang, G. B. e. a. (2008). “Labeled faces in the wild: A database forstudying face recognition in unconstrained environments.” *Workshop on Faces in ’Real-Life’ Images: Detection, Alignment, and Recognition*.

[16] Intel Corporation, W. G. (2018). “Opencv v4.0.1.” *https://opencv.org/*.

[17] Jiankang Deng, J. G. e. a. (2018). “Arcface: Additive angular margin loss for deep face recognition.” *The International Conference on Computer Vision and Pattern Recognition (CVPR)*.

[18] Kaiming He, X. Z. e. a. (2016). “Deep residual learning for image recognition.” *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

[19] Kaipeng Zhang, Z. Z. e. a. (2016). “Joint face detection and alignment using multitask cascaded convolutional networks.” *IEEE Signal Processing Letters*, 23, 1499–1503.

[20] Kazemi, V. and Sullivan, J. (2014). “One millisecond face alignment with an ensemble of regression trees.” *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1867–1874.

[21] Kilian Q. Weinberger, L. K. S. (2009). “Distance metric learning for large margin nearest neighbor classification.” *The Journal of Machine Learning Research*, 10, 207–244.

[22] King, D. E. (2009). “Dlib-ml: A machine learning toolkit.” *The Journal of Machine Learning Research*, 1755–1758.

[23] King, D. E. (2018). “Dlib v19.16.” *http://dlib.net/.*

[24] Liu, W. and et al, W. (2017). “Sphereface: Deep hypersphere embedding for face recognition.” *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[25] Murray, C. (2017). “Building a facial recognition pipeline with deep learning in tensorflow.” *https://hackernoon.com/building-a-facial-recognition-pipeline-with-deep-learning-in-tensorflow-66e7645015b8.*

[26] N.Dalal, B. (2005). “Histograms of oriented gradients for human detection.” *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, 886–893.

[27] Oliphant, T. (2019). “Numpy v1.16.2.” *http://www.numpy.org/.*

[28] Omkar M. Parkhi, A. V. e. a. (2015). “Deep residual learning for image recognition.” *Visual Geometry Group,Department of Engineering Science,University of Oxford.*

[29] P.Viola, M. (2001). “Rapid object detection using a boosted cascade of simple features.” *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[30] Qiong Cao, L. S. e. a. (2018). “Vggface2: A dataset for recognising faces across pose and age.” *13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*.

[31] Ren, Shaoqing, H. e. a. (2015). “Faster r-cnn:towards real-time object detection with region proposal networks.” *Advances in Neural Information Processing Systems 28*, 91–99.

[32] Rosebrock, A. (2017). “Facial landmarks with dlib, opencv, and python - pyimagesearch..” *https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/*.

[33] Ruder, S. (2016). “An overview of gradient descent optimization algorithms.” *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, abs/1609.04747.

[34] Sandberg, D. (2018). “Face recognition using tensorflow.” *https://github.com/davidsandberg/misc*.

[35] Sandler, M., Howard, A., and et al, Z. (2018). “Mobilenetv2: Inverted residuals and linear bottlenecks.” *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[36] Sheng Chen, Y. L. e. a. (2018). “Mobilefacenets: Efficient cnns for accurate real-time face verification on mobile devices.” *Springer International Publishing*, 428–438.

[37] stanford university (2018). “Cs231n convolutional neural networks for visual recognition.” *https://www.youtube.com/playlist?list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv*.

[38] van Rossum, G. (2018). “Python v3.7.2.” *https://www.python.org/*.

[39] Yaniv Taigman, M. Y. e. a. (2014a). “Deepface: Closing the gap to human-level performance in face verification.” *The IEEE Conference on Computer Vision and Pattern Recognition*, 1701–1708.

[40] Yaniv Taigman, M. Y. e. a. (2014b). “Dropout: A simple way to prevent neural networks from overfitting.” *Journal of Machine Learning Research*, 1929–1958.

[41] YoutuLab, T. (2018). “Face analysis.” *http://bestimage.qq.com/faceanalyze.html*.

[42] Yuanyi Zhong, Jiansheng Chen, B. H. (2017). “Toward end-to-end face recognition through alignment learning.” *IEEE Signal Processing Letters*, 24, 1213 – 1217.

APPENDIX A

CODE

```
from django.shortcuts import render, redirect, get_object_or_404
from django.conf import settings
from django.http import HttpResponse
from django.core.files.storage import FileSystemStorage
from django.contrib.auth.models import User
import io
from django.http import FileResponse
from reportlab.pdfgen import canvas
from uploads.models import Document
from uploads.forms import DocumentForm
import face_recognition
import cv2
import os
import glob
import pickle
import numpy as np
import dlib
from sklearn import svm
from celery import task
nme=
nm=
from django.contrib.auth import logout
def logout_view(request):
nme.clear()
nm.clear()
logout(request)
return render(request,’end.html’)
```

```
def results_view(request):
if request.user.is_authenticated:
if request.method=="POST":
response = HttpResponse(content_type='application/pdf')
response['Content-Disposition'] = 'attachment; filename="results.pdf"'
buffer = io.BytesIO()
p = canvas.Canvas(buffer)
p.drawString(50, 800, "<This is a system generated pdf document>")
p.drawString(50, 775, "<contact@: shashanksinghchandel_k@srmuniv.edu.in>")
p.drawString(
150,730, "Welcome to Facial Recognition and Tracking System")
p.drawString(60,700,"Hello user:"+str(request.user))
documents = Document.objects.filter(
profile_user=request.user).count()
p.drawString(60, 650, "Total no. of identities uploaded by the user: "+str(documents))
p.drawString(60, 625, "Total no. of individuals identified: "+str(len(nme)))
p.drawString(60,600,"List of Individuals identified based on Face Recognition:")
p.drawString(70, 575, "Individual")
p.drawString(230, 575, "Face distance")
p.drawString(390, 575, "Probability")
i=540
for name,dist in nme.items():
p.drawString(70, i, str(name))
p.drawString(230, i, ':.5'.format(str(dist)))
i-=50
i=540
for name,prob in nm.items():
p.drawString(70, i, str(name))
p.drawString(390, i, ':.5'.format(str(prob[0])))
i -= 50
p.showPage()
```

```
p.save()
pdf = buffer.getvalue()
buffer.close()
response.write(pdf)
return response
context=
’dist_tab’:nme,
’prob_tab’:nm
return render(request,’results.html’,context)
else:
return redirect("login")

def delete_view(request,id):
if request.user.is_authenticated:
if request.method == ’POST’:
doc = Document.objects.get(id=id)
doc.delete()
return redirect("home")
else:
return redirect("login")

def home(request):
if request.user.is_authenticated:
documents = Document.objects.filter(profile_user=request.user)
return render(request, ’home.html’, ’documents’: documents)
else :
return redirect("login")
def simple_upload(request):
if request.method == ’POST’ and request.FILES[’myfile’]:
myfile = request.FILES[’myfile’]
fs = FileSystemStorage()
filename = fs.save(myfile.name, myfile)
```

```
uploaded_file_url = fs.url(filename)
return render(request, ’simple_upload.html’,
’uploaded_file_url’: uploaded_file_url
)
return render(request, ’simple_upload.html’)

def model_form_upload(request):
if request.method == ’POST’:
form = DocumentForm(request.POST, request.FILES)
if form.is_valid():
newdoc=form.save(commit=False)
newdoc.profile_user=request.user
newdoc.keep_owner(request.user)
newdoc.save()
return redirect(’home’)
else:
form = DocumentForm()
return render(request, ’model_form_upload.html’,
’form’: form
)

def face_recognition_view(request):
img_dir = os.getcwd()
pt = "media/user_"+str(request.user)
img = os.path.join(img_dir, pt)
face_locations = []
face_encodings = []
face_names = []
known_face_encodings = []
known_face_names = []
process_this_frame = True
if request.method == "POST":
```

```
files = os.listdir(img)
print(files)
for file in files:
print(file)
sample_image = face_recognition.load_image_file(img+"/"+file)
st = str(file).split(".jpg")[0]
print(st)
known_face_names.append(st)
sample_face_encoding = face_recognition.face_encodings(sample_image)[
0] known_face_encodings.append(sample_face_encoding)
video_capture = cv2.VideoCapture(0)
while True:
ret, frame = video_capture.read()
small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
rgb_small_frame = small_frame[:, :, ::-1]
if process_this_frame:
face_locations = face_recognition.face_locations(
rgb_small_frame, model='cnn')
face_encodings = face_recognition.face_encodings(
rgb_small_frame, face_locations)
face_names = []
for face_encoding in face_encodings:
matches = face_recognition.compare_faces(
known_face_encodings, face_encoding, tolerance=0.5)
fdist= face_recognition.face_distance(known_face_encodings,face_encoding).tolist()
print(fdist)
name = "Unknown"
if True in matches:
first_match_index = matches.index(True)
name = known_face_names[first_match_index]
fd=fdist[first_match_index]
face_names.append(name)
```

```
nme[name]=fd
process_this_frame = not process_this_frame
for (top, right, bottom, left), name in zip(face_locations, face_names):
top *= 4
right *= 4
bottom *= 4
left *= 4
cv2.rectangle(frame, (left, top),
(right, bottom), (255, 0, 0), 2)
cv2.rectangle(frame, (left, bottom - 35),
(right, bottom), (255, 0, 0), cv2.FILLED)
font = cv2.FONT_HERSHEY_DUPLEX
cv2.putText(frame, name, (left + 6, bottom - 6),
font, 1.0, (255, 255, 255), 1)
cv2.imshow('Press "q" for exiting video', frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
break
video_capture.release()
cv2.destroyAllWindows()
return render(request, "recognition.html", )

def face_recognition_classify(request):
face_locations = []
face_encodings = []
face_names = []
process_this_frame = True
if request.method == "POST" and request.user.is_authenticated:
classifier_filename = str(request.user)+'_classifier.pkl'
model= pickle.load(open(classifier_filename,'rb'))
video_capture = cv2.VideoCapture(0)
while True:
ret, frame = video_capture.read()
```

```
small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
rgb_small_frame = small_frame[:, :, ::-1]
if process_this_frame:
face_locations = face_recognition.face_locations(
rgb_small_frame, model='cnn')
face_encodings = face_recognition.face_encodings(
rgb_small_frame, face_locations)
face_names = []
for face_encoding in face_encodings:
matches = model.predict(face_encoding.reshape(1, -1))
match = model.predict_proba(face_encoding.reshape(1, -1))
name = "Unknown"
if len(matches)!=0:
face_names=matches
ind = np.argmax(match, axis=1)
bsp = match[np.arange(len(ind)), ind]
nm[matches[0]]=bsp
else:
face_names.append(name)
process_this_frame = not process_this_frame
for (top, right, bottom, left), name in zip(face_locations, face_names):
top *= 4
right *= 4
bottom *= 4
left *= 4
cv2.rectangle(frame, (left, top),
(right, bottom), (0, 255, 0), 2)
cv2.rectangle(frame, (left, bottom - 35),
(right, bottom), (0, 255, 0), cv2.FILLED)
font = cv2.FONT_HERSHEY_DUPLEX
cv2.putText(frame, name, (left + 6, bottom - 6),
font, 1.0, (255, 255, 255), 1)
```

```
cv2.imshow('Press "q" for exiting video', frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
break
video_capture.release()
cv2.destroyAllWindows()
return redirect("face_classify")
return render(request, "train.html", )

def train_view(request):
known_face_encodings = []
known_face_names = []
img_dir = os.getcwd()
pt = "media/user_"+str(request.user)
img = os.path.join(img_dir, pt)
if request.method == "POST" and request.user.is_authenticated:
files = os.listdir(img)
for file in files:
sample_image = face_recognition.load_image_file(img+"/"+file)
st = str(file).split(".jpg")[0]
print(st)
known_face_names.append(st)
sample_face_encoding = face_recognition.face_encodings(
sample_image, num_jitters=10)[0]
known_face_encodings.append(sample_face_encoding.tolist())
clf = svm.SVC(kernel='linear', probability=True)
clf.fit(known_face_encodings, known_face_names)
classifier_filename = str(request.user)+'_classifier.pkl'
pickle.dump(clf, open(classifier_filename, 'wb'))
return redirect("train_view")
return render(request, 'classifier.html', )
```