Ahram Canadian University

Faculty of Computer Science and Information Technology

Face Detection and Emotion

Submitted by:

#	ID	Name
1	41510114	AbdelRahman Awais Shaban
2	41510105	Alhassan Mohamed Abdel Gelel
3	41510210	Yasmine Mustafa Mohamed
4	41510092	Seif El Eslam Mahmoud
5	41510195	Adham Essam Morsy

A dissertation submitted in partial fulfillment of the requirements for the degree of Bachelor of computer science and information technology

Supervised by:

Dr. Tarek Abdelhamid

Egypt 2019

Committee Report

We certify that we have read this graduation project report as examining committee, examine the student in its content and that in our opinion it its adequate as a project document for "Integrated Electronic Healthcare Insurance System".

Chairman:	***Supervisor:***
Name: **Dr. Sherif El shafay**	Name: **Dr. Tarek Abdelhamid**
Signature:	Signature:
Date:1 /7 /2019	Date: 1/7 /2019
Examiner:	***Examiner:***
Name: **Dr. Ahmed Seif**	Name: **Dr. Ehab Talkhan**
Signature:	Signature:
Date: 1/7 /2019	Date: 1/7 /2019

Abstract

Our world today happens to develop extremely fast when it comes to technology as its market been expanding widely throughout the past years, hence humans' proactive interactions and acknowledging technology demands had them aware of it comprehensively, Face detection method is one of those inventions, becoming a more and more important technique in our social lives. From face detection technology implemented in our cheap cameras to intelligent agencies' sophisticated global Skynet surveillance system, such techniques have been widely used in a large number of areas and the market is still growing with a high speed. Face detection has been an active research area with many successful traditional and deep learning methods. In our project, we are introducing a new convolutional neural network method to tackle face detection to maintain a high accuracy while running in real time.

Facial expression recognition is one of the ongoing problems in computer biometrics. Robust neutral face recognition in real time is a major challenge for various supervised learning-based facial expression recognition methods. This is due to the fact that supervised methods cannot accommodate all appearance variability across the faces with respect to race, pose, lighting, facial biases, and so on, in the limited amount of training data. Moreover, processing each and every frame to classify emotions is not required, as the user stays neutral for the majority of the time in usual applications like video chat or photo album/web browsing. Detecting a neutral state at an early stage, thereby bypassing those frames from emotion classification would save computational power. In this paper, we propose a light-weight neutral versus emotion classification engine, which acts as a preprocessor to the traditional supervised emotion classification approaches. It dynamically learns neutral appearance at key emotion (KE) points using a statistical texture model, constructed by a set of reference neutral frames for each user. The proposed method is made robust to various types of user head motions by accounting for affine distortions based on a statistical texture model. Robustness to a dynamic shift of KE points is achieved by evaluating the similarities on a subset of neighborhood patches around each KEY point using the prior information regarding the directionality of specific facial action units acting on the respective KE point. The proposed method, as a result, improves emotion recognition (ER) accuracy and simultaneously reduces the computational complexity of the ER system, as validated on multiple databases.

Acknowledgement

The authors would like to extend their thanks and gratitude to their supervisor Dr. Tarek Abdelhamid for continuous guidance and encouragement throughout the project.

List of Figures

List of Tables

Table of Contents

INTRODUCTION Chapter 1

1.1 Overview

Humans have been using physical characteristics such as facial gestures, sound diversity, etc. to recognize each other for thousands of years. With new advances in technology, biometrics has become an emerging technology for recognizing individuals using their biological traits. Now, biometrics is becoming part of day to day life, where a person is recognized by his/her personal biological characteristics. Our goal is to develop an inexpensive security surveillance system, which will be able to detect and identify facial and body characteristics in adverse weather conditions. There are many factors which influence this type of methods i.e. lighting condition background noise, fog, and rain.

Particular attention is given to face recognition. Face recognition refers to an automated or semi-automated process of matching facial images. Many techniques are available to apply face recognition one of them is Principle Component Analysis (PCA). PCA is a way of identifying patterns in data and expressing the data in such a way to highlight their similarities and differences. We will add a property of facial emotion, for example, is this person happy or sad.

WHAT IS FACE RECOGNITION?

Face recognition is the task of identifying an already detected object as a known or unknown face, and in more advanced cases, telling exactly who is face it is (Figure 1).
Often the problem of face recognition is confused with the problem of face detection.
Face Detection is to identify an object as a "face" and locate it in the input image.
Face Recognition on the other hand is to decide if the "face" is someone known, or unknown, using for this purpose a database of faces in order to validate this input face.

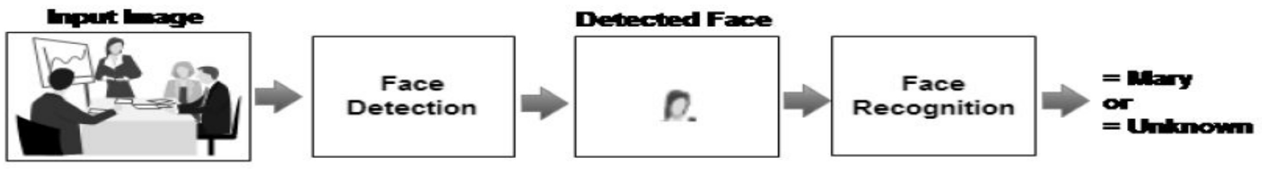


Figure [1] : Face Recognition

1.2 Motivation

face detection and recognition system can help in many ways:

- Access Control.
- Face Databases.
- Face ID.
- HCI - Human Computer Interaction.
- Law Enforcement.
- Multimedia Management.
- Security.
- Smart Cards.
- Surveillance.
- Others.

Why we chose the Face Detection Project?

Compatible with Modern Era.Basic program for Recognition (Recognition is not possible without Detection).Security Maintenance and Media Empowering.Needed for visual applications in Robotics.

1.3 Objective

Our objective is to make a system that will use computer vision techniques to detect and identify faces from the digital images which are extracted from the input images. The identification and recognition is based on prominent facial features such as region of the eyes, face shape etc. The main objectives of this project are stated below:

1. We are trying to build a fast and efficient face recognition system that detects faces very quickly in cluttered backgrounds. Using a YOLO object detection algorithm.

2. Once the face detection part is done, our next motive is to train our system with sufficient images .For each image, a feature vector is to be computed using CNN.

[2]YOLO ("you only look once") is a popular algorithm because it achieves high accuracy while also being able to run in real-time. This algorithm "*only looks once*" at the image in the sense that it requires only one forward propagation pass through the network to make predictions. After *non-max suppression*, it then outputs recognized objects together with the bounding boxes.

1.4 Aim

The main aim is face identification and verification, face recognition is a method of biometric authentication, based on extraction features of the face of an individual's image. Each individual has a unique face, Biometric face recognition systems should provide a reliable personal recognition schemes to either confirm or determine the identity of a person.

- **Face detection**

  The main function of this step is to determine :

  1 -whether human faces appear in a given image.

  2- where these faces are located at.

  The expected outputs of this step are patches containing each face in the input image. In order to make further face recognition system more robust and easy to design, face alignment are performed to justify the scales and orientations of these patches. Besides serving as the pre-processing for face recognition, face detection could be used for region-of-interest detection, retargeting, video and image classification. [3] Face Detection Feature Extraction Face Recognition Image Verification/ Identification .

- **Feature Extraction**

  Directly using these patches for face recognition have some disadvantages, first, each patch usually contains over 1000 pixels, which are too large to build a robust recognition system . Second, face patches may be taken from different camera alignments, with different face expressions, illuminations, and may suffer from occlusion and clutter. To overcome these drawbacks, feature extractions are performed to do information packing, dimension reduction, salience extraction, and noise cleaning. After this step, a face patch is usually transformed into a vector with fixed dimension or a set of fiducially points and their corresponding locations. In some literatures, feature extraction is either included in face detection or face recognition. [8]

1.5 Scope

Face detection and recognition has its applicability in various fields.the main scope for this project achieve the most friendly user experience and user interface for the users. The future scope can be stated as:

1. Room for improving accuracy:
   The accuracy for multiclass classifier can be improved. Various other techniques can be implemented and compared to obtain better accuracy results for a large database.

2. On getting a better accuracy we can use it in different fields for the purpose of security.

1.6 Hardware and Software Requirements

❖ Software Requirements:-
   1. Opencv 2.0 .
   2. Python 2.7 or 3.0 .

❖ Hardware Requirements:-
   1. Web camera .
   2. Hard disk:Minimum 1GB hard disk space.
   3. RAM:Minimum 64MB primary memory.

❖ Data Set:-
   1. Dataset created by a group of people manually.

Background knowledge and Previous work — Chapter 2

Identification of faces was work field of some researchers in previous years because it can facilitate the recognition, and this problem was addressed by several methods. Simple review about some solutions are shown to readers in order to have a good background about this topic.

- **Face Recognition**

  Face recognition has been an active research topic since the 1970's . Given an input image with multiple faces, face recognition systems typically first run face detection to isolate the faces. Each face is preprocessed and then a low-dimensional representation (or embedding) is obtained. A low-dimensional representation is important for efficient classification. Challenges in face recognition arise because the face is not a rigid object and images can be taken from many different viewpoints of the face. Face representations need to be resilient to intrapersonal image variations such as age, expressions, and styling while distinguishing between interpersonal image variations between different people . Jafri and Arabnia provide a comprehensive survey of face recognition techniques up to 2009. To summarize the regimes, face recognition research can be characterized into feature based and holistic approaches. The earliest work in face recognition was feature-based and sought

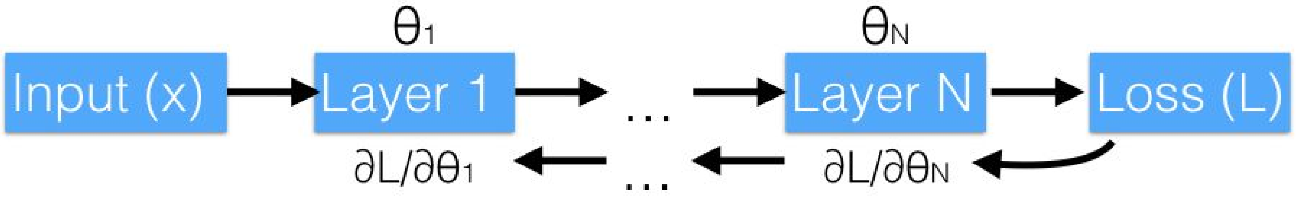


Figure [2]: Training flow for a feed-forward neural network.

to explicitly define a low-dimensional face representation based on ratios of distances, areas, and angles . An explicitly defined face representation is desirable for an intuitive feature space and technique. However, in practice, explicitly defined representations are not accurate. Later work sought to use holistic approaches stemming from statistics and Artificial Intelligence (AI) that learn from and perform well on a dataset of face images. Statistical techniques such as Principal Component Analysis (PCA) represent faces as a combination of eigenvectors. Eigenfaces and fisherfaces are landmark techniques in PCA-based face recognition. Lawrence et al.present an AI technique that uses convolutional neural networks to classify an image of a face. Today's top-performing face recognition techniques are based on convolutional neural networks. Facebook's DeepFace and Google's FaceNet systems yield the highest accuracy. However, these deep neural network-based techniques are trained with private datasets containing millions of social media images that are orders of magnitude larger than available datasets for research.

- **Face Recognition with Neural Networks**

This section provides a deeper introduction to face recognition with neural networks from the techniques used in Facebook's DeepFace and Google's FaceNet systems that are used within OpenFace. This section is intended to give an overview of the two most impactful works in this space and is not meant to be a comprehensive overview of the thriving field of neural network-based face recognition. Other notable efforts in face recognition with deep neural networks include the Visual Geometry Group (VGG) Face Descriptor and Lightened Convolutional Neural Networks (CNNs) , which have also released code.

A feed-forward neural network consists of many function compositions, or layers, followed by a loss function L as shown in Figure 1. The loss function measures how well the neural network models the data, for example how accurately the neural network classifies an image. Each layer i is parameterized by θ
, which can be a vector or matrix. Common layer operations are:

- **Spatial convolutions** that slide a kernel over the input feature maps,
- **Linear** or fully connected layers that take a weighted sum of all the input units, and
- **Pooling** that take the max, average, or Euclidean norm over spatial regions.

These operations are often followed by a nonlinear activation function, such as Rectified Linear Units (ReLUs), which are defined by f(x) = max{0, x}. Neural network training is a (nonconvex) optimization problem that finds a θ that minimizes (or maximizes) L. With differentiable layers, can be computed with backpropagation. The optimization problem is then solved with a first-order method, which iteratively progress towards the optimal value based on . See for a more thorough introduction to modern deep neural networks.

Figure 3 shows the logic flow for face

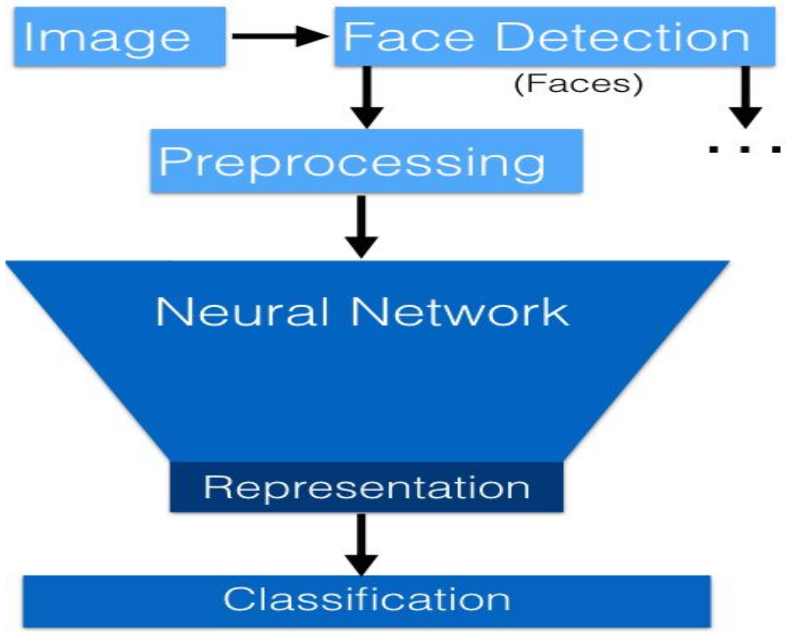


Figure [3]: Logic flow for face recognition with a neural network.

recognition with neural networks. There are many face detection methods to choose from, as it is another active research topic in computer vision. Once a face is detected, the systems preprocess each face in the image to create a normalized and fixed-size input to the neural network. The preprocessed images are too high-dimensional for a classifier to take directly on input. The neural network is used as a feature extractor to produce a low dimensional representation that characterizes a person’s face. A low-dimensional representation is key so it can be efficiently used in classifiers or clustering techniques.

DeepFace first preprocesses a face by using 3D face modeling to normalize the input image so that it appears as a frontal face even if the image was taken from a different angle. DeepFace then defines classification as a fully connected neural network layer with a softmax function, which makes the network's output a normalized probability distribution over identities. The neural network predicts some probability distribution p ˆ and the loss function L measures how well p ˆ predicts the person's actual identity i.
1 DeepFace's innovation comes from three distinct factors: (a) the 3D alignment, (b) a neural network structure with 120 million parameters, and
(c) training with 4.4 million labeled faces. Once the neural network is trained on this large set of faces, the final classification layer is removed and the output of the preceding fully connected layer is used as a low-dimensional face representation.

Often, face recognition applications seek a desirable low-dimensional representation that generalizes well to new faces that the neural network wasn't trained on. DeepFace's approach to this works, but the representation is a consequence of training a network for high-accuracy classification on their training data. The drawback of this approach is that the representation is difficult to use because the faces of the same person aren't necessarily clustered, which classification algorithms can take advantage of. FaceNet's triplet loss function is defined directly on the representation.
Figure 4 illustrates how FaceNet's training procedure learns to cluster face representations of the same person. The unit hypersphere is a high-dimensional sphere such that every point has distance

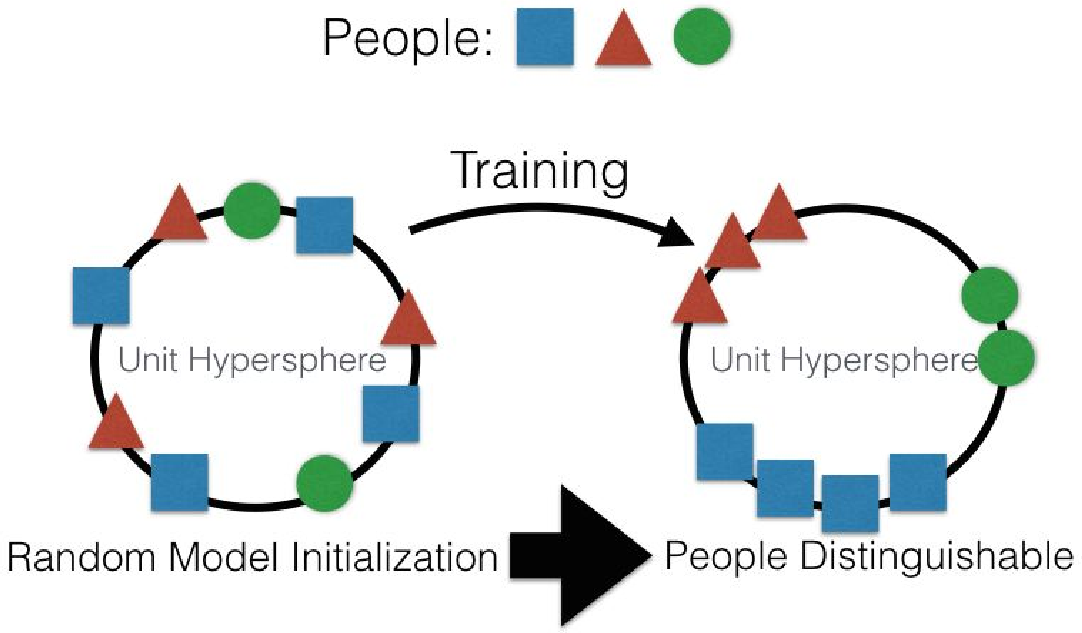


Figure [4]: Illustration of FaceNet's triplet-loss training procedure.

1 from the origin. Constraining the embedding to the unit hypersphere provides a structure to a space that is otherwise unbounded. FaceNet's innovation comes from four distinct factors:
(a) the triplet loss,(b) their triplet selection procedure,(c) training with 100 million to 200 million labeled images, and (d) (not discussed here) large-scale experimentation to find a network architecture. For reference.

- **Face Recognition in Mobile Computing**

The mobile computing community studies and improves off-the-shelf face recognition techniques in mobile scenarios. Due to lack of availability, these studies often use techniques with an order of magnitude less accuracy than the state-of-the-art. Soyata et al. study how to partition an Eigenfaces-based face recognition system between the mobile device, cloud, and cloud. Hsu et al. study the accuracies of cloud-based face recognition services as a drone's distance and angle to a person is varied. There has also been a rise of efficient GPU architectures for mobile devices, such as NVIDIA's Jetson TK1.

These studies and research directions are complementary to our studies in this paper. We do not study the impacts of executing techniques on different architectures, such as on an embedded GPU or offloaded to a surrogate. We instead present performance experiments showing that OpenFace's execution time is well-suited for mobile scenarios compared to other techniques.

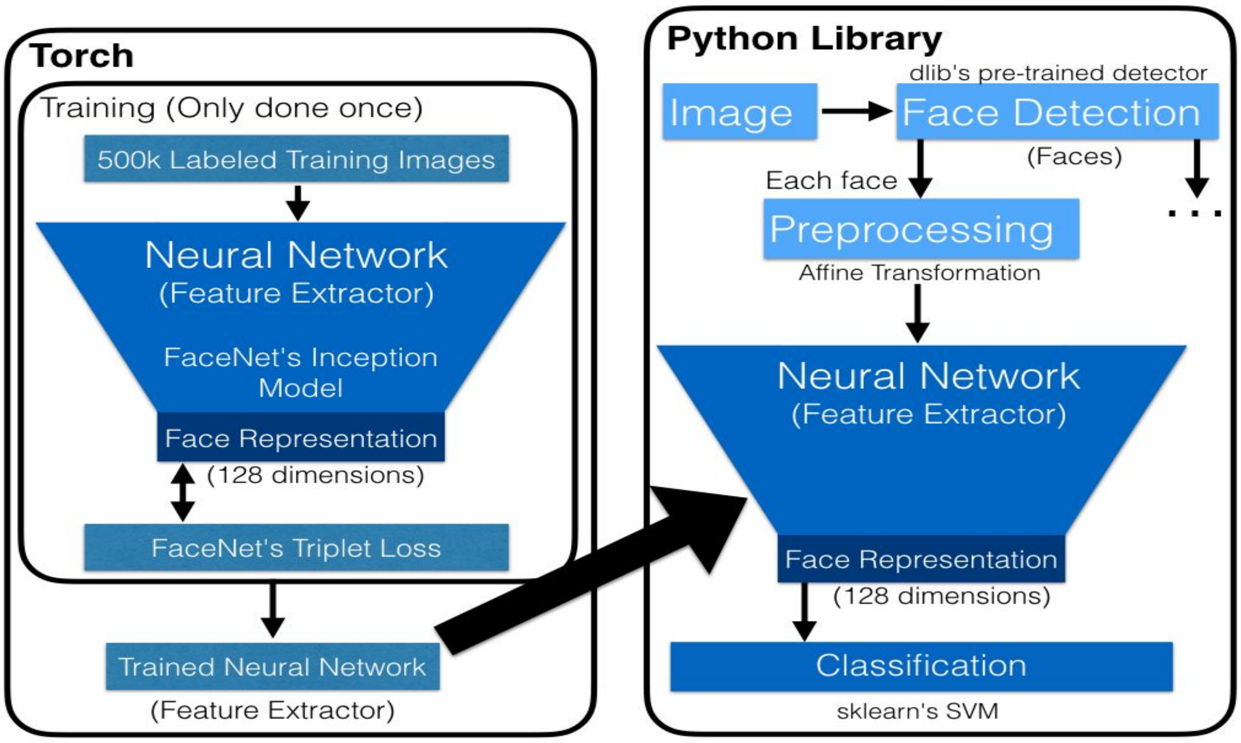
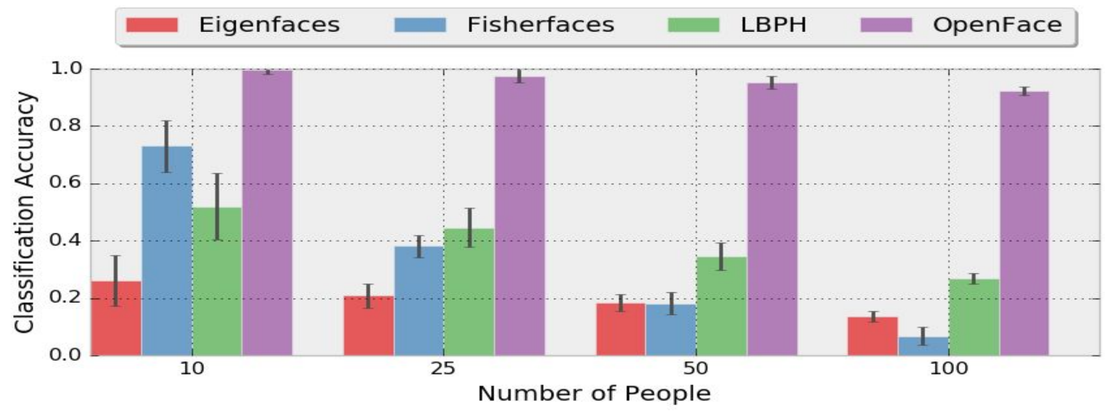
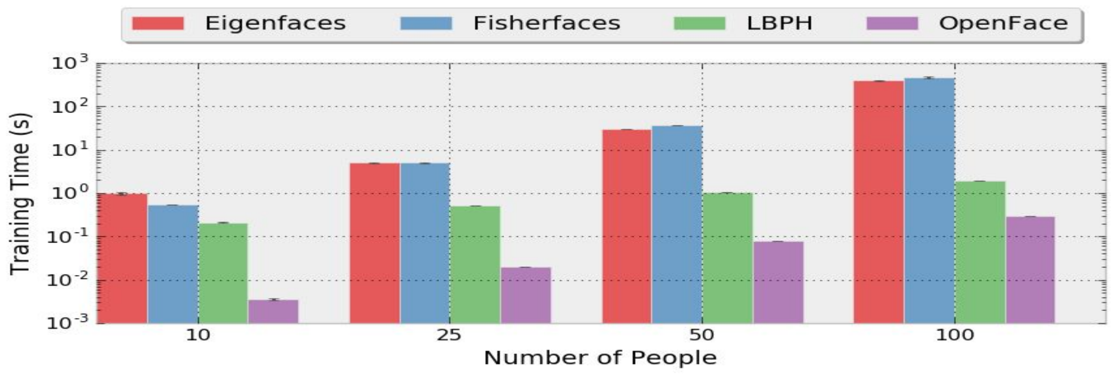


Figure [5]: OpenFace's project structure.

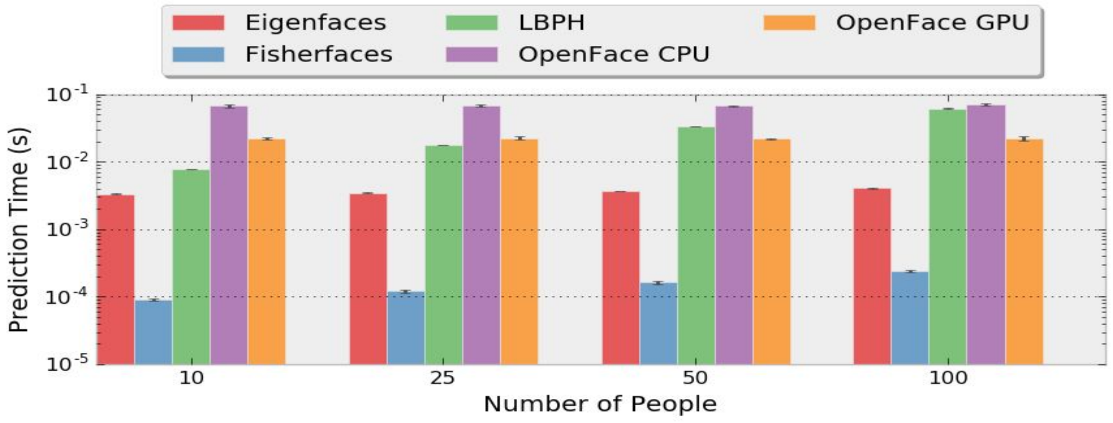
- **Conclusion**



(a) Classification accuracy



(b) Training times.



(c) Per-image prediction times.

Figure [6]: Accuracy and performance comparisons between OpenFace and prior non-proprietary face recognition implementations (from OpenCV).

3.1 Analysis and limitations of existing system

Systems are created to solve problems. One can think of the systems approach as an organized way of dealing with a problem . In this dynamic world , the subject system analysis and design , mainly deals with the software development activities.
Next is to define a system , explain the different phases of system development life cycle , enumerate the components of system analysis and explain the components of system designing.

3.2 Need for new system

The natural use of face recognition technology is the replacement of PIN

- **Government use:**

  1-security /counterterrorism :access control ,comparing surveillance images to know terrorist.

  Immigration: rapid progression through customer.

  Voter verification :where eligible politicians are required to verify their identity during the voting process.

- **Commercial use:**

  Residential security : alert homeowners of approaching personnel .

  Banking using ATM: the software is able to quickly verify a customer’s face.

  Physical access control of buildings , areas , doors , cars or not access.

3.3 Analysis of new system

3.3.1 User Requirements

A face detection and recognition system given an image, the goal of the face detection system is to determine whether or not there are any faces in the image and, if present, return the face location and the name . The system should operate at real-time to make for a passive and fully automatic Access Control system. A user would only be required to stand in front of the camera in order to be recognized. The main motivation for a face detection system is that the user wouldn’t be required to position his/her face into a fixed size box in order to be recognized by the face recognition system.

Face localization : seeks to determine the position of face within an image; the detection problem is simplified since the input image contains face or more than one.

Facial feature detection: seeks to detect the presence and location of features, such as the mouth, nose, eyes, lips, ears, etc.; the detection problem is simplified since the input image contain only on face or more than one .

Facial expression recognition :identifies the emotional states of humans, e.g. happy, sad, anger.

3.3.2 System Requirements

The system needs a database that stores images in order to identify the people who have already been stored in the database . And then analyze and identify the person and know his condition in terms of whether happy or sad or otherwise, in the case that the person has opened the camera and take a picture, the program works on the search in the database, if the image was found in advance, sends a message in his passion to the user , but if it is not already found in the database, the program works to save the image and analyze it in terms of knowledge of the emotional state.

3.3.3 Functional Requirements

- Input: Data set of the image of the face and object.
- Processing : Check the training and the test. Compare the current image by those in the database.
- Output : The information is saved in the database then the detection and recognition are ready.

3.3.4 Non- Functional Requirements

1- Security:

The database must not be accessed or altered from any other system except the system itself because it contains the data of only the authorized people .

2- Performance:

- Response time:

  The system shall not take more than 5 seconds to authenticate the user.
- Load time:

  The system will take from 2-8 seconds run.

3- Usability:

The customer will deal with our system through the graphical user interface(GUI); he/she shall not access the code itself.

4- preparing for server failure or crashes:

Database server:

if the server faced any failures or sudden shut down, the system may have an emergency restart , if the database server crashed,a backup server should take the lead .

3.4 Advantages of new system

Advantages:

a- There are many benefits to face recognition system such as its convenience and social acceptability . All you need is your picture taken for it to work

b- Face recognition is easy to use and in many cases it can be performed without a person even knowing .

c- Face recognition is also one of the most inexpensive biometrics in the market and its price should continue to go down.

3.5 User characteristics

This system is used to recognize people from their faces so the user will interact with it easily all the needed is to enter the name and the ID then take an image to access the system so general users with basic computer skills can use this software .

3.6 Planning

3.6.1 Feasibility study and estimated cost

- **<u>Feasibility Study</u>**
  - The way to recognize the face relative to the data saved
  - Process of recognition of the face in different lighting
  - The way to recognize the face if taken from the side
  - How to identify people who resemble each other

- **<u>Technical Feasibility</u>**
  - We have the technology to execute the project.
  - We have the technical expertise and time to solve problems.
  - We must analyze and identify common errors in order to implement the project correctly.

Economic Feasibility and Benefits:

- too costly to implement.
- There are no mistakes in the recognition and detect emotion
- The benefits of this project are most important to avoid all the wrong detect emotions , no more error
- Provide time to correct errors in recognition and detect emotions (such as happy , sad , angry, etc.)

3.6.2 Gantt chart

Task/Week of	1/10	8/10	16/10	23/10	30/10	7/11	14/11	21/11	28/11	4/12	29/6
Decide Topic of interest	■	■									
Write survey and prepare a presentation			■	■	■						
Decide a specific project idea				■	■	■					
Set up programming environment							■				
Create initial dataset							■	■			
Importing Methods needs									■		
Run algorithms against test data									■		
UMI Diagram										■	■
Write a final paper and prepare a presentation											■

table 1:Gantt chart

4.1 Design and Implementation Constraints

The analysis of the feelings of wrinkles or facial features such as the smile caused by the lips or the sadness that appears on the eyebrows and eyes is challenging and has some limitations due to similarities in some emotions. This task effectively requires strategies that collect facial analysis content with prior knowledge and use more than a package of The images stored in the database are preset, the emotion analysis tools can identify and analyze many faces automatically and quickly.But computer programs have problems recognizing things like irony, irony, threats, jokes, exaggerations - the kinds of things a person can not recognize. Failure to recognize this can spoil the results.

A person suffering from "disappointment" may be classified as negative for emotional analysis, but within the phrase "I am not disappointed", it should be classified as positive. We find it easy to admit that irony is the phrase "ridiculous grin with pride", while the instrument of analysis of automatic emotion may not be, and is likely to classify it as an example of a positive feeling. Using images, for example, such as those found on Twitter especially, sometimes on Facebook, there may not be enough context to analyze trusted emotions. However, in general, face recognition emotion has a good reputation as a good source of emotion analysis, and as the number of images in the database increases, it is likely to become more useful. Therefore, the tools of analyzing the emotions of the mechanism do a wonderful job already in the analysis of the face, The result of emotion analysis should be accurate and fast. To make it more accurate, we must use more than one algorithm, but the process often takes longer and becomes slower

4.2 Risks and risk management

Facial recognition can be a useful tool for governments, companies and consumers, but it also comes with risks, especially to individuals. The latter includes:

- **The lack of permission**

  Facial recognition data can easily be collected in public places – all the software would need is a clear image of the subject’s face.

- **Predatory marketing**

  Software which analyses facial expressions could potentially be put to use by some companies to prey on vulnerable customers. This could be done by segmenting extreme emotions – such as distress – and tailoring their products and services to these individuals.

- **Disadvantage when applying for jobs**

  Job applicants who don’t want to give potential employers access to details of their personal lives can keep these private, such as by selecting the related privacy settings on social media. However, facial recognition could potentially allow recruiters to find out more about you than you’d realise.
- **Stalking**

  Tools like reverse image searches can provide stalkers with more data about their victims.
- **Identity fraud**

  Criminals who have collected enough personal information on you could commit identity fraud[4]. This could have a significant effect on your personal life, including on your finances.

How to protect yourself against the risks?

Used responsibly, facial recognition can be a useful tool. However, it's still practical to try to protect yourself from the potential risks involved. If you’re offered the option for your image to be captured for facial recognition purposes, stop and consider whether it’s worth it. Unfortunately, some practices don’t involve asking individuals for permission to collect facial data. This means that it’s worth being vigilant and keeping an eye out for any signs that your identity may have been stolen[5]. You may also want to utilise your Equifax Credit Report & Score [6] – free for the first 30 days then £7.95 monthly, it includes WebDetect, which will alert you if it finds that your financial details are being shared on the internet by fraudsters.

4.3 Design of database ERD

4.3.1 Entity Relationship Diagram

An entity relationship diagram(ERD) is a graphical representation of an information system that shows the relationship between people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and can be used as the foundation for relational database.

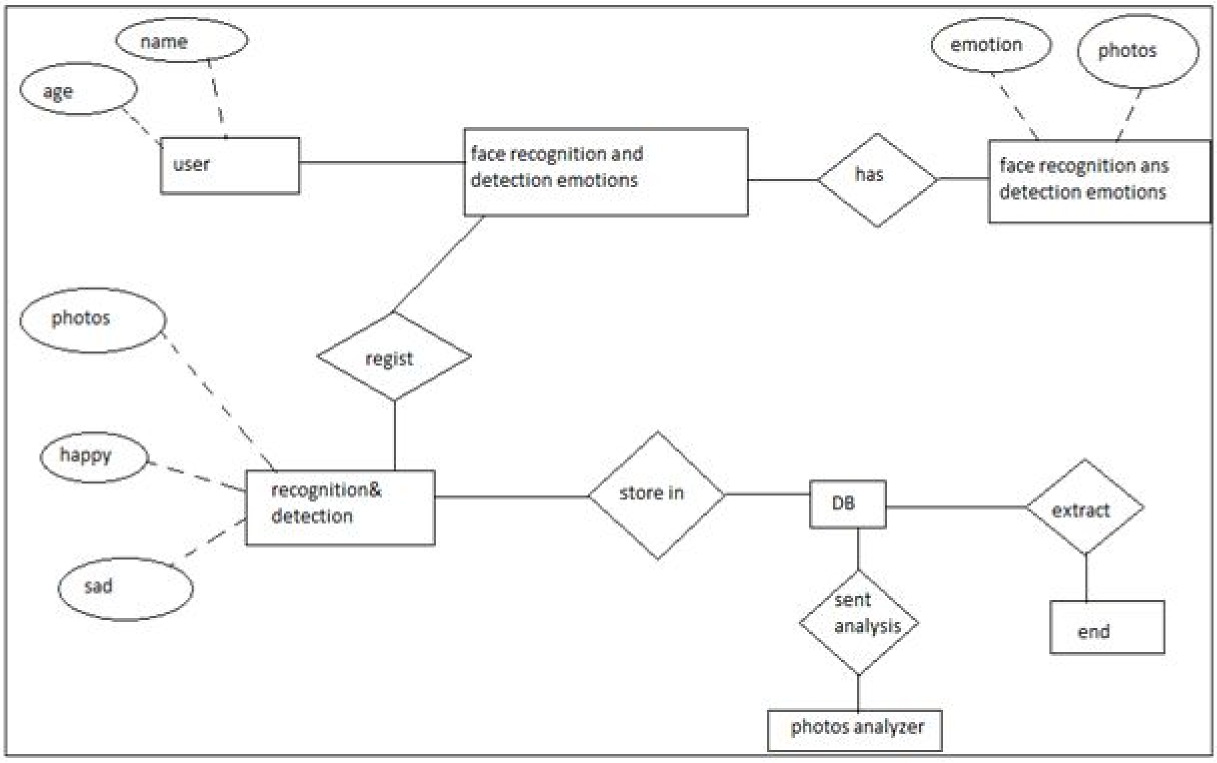


Figure [8] : ERD Diagram.

4.4 Class diagram

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application. The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages. The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a structural diagram. The purpose of the class diagram is to model the static view of an application. The class diagrams are the only diagrams which can be directly mapped with object oriented languages and thus widely used at the time of construction. The UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application but class diagram is a bit different. So it is the most popular UML diagram in the coder community.

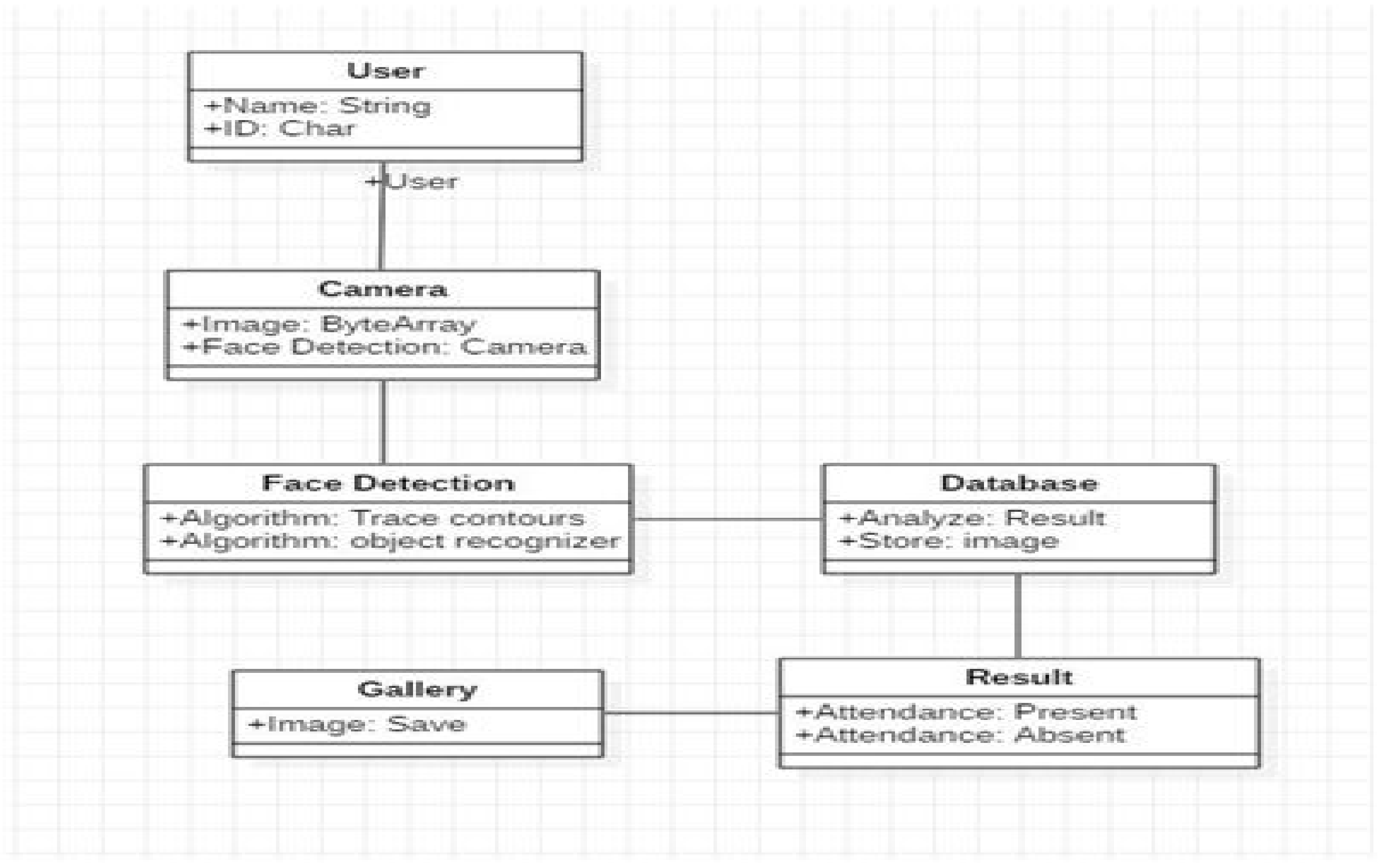


Figure [9]:Class Diagram.

4.5 Use case diagram

Use case diagrams are considered for high level requirement analysis of a system. So when the re- quirements of a system are analyzed the functionalities are captured in use cases. This diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. Use case diagrams are drawn to capture the functional requirements of a system.

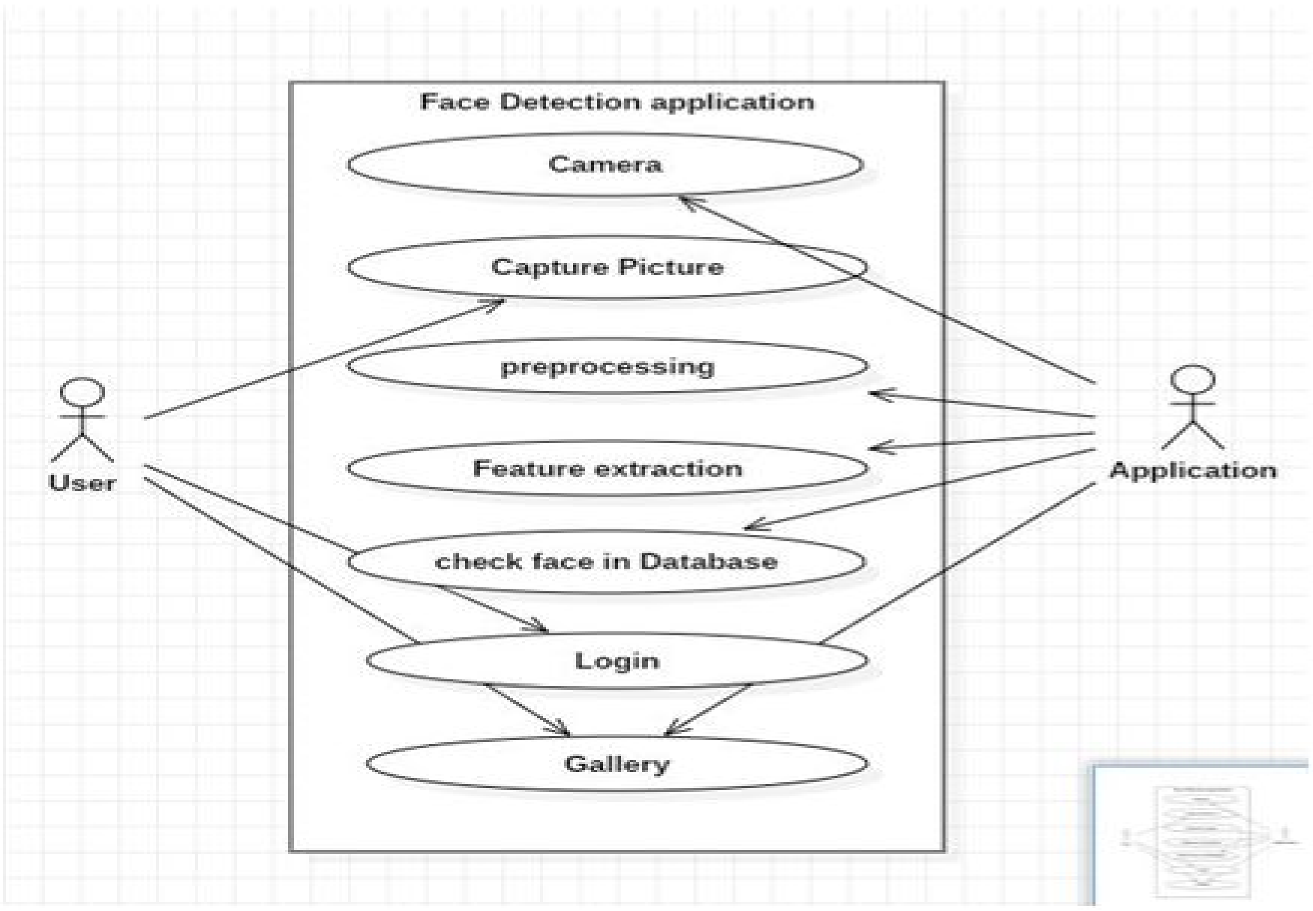


Figure [10]: Use Case Diagram.

4.6 Activity diagram

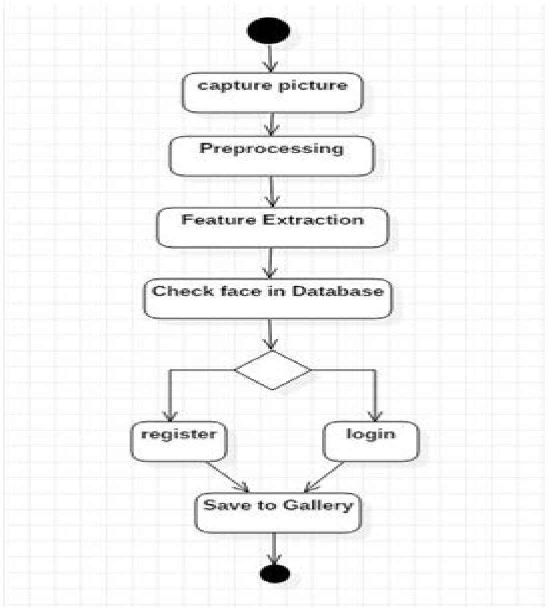
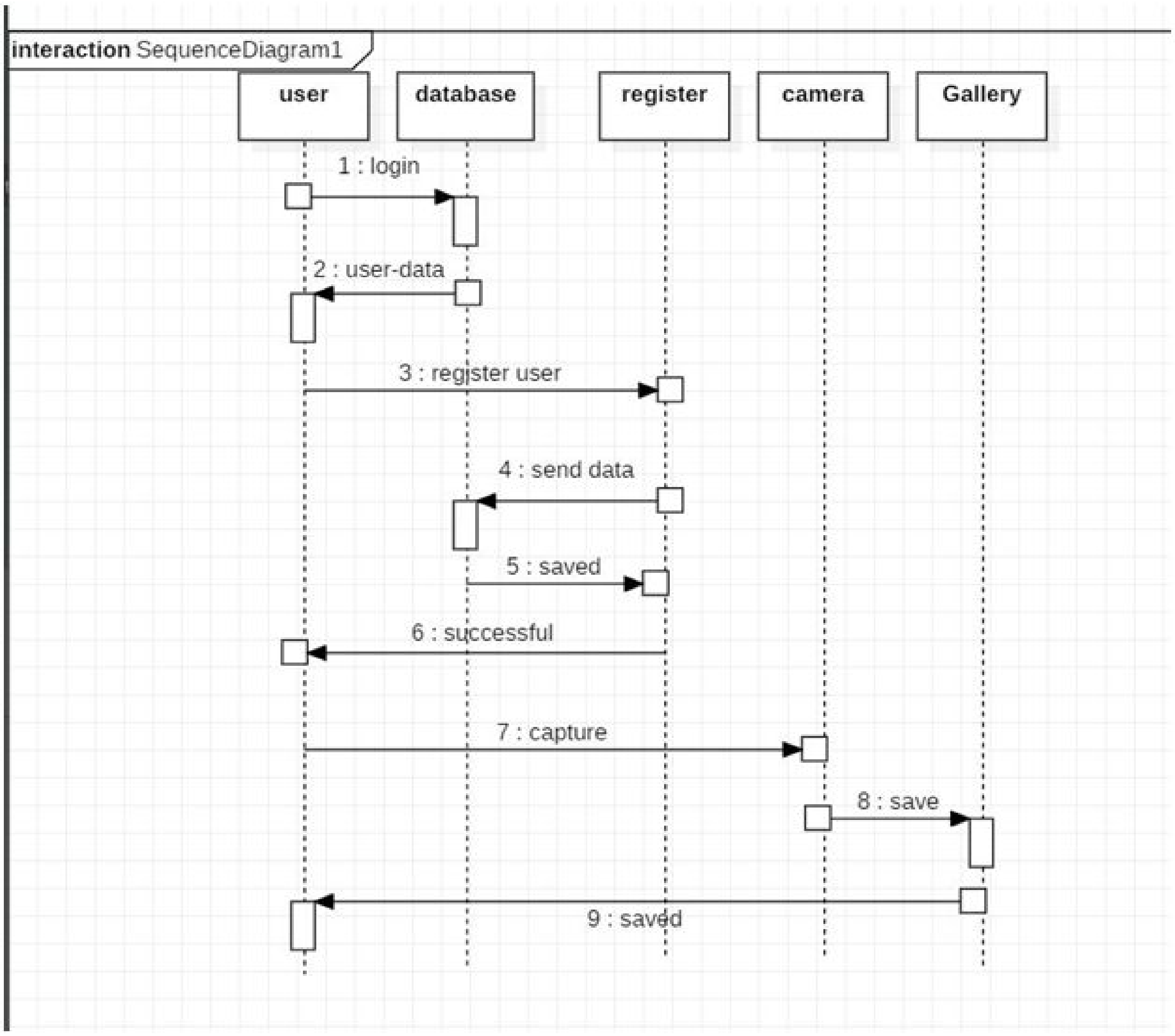


Figure [11]: Activity Diagram.

4.7 Sequence diagram

UML sequence diagrams are used to show how objects interact in a given situation. An important char- acteristic of a sequence diagram is that time passes from top to bottom : the interaction starts near the top of the diagram and ends at the bottom (i.e. Lower equals Later). A popular use for them is to document the dynamics in an object-oriented system. For each key collabo- ration, diagrams are created that show how objects interact in various representative scenarios for that collaboration. The sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur. Much like the class diagram, developers typically think sequence diagrams were meant exclusively for them. However, an organization's business staff can find sequence diagrams useful to communicate how the business currently works by showing how various business objects in- teract. Besides documenting an organization's current affairs, a business-level sequence diagram can be used as a requirements document to communicate requirements for a future system implementation. During the requirements phase of a project, analysts can take use cases to the next level by providing a more formal level of refinement. When that occurs, use cases are often refined into one or more se- quence diagrams.In the sequence diagram presented, the camera will capture images for the users. The camera is always alive to capture users face images` at any time. Once the camera captured an image, it will be sent to the system for processing that follows all steps in the system overview. At first there is a preprocessing step using Retinex algorithm that adjust image brightness. Then skin pixels are detected from the images and based on skin face regions are extracted. Then if there is any region that is likely to be a face is detected, it will be sent to the feature extraction step for further processing. If there are features in the regions, then further processing by enhancing de- tected features and measuring distances will take place. If no feature extracted, then it will go back to select another region.



Figure[12]: UML Sequence Diagram.

4.8 State diagram

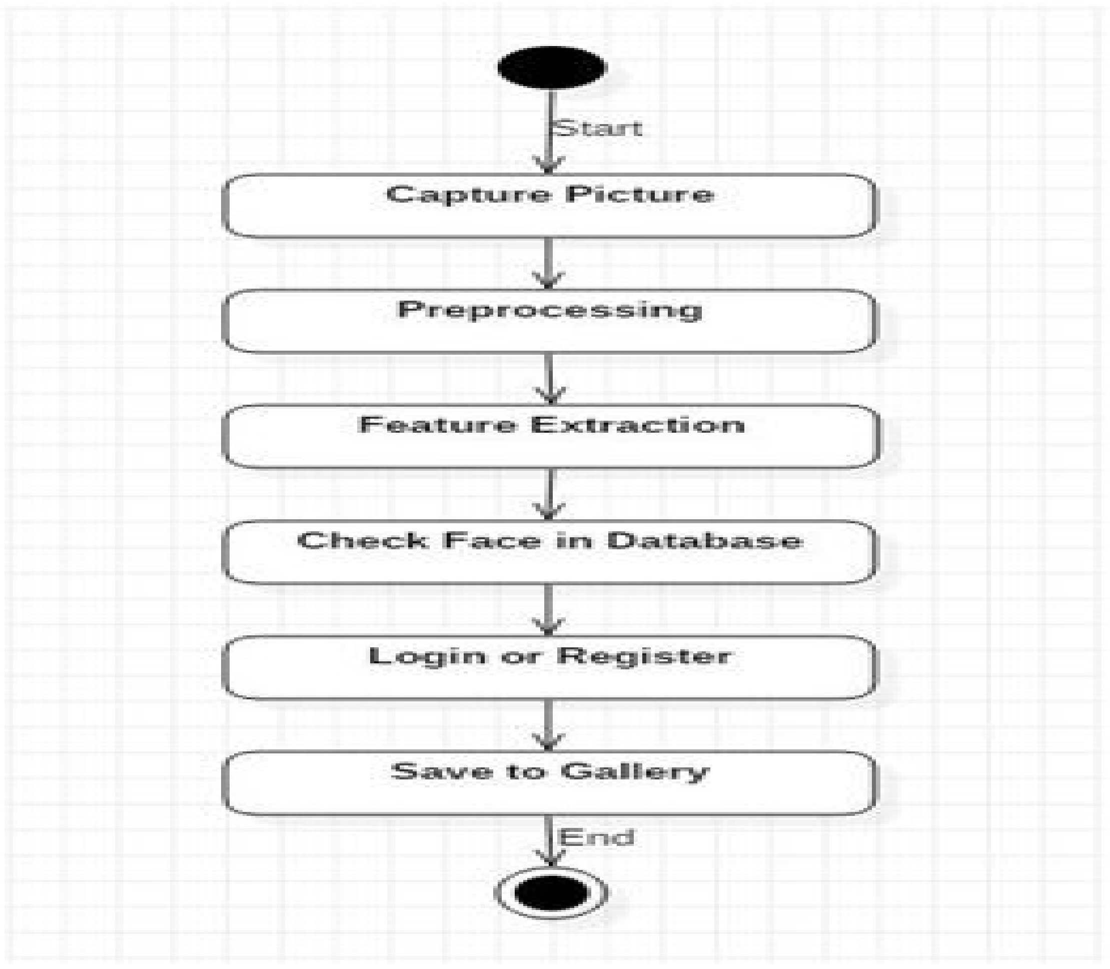


Figure [13]: State Diagram

ALGORITHMS AND CODE

Algorithms – For the algorithms were carried out an extended research to select the proper algorithms that are best suited for the development of the project.Demonstration of Facial Emotion Recognition on Real Time Video Using CNN : Python & Keras

Software – In this section we will point out the parts that was researched and studied to be implemented, adjusted and optimized in the real world' conditions to our project.

Main Form

Login Form

Register Form

Tracking Mode - Face Detection

Tracking Mode - Face Emotion

FACE DETECTION

Face detection is a technique that identifies or locates human faces in digital images. A typical example of face detection occurs when we take photographs through our smartphones, and it instantly detects faces in the picture. Face detection is different from Face recognition. Face detection detects merely the presence of faces in an image while facial recognition involves identifying whose face it is. In this article, we shall only be dealing with the former.Face detection is performed by using classifiers. A classifier is essentially an algorithm that decides whether a given image is positive(face) or negative(not a face). A classifier needs to be trained on thousands of images with and without faces. Fortunately, OpenCV already has two pre-trained face detection classifiers, which can readily be used in a program. The two classifiers are: Haar Classifier and Local Binary Pattern(LBP) classifier.In this article, however, we will only discuss the Haar Classifier.

Haar feature-based cascade classifiers

Haar-like features are digital image features used in object recognition. They owe their name to their intuitive similarity with Haar wavelets and were used in the first real-time face detector. Paul Viola and Michael Jones in their paper titled "Rapid Object Detection using a Boosted Cascade of Simple Features" used the idea of Haar-feature classifier based on the Haar wavelets. This classifier is widely used for tasks like face detection in computer vision industry.

Haar cascade classifier employs a machine learning approach for visual object detection which is capable of processing images extremely rapidly and achieving high detection rates. This can be attributed to three main reasons:

- Haar classifier employs 'Integral Image' concept which allows the features used by the detector to be computed very quickly.
- The learning algorithm is based on AdaBoost. It selects a small number of important features from a large set and gives highly efficient classifiers.
- More complex classifiers are combined to form a 'cascade' which discard any non-face regions in an image, thereby spending more computation on promising object-like regions.

Let us now try and understand how the algorithm works on images in steps:

1. 'Haar features' extraction

After the tremendous amount of training data (in the form of images) is fed into the system, the classifier begins by extracting Haar features from each image. Haar Features are kind of convolution kernels which primarily detect whether a suitable feature is present on an image or not. Some examples of Haar features are mentioned below:

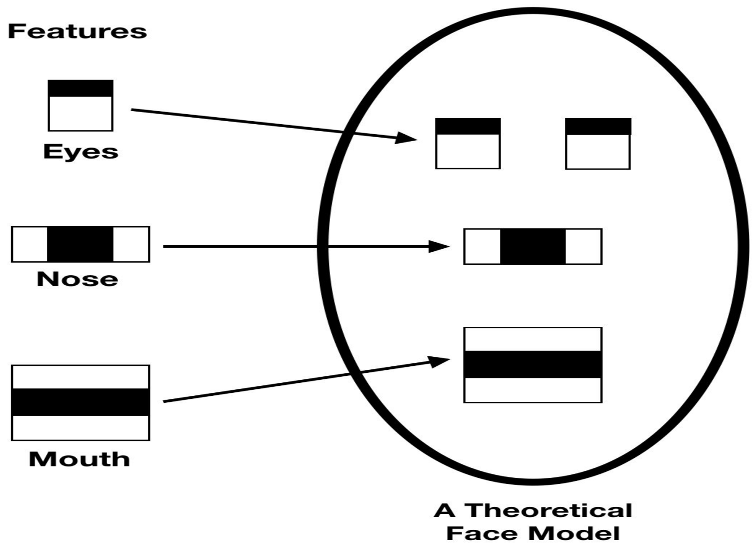


Figure [14]: Haar features extraction

These Haar Features are like windows and are placed upon images to compute a single feature. The feature is essentially a single value obtained by subtracting the sum of the pixels under the white region and that under the black

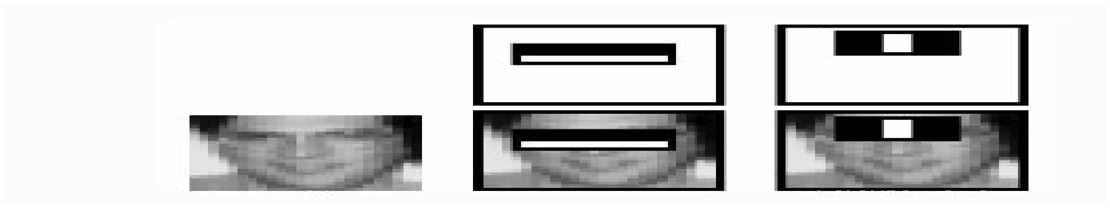
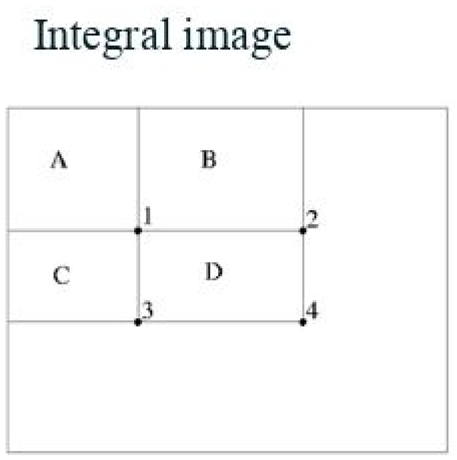
.


Figure [15]: Haar Features are like windows

For demonstration purposes, let's say we are only extracting two features, hence we have only two windows here. The first feature relies on the point that the eye region is darker than the adjacent cheeks and nose region. The second feature focuses on the fact that eyes are kind of darker as compared to the bridge of the nose. Thus, when the feature window moves over the eyes, it will calculate a single value. This value will then be compared to some threshold and if it passes that it will conclude that there is an edge here or some positive feature.

2. 'Integral Images' concept

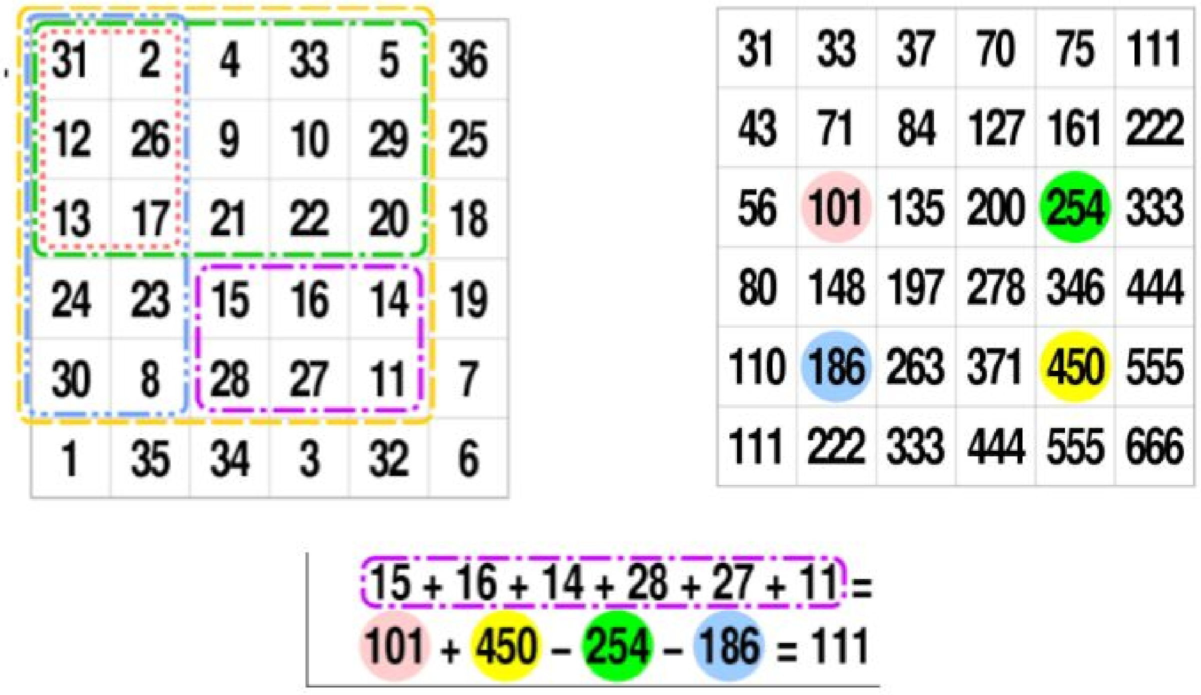
The algorithm proposed by Viola Jones uses a 24X24 base window size, and that would result in more than 180,000 features being calculated in this window. Imagine calculating the pixel difference for all the features? The solution devised for this computationally intensive process is to go for the Integral Image concept. The integral image means that to find the sum of all pixels under any rectangle, we simply need the four corner values.



Sum of all pixels in
D = 1+4-(2+3)
= A+(A+B+C+D)-(A+C+A+B)
= D

Figure [16]: Integral images.

This means, to calculate the sum of pixels in any feature window, we do not need to sum them up individually. All we need is to calculate the integral image using the 4 corner values. The example below will make the process transparent.



31	2	4	33	5	36
12	26	9	10	29	25
13	17	21	22	20	18
24	23	15	16	14	19
30	8	28	27	11	7
1	35	34	3	32	6

31	33	37	70	75	111
43	71	84	127	161	222
56	101	135	200	254	333
80	148	197	278	346	444
110	186	263	371	450	555
111	222	333	444	555	666

$$15 + 16 + 14 + 28 + 27 + 11 =$$
$$101 + 450 - 254 - 186 = 111$$

Figure [17]: process transparent.

3. 'Adaboost' : to improve classifier accuracy

As pointed out above, more than 180,000 features values result within a 24X24 window. However, not all features are useful for identifying a face. To only select the best features out of the entire chunk, a machine learning algorithm called Adaboost is used. What it essentially does is that it selects only those features that help to improve the classifier accuracy. It does so by constructing a strong classifier which is a linear combination of a number of weak classifiers. This reduces the amount of features drastically to around 6000 from around 180,000.

4. Using 'Cascade of Classifiers'

Another way by which Viola Jones ensured that the algorithm performs fast is by employing a cascade of classifiers. The cascade classifier essentially consists of stages where each stage consists of a strong classifier. This is beneficial since it eliminates the need to apply all features at once on a window. Rather, it groups the features into separate sub-windows and the classifier at each stage determines whether or not the sub-window is a face. In case it is not, the sub-window is discarded along with the features in that window. If the sub-window moves past the classifier, it continues to the next stage where the second stage of features is applied. The process can be understood with the help of the diagram below.

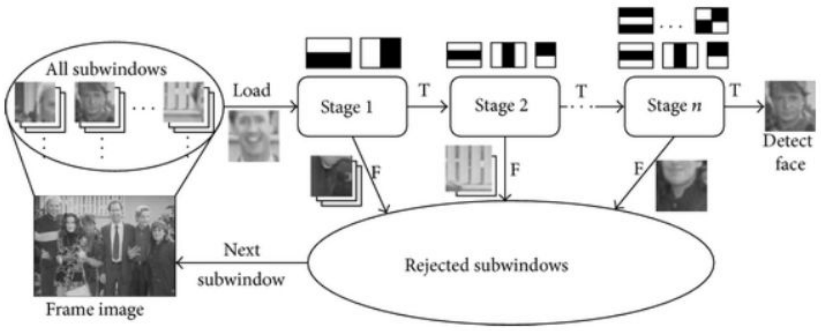


Figure [18]: cascade of classifiers structure.

code:

```
# Loading the cascades
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
smile_cascade = cv2.CascadeClassifier('haarcascade_smile.xml')
```

Face Detection with OpenCV-Python

Now we have a fair idea about the intuition and the process behind Face recognition. Let us now use OpenCV library to detect faces in an image.

Load the necessary Libraries

```
from keras.preprocessing.image import img_to_array
from keras.models import load_model
```

```
import imutils
import cv2
import numpy as np
import sys
```

Face detection

We shall be using the detectMultiscale module of the classifier. This function will return a rectangle with coordinates(x,y,w,h) around the detected face. This function has two important parameters which have to be tuned according to the data.

- scalefactor in a group photo, there may be some faces which are near the camera than others. Naturally, such faces would appear more prominent than the ones behind. This factor compensates for that.
- minNeighbors this parameter specifies the number of neighbors a rectangle should be called a face.

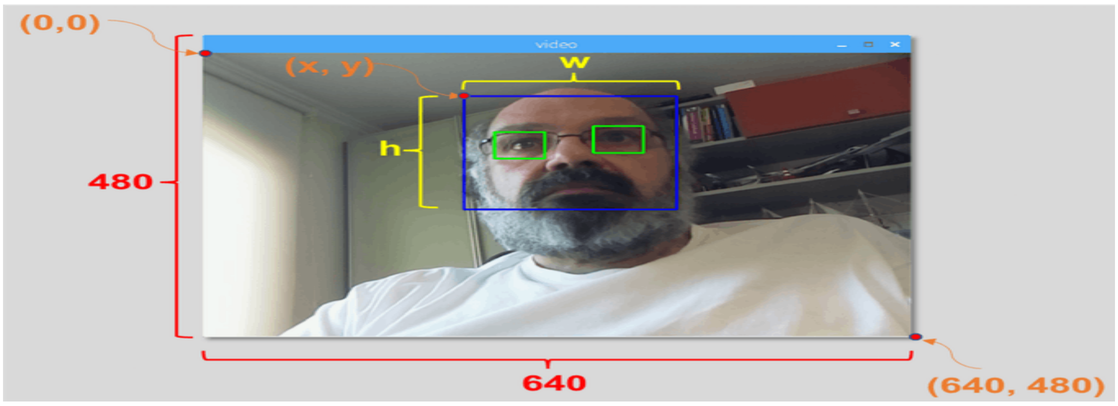


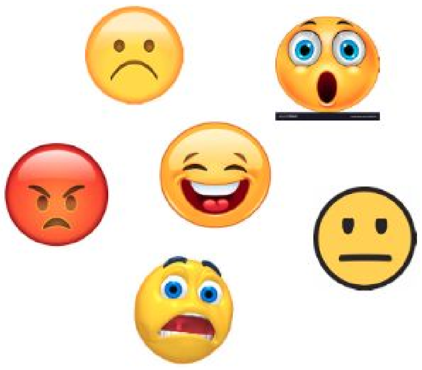
Figure [19]: locations of detect face.

code:

```
# Defining a function that will do the detections
def detect(gray, frame):
# Applying the haar classifier to detect faces
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = frame[y:y+h, x:x+w]
        eyes = eye_cascade.detectMultiScale(roi_gray, 1.1, 22)
        for (ex, ey, ew, eh) in eyes:
            cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)
        smiles = smile_cascade.detectMultiScale(roi_gray, 1.7, 22)
        for (sx, sy, sw, sh) in smiles:
            cv2.rectangle(roi_color, (sx, sy), (sx+sw, sy+sh), (0, 0, 255), 2)
    return frame
```

Face Emotion

A face emotion recognition system comprises of two step process i.e. face detection (bounded face) in image followed by emotion detection on the detected bounded face. The following two techniques are used for respective mentioned tasks in face recognition system.



(1) Haar feature-based cascade classifiers : It detects frontal faces in an image well. It is real time and faster in comparison to other face detector. This blog-post uses an implementation from Open-CV.

(2) Xception CNN Model (Mini_Xception, 2017) : We will train a classification CNN model architecture which takes bounded face (48*48 pixels) as input and predicts probabilities of 7 emotions in the output layer.

Data-set

We can download the facial expression recognition (FER) data-set from Kaggle challenge . The data consists of 48×48 pixel gray scale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

The training set consists of 35,888 examples. train.csv contains two columns, “emotion” and “pixels”. The “emotion” column contains a numeric code ranging from 0 to 6, inclusive, for the emotion that is present in the image. The “pixels” column contains a string surrounded in quotes for each image. The contents of this string a space-separated pixel values in row major order

Loading FER Data-set

The below code loads the data-set and pre-process the images for feeding it to CNN model. There are two definitions in the code snippet here:

1. def load_fer2013 : It reads the csv file and convert pixel sequence of each row in image of dimension 48*48. It returns faces and emotion labels.

2. def preprocess_input: It is a standard way to pre-process images by scaling them between -1 to 1. Images are scaled to [0,1] by dividing it by 255. Further, subtraction by 0.5 and multiplication by 2 changes the range to [-1,1]. [-1,1] has been found a better range for neural network models in computer vision problems.

code:

```
import pandas as pd
import cv2
import numpy as np

dataset_path = 'fer2013/fer2013.csv'
image_size=(48,48)

def load_fer2013():
    data = pd.read_csv(dataset_path)
    pixels = data['pixels'].tolist()
    width, height = 48, 48
    faces = []
    for pixel_sequence in pixels:
        face = [int(pixel) for pixel in pixel_sequence.split(' ')]
        face = np.asarray(face).reshape(width, height)
        face = cv2.resize(face.astype('uint8'),image_size)
        faces.append(face.astype('float32'))
    faces = np.asarray(faces)
    faces = np.expand_dims(faces, -1)
    emotions = pd.get_dummies(data['emotion']).as_matrix()
    return faces, emotions

def preprocess_input(x, v2=True):
    x = x.astype('float32')
    x = x / 255.0
    if v2:
        x = x - 0.5
        x = x * 2.0
    return x

faces, emotions = load_fer2013()
faces = preprocess_input(faces)
xtrain, xtest,ytrain,ytest = train_test_split(faces, emotions,test_size=0.2,shuffle=True)
```

5 expression samples of each of the 7 emotions in the data-set can be seen below.

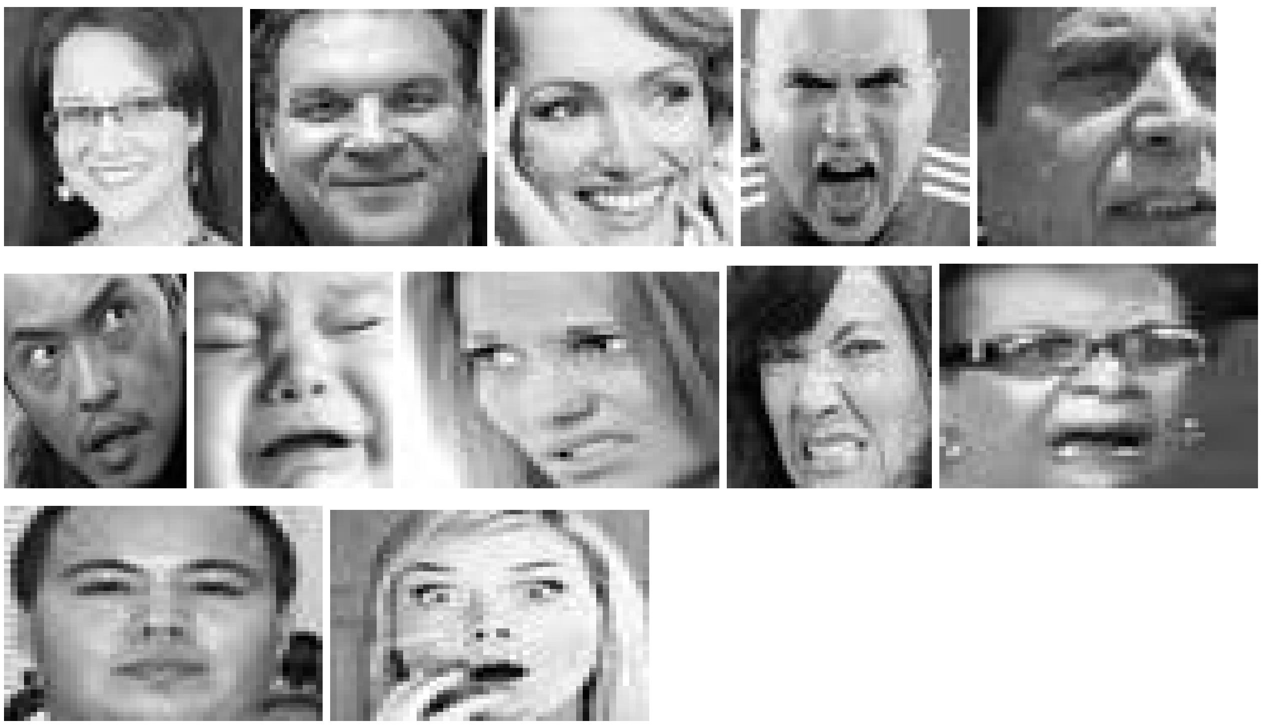


Figure [20]: different emotions.

Originally in the dataset provided in kaggle link, each image is given as string which is a row 1×2304 which is 48×48 image stored as row vector. The strings in the .csv files can be converted into images.

Training CNN model : Mini Xception

Here comes the exciting architecture which is comparatively small and achieves almost state-of-art performance of classifying emotion on this data-set. The below architecture was proposed by Octavio Arragia et al. in this paper [12].

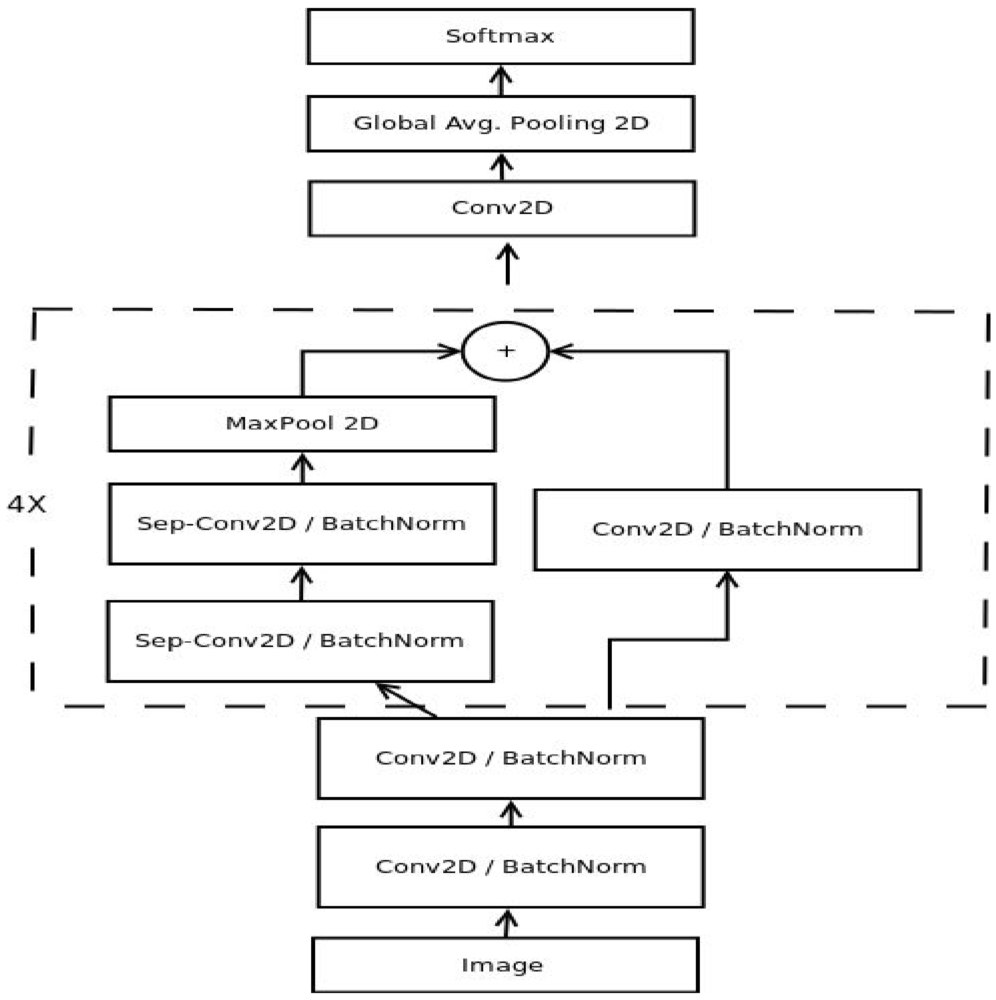


Figure [20]:**Proposed Mini_Xception architecture for emotion classification**

We can notice that the center block is repeated 4 times in the design. This architecture is different from the most common CNN architecture like one used in the blog-post here[13]. Common architectures uses fully connected layers at the end where most of parameters resides. Also, they use standard convolutions. Modern CNN architectures such as Xception leverage from the combination of two of the most successful experimental assumptions in CNNs: the use of residual modules and depth-wise separable convolutions.

There are various techniques that can be kept in mind while building a deep neural network and is applicable in most of the computer vision problems. Below are few of those techniques which are used while training the CNN model below.

1. **Data Augmentation** : More data is generated using the training set by applying transformations. It is required if the training set is not sufficient enough to learn representation. The image data is generated by transforming the actual training images by rotation, crop, shifts, shear, zoom, flip, reflection, normalization etc.
2. **Kernel_regularizer** : It allows to apply penalties on layer parameters during optimization. These penalties are incorporated in the loss function that the network optimizes. Argument in convolution layer is nothing but L2 regularisation of the weights. This penalizes peaky weights and makes sure that all the inputs are considered.
3. **Batch Normalization** : It normalizes the activation of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1. It addresses the problem of internal covariate shift. It also acts as a regularizer, in some cases eliminating the need for Dropout. It helps in speeding up the training process.
4. **Global Average Pooling** : It reduces each feature map into a scalar value by taking the average over all elements in the feature map. The average operation forces the network to extract global features from the input image.
5. **Depthwise Separable Convolution** : These convolutions are composed of two different layers: depth-wise convolutions and point-wise convolutions. Depth-wise separable convolutions reduces the computation with respect to the standard convolutions by reducing the number of parameters. A very nice and visual explanation of the difference between standard and depth-wise separable convolution is given in the paper.

Below python codes implements the above architecture in Keras.

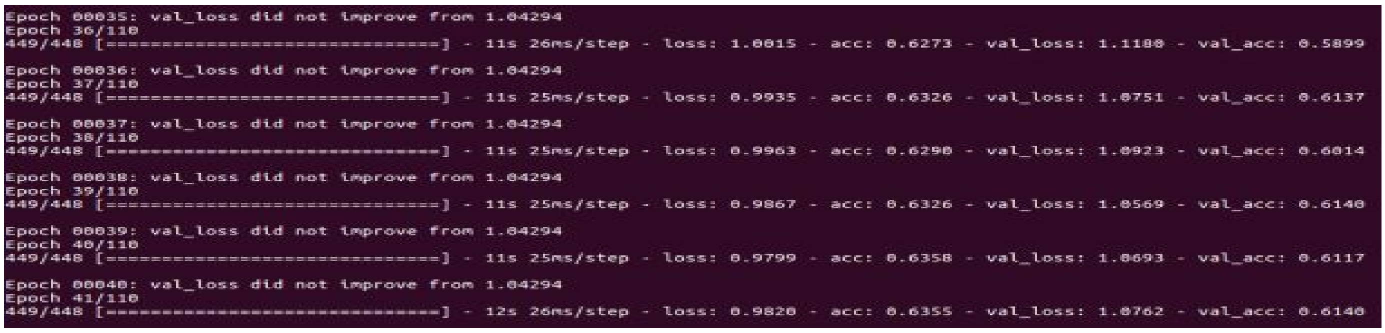
code:

```
from keras.callbacks import CSVLogger, ModelCheckpoint, EarlyStopping
from keras.callbacks import ReduceLROnPlateau
from keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from keras.layers import Activation, Convolution2D, Dropout, Conv2D
from keras.layers import AveragePooling2D, BatchNormalization
from keras.layers import GlobalAveragePooling2D
from keras.models import Sequential
from keras.layers import Flatten
from keras.models import Model
from keras.layers import Input
from keras.layers import MaxPooling2D
from keras.layers import SeparableConv2D
from keras import layers
from keras.regularizers import l2
import pandas as pd
import cv2
import numpy as np

# parameters
batch_size = 32
num_epochs = 110
input_shape = (48, 48, 1)
verbose = 1
num_classes = 7
patience = 50
base_path = 'models/'
l2_regularization=0.01

# data generator
data_generator = ImageDataGenerator(
                        featurewise_center=False,
                        featurewise_std_normalization=False,
                        rotation_range=10,
                        width_shift_range=0.1,
                        height_shift_range=0.1,
                        zoom_range=.1,
                        horizontal_flip=True)
# model parameters
regularization = l2(l2_regularization)
 # base
img_input = Input(input_shape)
x = Conv2D(8, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
use_bias=False)(img_input)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = Conv2D(8, (3, 3), strides=(1, 1), kernel_regularizer=regularization, use_bias=False)(x)
x = BatchNormalization()(x)
```

```
x = Activation('relu')(x)
 # module 1
residual = Conv2D(16, (1, 1), strides=(2, 2), padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)
x = SeparableConv2D(16, (3, 3), padding='same', kernel_regularizer=regularization,
use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(16, (3, 3), padding='same', kernel_regularizer=regularization,
use_bias=False)(x)
x = BatchNormalization()(x)
x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])
 # module 2
residual = Conv2D(32, (1, 1), strides=(2, 2), padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)
x = SeparableConv2D(32, (3, 3), padding='same', kernel_regularizer=regularization,
use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(32, (3, 3), padding='same', kernel_regularizer=regularization,
use_bias=False)(x)
x = BatchNormalization()(x)
x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])
 # module 3
residual = Conv2D(64, (1, 1), strides=(2, 2),padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)
x = SeparableConv2D(64, (3, 3),
padding='same',kernel_regularizer=regularization,use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = SeparableConv2D(64, (3, 3),
padding='same',kernel_regularizer=regularization,use_bias=False)(x)
x = BatchNormalization()(x)
x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])
 # module 4
residual = Conv2D(128, (1, 1), strides=(2, 2),padding='same', use_bias=False)(x)
residual = BatchNormalization()(residual)
x = SeparableConv2D(128, (3, 3),
padding='same',kernel_regularizer=regularization,use_bias=False)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
```

```
x = SeparableConv2D(128, (3, 3), padding='same',kernel_regularizer=regularization,use_bias=False)(x)
x = BatchNormalization()(x)
x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
x = layers.add([x, residual])
x = Conv2D(num_classes, (3, 3), padding='same')(x)
x = GlobalAveragePooling2D()(x)
output = Activation('softmax',name='predictions')(x)

model = Model(img_input, output)
model.compile(optimizer='adam', loss='categorical_crossentropy',metrics=['accuracy'])
model.summary()
 # callbacks
log_file_path = base_path + '_emotion_training.log'
csv_logger = CSVLogger(log_file_path, append=False)
early_stop = EarlyStopping('val_loss', patience=patience)
reduce_lr = ReduceLROnPlateau('val_loss', factor=0.1, patience=int(patience/4), verbose=1)
trained_models_path = base_path + '_mini_XCEPTION'
model_names = trained_models_path + '.{epoch:02d}-{val_acc:.2f}.hdf5'
model_checkpoint = ModelCheckpoint(model_names, 'val_loss', verbose=1,save_best_only=True)
callbacks = [model_checkpoint, csv_logger, early_stop, reduce_lr]
 model.fit_generator(data_generator.flow(xtrain, ytrain,batch_size),
                        steps_per_epoch=len(xtrain) / batch_size,
                        epochs=num_epochs, verbose=1, callbacks=callbacks,
                        validation_data=(xtest,ytest))# -*- coding: utf-8 -*-
```

The model gives 65-66% accuracy on validation set while training the model. The CNN model learns the representation features of emotions from the training images. Below are few epochs of training process with batch size of 64.



```
Epoch 00035: val_loss did not improve from 1.04294
Epoch 36/110
449/448 [==============================] - 11s 26ms/step - loss: 1.0015 - acc: 0.6273 - val_loss: 1.1180 - val_acc: 0.5899

Epoch 00036: val_loss did not improve from 1.04294
Epoch 37/110
449/448 [==============================] - 11s 25ms/step - loss: 0.9935 - acc: 0.6326 - val_loss: 1.0751 - val_acc: 0.6137

Epoch 00037: val_loss did not improve from 1.04294
Epoch 38/110
449/448 [==============================] - 11s 25ms/step - loss: 0.9963 - acc: 0.6290 - val_loss: 1.0923 - val_acc: 0.6014

Epoch 00038: val_loss did not improve from 1.04294
Epoch 39/110
449/448 [==============================] - 11s 25ms/step - loss: 0.9867 - acc: 0.6326 - val_loss: 1.0569 - val_acc: 0.6140

Epoch 00039: val_loss did not improve from 1.04294
Epoch 40/110
449/448 [==============================] - 11s 25ms/step - loss: 0.9799 - acc: 0.6358 - val_loss: 1.0693 - val_acc: 0.6117

Epoch 00040: val_loss did not improve from 1.04294
Epoch 41/110
449/448 [==============================] - 12s 26ms/step - loss: 0.9820 - acc: 0.6355 - val_loss: 1.0762 - val_acc: 0.6140
```

Figure [22]:**screenshot of accuracy.**

Testing the Model

While performing tests on the trained model, I felt that model detects the emotion of faces as neutral if the expressions are not made distinguishable enough. The model gives probabilities of each emotion class in the output layer of trained mini_xception CNN model. Below are the 18 facial expressions taken from google images to validate the trained model.

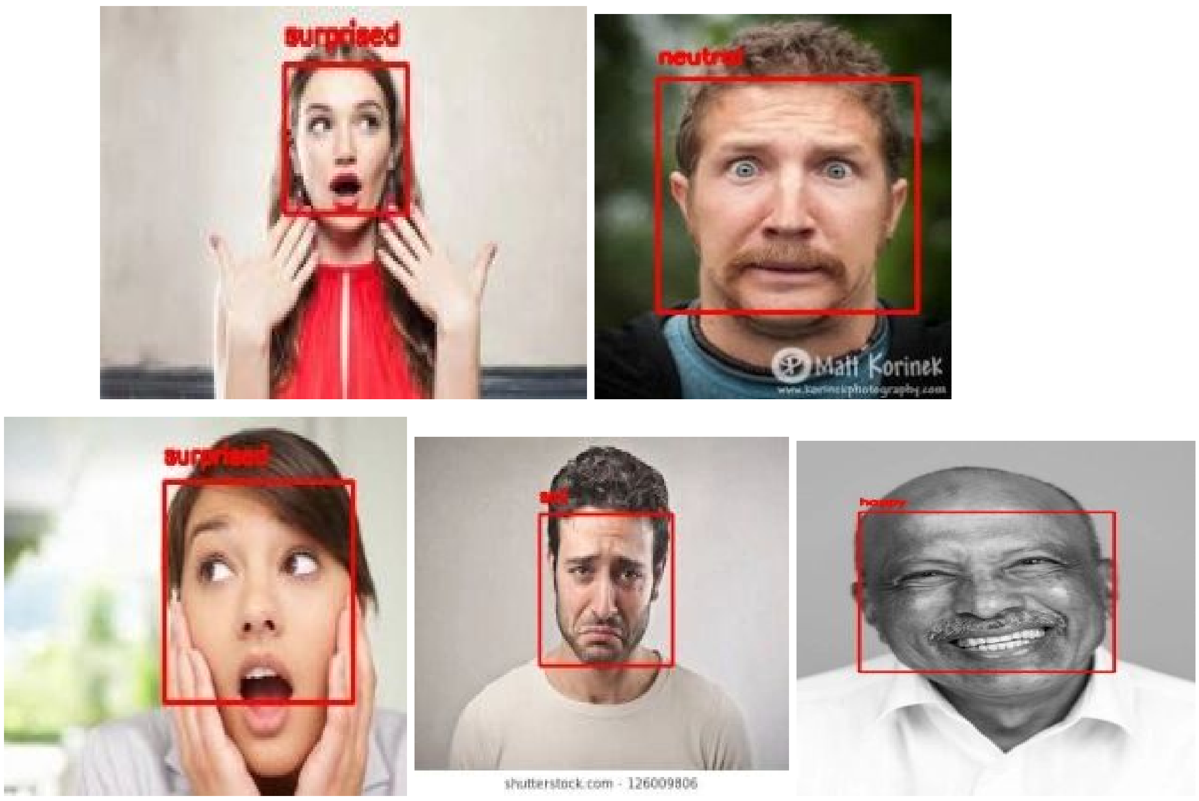


Figure [23]: some facial expressions .

In order to detect emotion in a single image, one can execute the python code below.

code:

```
from keras.preprocessing.image import img_to_array
from keras.models import load_model
import imutils
import cv2
import numpy as np
import sys

# parameters for loading data and images
detection_model_path = 'haarcascade_files/haarcascade_frontalface_default.xml'
```

```
emotion_model_path = 'models/_mini_XCEPTION.106-0.65.hdf5'
img_path = sys.argv[1]

# hyper-parameters for bounding boxes shape
# loading models
face_detection = cv2.CascadeClassifier(detection_model_path)
emotion_classifier = load_model(emotion_model_path, compile=False)
EMOTIONS = ["angry","disgust","scared", "happy", "sad", "surprised","neutral"]

#reading the frame
orig_frame = cv2.imread(img_path)
frame = cv2.imread(img_path,0)
faces                                                                          =
face_detection.detectMultiScale(frame,scaleFactor=1.1,minNeighbors=5,minSize=(30,30)
,flags=cv2.CASCADE_SCALE_IMAGE)

if len(faces) &amp;amp;amp;gt; 0:
    faces = sorted(faces, reverse=True,key=lambda x: (x[2] - x[0]) * (x[3] - x[1]))[0]
    (fX, fY, fW, fH) = faces
    roi = frame[fY:fY + fH, fX:fX + fW]
    roi = cv2.resize(roi, (48, 48))
    roi = roi.astype("float") / 255.0
    roi = img_to_array(roi)
    roi = np.expand_dims(roi, axis=0)
    preds = emotion_classifier.predict(roi)[0]
    emotion_probability = np.max(preds)
    label = EMOTIONS[preds.argmax()]
     cv2.putText(orig_frame, label, (fX, fY - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0,
0, 255), 2)
    cv2.rectangle(orig_frame, (fX, fY), (fX + fW, fY + fH),(0, 0, 255), 2)

cv2.imshow('test_face', orig_frame)
cv2.imwrite('test_output/'+img_path.split('/')[-1],orig_frame)
if (cv2.waitKey(2000) &amp;amp;amp;amp; 0xFF == ord('q')):
    sys.exit("Thanks")
cv2.destroyAllWindows()
```

User interface

Desktop Application

Home Page:

It contains a login button , register button and exit button so the user can choose any of them if he/she is registered before and want to access the system for the first time , he/she will press the register button.And this is the picture of the first page :

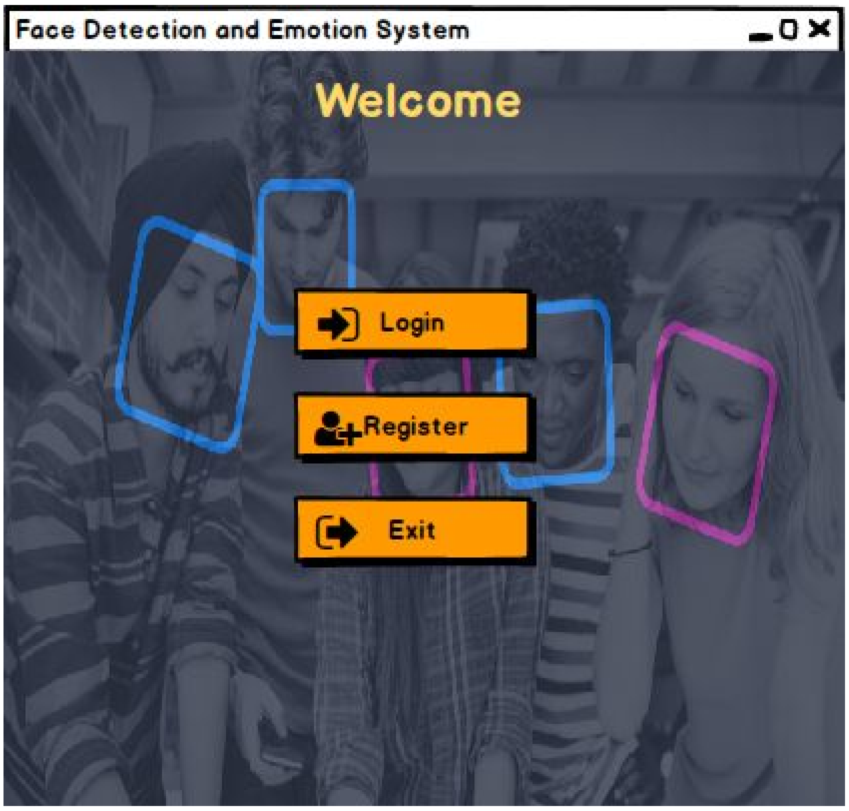


Figure [24]: Home page .

Register Page:

In this page we'll find two text fields in them the user can enter the user name and the password and back button to go back to the home page . The following image is the register page.

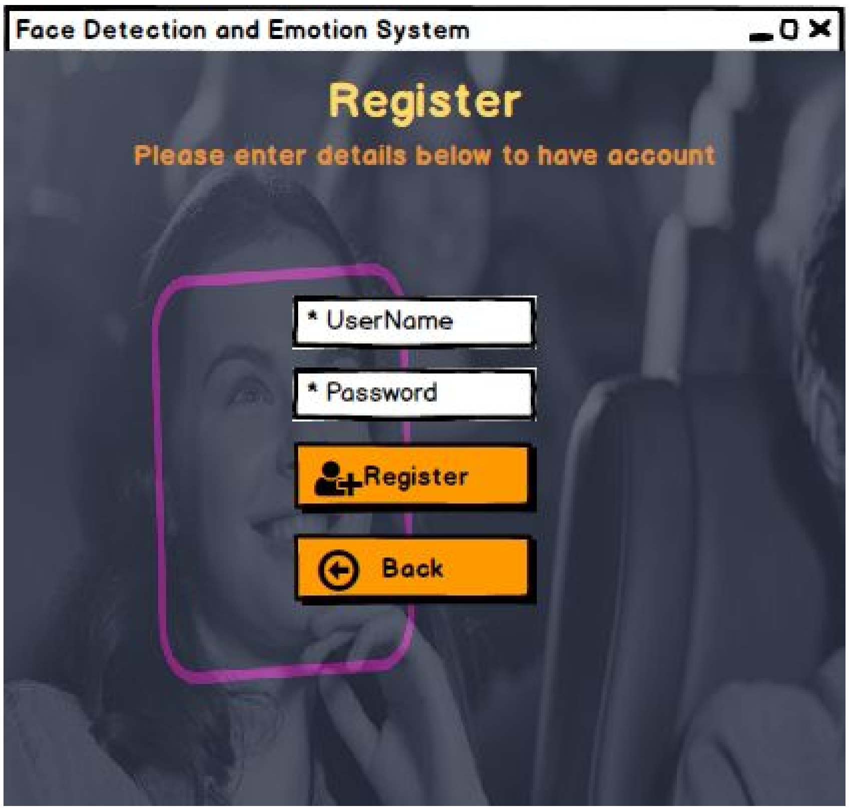

Face Detection and Emotion System
Register
Please enter details below to have account
* UserName
* Password
Register
Back

Figure [25]: Register page .

Login Page:

In this page we'll find two text fields in them the user can enter the user name and the password and the login button ,then he/she access the system.The following image is the register page.

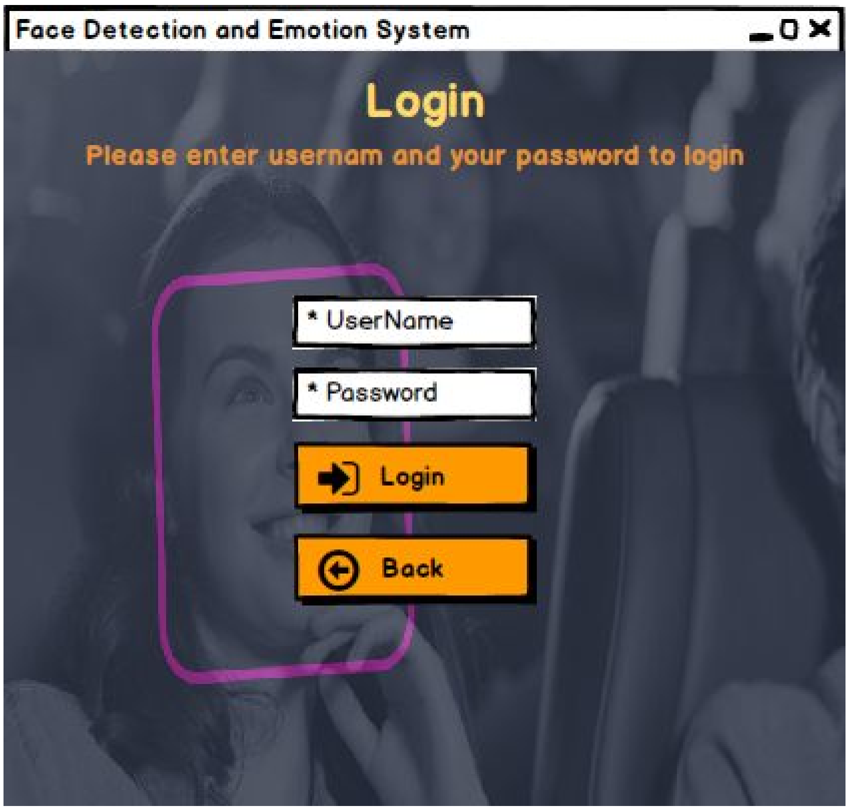
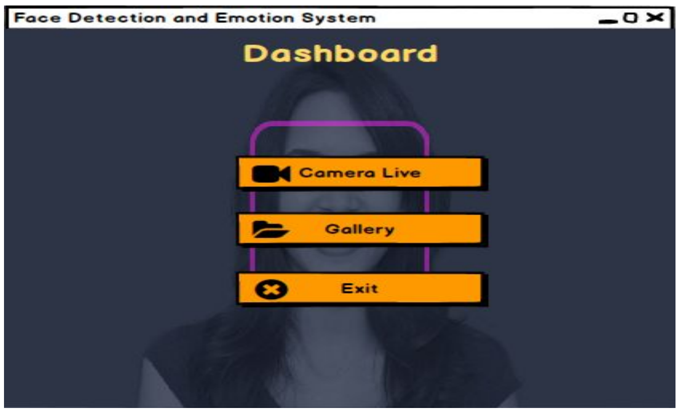

Face Detection and Emotion System
Login
Please enter usernam and your password to login
* UserName
* Password
Login
Back

Figure [26]: Login page .

Dashboard Page:

Now the user has the possibility to use the system, the page it contains two buttons the first one the Camera Live button ,the user can access the camera and the system will detect face and emotion. This is a screenshot of the user open camera live button ,as shown in the figure[24]. And second one Gallery button , the user can choose an image from your device and the system show image with detect face and emotion for you , as shown in the figure[25]. The following image is the Dashboard page ,figura[26] .



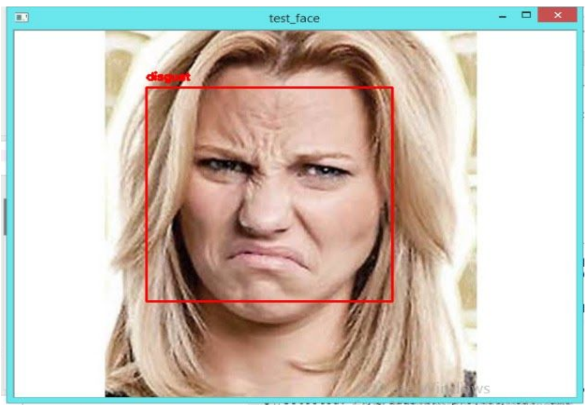
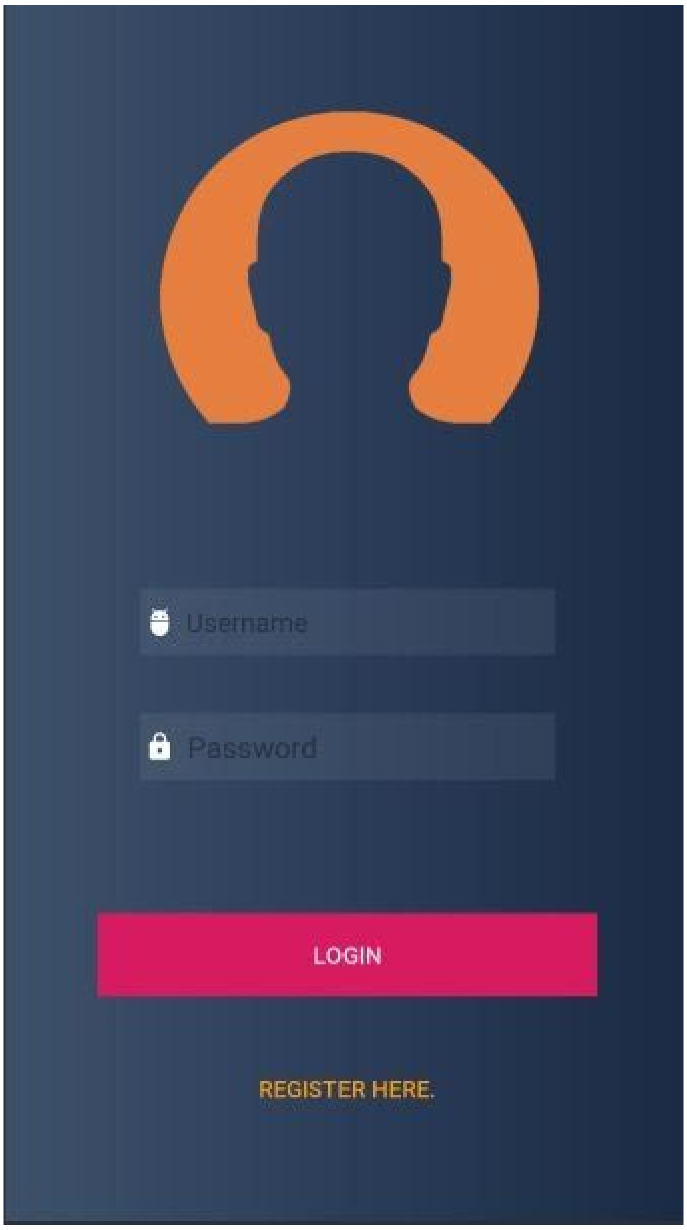


Figure [27]: dashboard page

Figure [28]: input image.

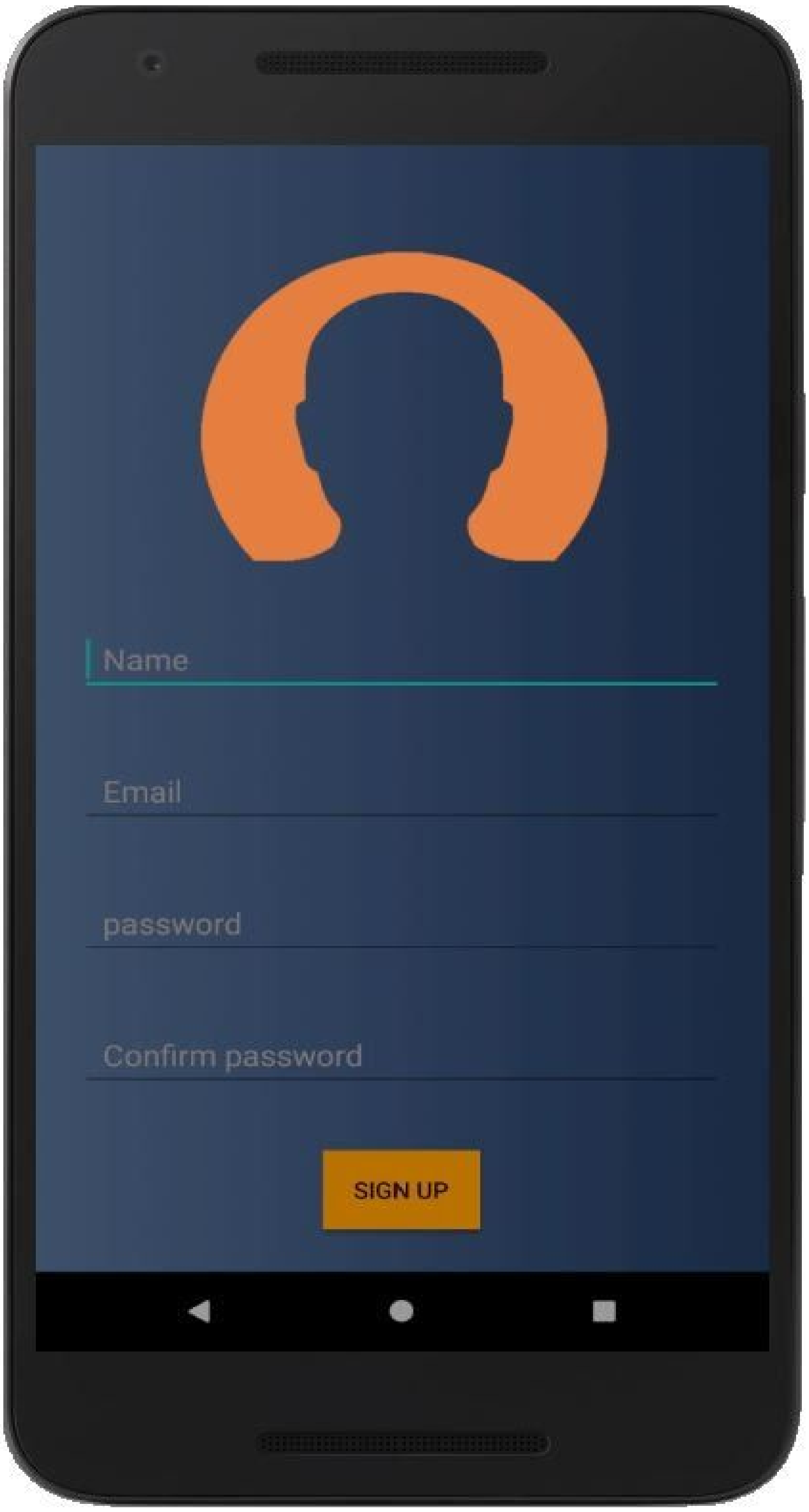
Mobile Application:

Login Page:



figure[29]:login page

Register Page:



figure[30]:register page.

Software testing is a critical element of software quality assurance and represents the ultimate service of specification design and coding. The increasing visibility of software as a system element and the attended costs associated with the software failure and motivating forces for well planned, thorough testing. It is not unusual for a software development to spend between 30 and 40 percent of total project effort in testing. System Testing Strategies for this system integrate test case design techniques into a well planned series of steps that result in the successful construction of this software. It also provides a road map for the developer, the quality assurance organization and the customer, a roadmap that describes the steps to be conducted as path of testing, when these steps are planned and then undertaken and how much effort, time and resources will be required.

The test provisions are as follows :-

System testing

Software Testing: As the coding is completed according to the requirements we have to test the quality of the software. Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. Although testing is to uncover the errors in the software but it also demonstrates that software functions appear to be working as per the specifications, those performance requirements appear to have been met. In addition, data collected as testing is conducted provide a good indication of software and some indications of software quality as a whole. To assure the software quality we conduct both White Box Testing and Black Box Testing.

There are many test cases designed with this is mind. The flow of testing is as follows :-

Code Testing

Specification testing is done to check if the program does with it should do and how it should behave under various conditions or combinations and submitted for processing in the system and it’s checked if any overlaps occur during the processing. This strategy examines the logic of the program. Here only syntax of the code is tested. In code testing syntax errors are corrected, to ensure that the code is perfect.

Unit Testing

The first level of testing is called unit testing. Here different modules are tested against the specifications produced during the design of the modules. Unit testing is done to test the working of individual modules with test oracles. Unit testing comprises a set of tests performed by an individual programmer prior to integration of the units into a large system. A program unit is small enough that the programmer who developed if can test it in great detail. Unit testing focuses first on the modules to locate errors. These errors are verified and corrected so that the unit perfectly fits to the project.

System Testing

The next level of testing is system testing and acceptance testing. This testing is done to check if the system has its requirements and to find the external behavior of the system. System testing involves two kinds of activities:

1. Integration testing
2. Acceptance testing

Integration Testing

The next level of testing is called the Integration Testing. In this many tested modules are combined into subsystems, which were tested. Test case data is prepared to check the control flow of all the modules and to exhaust all possible inputs to the program. Situations like treating the modules when there is no data entered in the text box is also tested. This testing strategy dictates the order in which modules must be available, and exerts a strong influence on the order in which the modules must be written, debugged and unit tested. In integration testing, all the modules / units on which unit testing is performed are integrated together and tested.

Acceptance Testing

This testing is performed finally by user to demonstrate that the implemented system satisfies its requirements. The user gives various inputs to get required outputs.

Specification Testing

Specification testing is done to check if the program does what is should do and how it should behave under various conditions or combination and submitted for processing in the system and it is checked if any overlaps occur during the processing.

Testing Objectives

The following are the testing objectives….

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an as yet undiscovered error.
- A successful test is one that uncovers an as yet undiscovered error.

The above objectives imply a dramatic change in view point. They move counter to the commonly held view that a successful test is one in which no errors are found. Our objective is to design tests that systematically verify different clauses of errors and do so with minimum amount of time and effort. If testing is conducted successfully, it will uncover errors in the software. As a secondary benefit, testing demonstrates that software functions appear to be working according to specification and that performance requirements appear to have been met. In addition, data collected as testing is conducted provides a good indication of software. Testing can't show the absence of defects, it can only show that software errors are present. It is important to keep this stated in mind as testing is being conducted.

Test cases

Test cases for login page

no	Task	Expected result	Obtained result	Remarks
1	Using valid username and password	Successful authentication	As expected	success
2	Using invalid username	Authentication failed	As expected	Invalid user name
3	Using invalid password	Authentication failed	As expected	Username and password are not correct
4	Without giving username and password	Authentication failed	As expected	Please enter user name and password
5	Username and without password	Authentication failed	As expected	Password cannot be empty

Table 2: Test cases for login page .

Test cases for registration page

no	Task	Expected result	Obtained result	Remarks
1	register your name and password	Registration success	As expected	success
2	not register your name	Should not allow to register	As expected	you should enter your name to be register
3	not register your password	Should not allow to register	As expected	you should enter your password to be register
4	Without register username and password	Should not allow to register	As expected	you should enter your name and password to be registe

Table 3: Test cases for registration page.

FUTURE WORK

Machine learning models are biased in accordance to their training data. In our specific application we have empirically found that our trained CNNs for gender classification are biased towards western facial features and facial accessories. We hypothesize that this misclassfications occurs since our training dataset consist of mostly western: actors, writers and cinematographers as observed in Figure 2. Furthermore, as discussed previously, the use of glasses might affect the emotion classification by interfering with the features learned. However, the use of glasses can also interfere with the gender classification. This might be a result from the training data having most of the images of persons wearing glasses assigned with the label “man”. We believe that uncovering such behaviours is of extreme importance when creating robust classifiers, and that the use of visualization techniques such as guided back-propagation will become invaluable when uncovering model biases.

CONCLUSIONS

We began by eliminating completely the fully connected layers and by reducing the amount of parameters in the remaining convolutional layers via depth-wise separable convolutions. We have shown that our proposed models can be stacked for multi-class classifications while maintaining real-time inferences. Specifically, we have developed a vision system that performs face detection, gender classification and emotion classification in a single integrated module. We have achieved human-level performance in our classifications tasks using a single CNN that leverages modern architecture constructs. Our architecture reduces the amount of parameters 80× while obtaining favorable results. Our complete pipeline has been successfully integrated in a Care-O-bot 3 robot. Finally we presented a visualization of the learned features in the CNN using the guided back-propagation visualization. This visualization technique is able to show us the high-level features learned by our models and discuss their interpretability.

References

[1]https://www.facefirst.com/blog/face-detection-vs-face-recognition/.

[2]https://pjreddie.com/darknet/yolo/.

[3]https://www.facefirst.com/blog/brief-history-of-face-recognition-software/.

[4]https://www.equifax.co.uk/resources/identity_protection/identity-fraud-how-they-could-try-to-access-your-information.

[5]https://www.equifax.co.uk/resources/identity_protection/7-signs-of-identity-theft.

[6]https://www.equifax.co.uk/Products/credit/credit-score.

[7]https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.

[8]https://medium.com/@smallfishbigsea/faster-r-cnn-explained-864d4fb7e3f8.

[9]https://medium.com/@hinasharma19se/facial-expressions-recognition-b022318d842a.

[10]https://en.wikipedia.org/wiki/Emotion_recognition.

[11]https://www.marketingmag.com.au/hubs-c/facial-recognition-technology-key-consumer-emotion-detection/.

[12]https://arxiv.org/pdf/1710.07557.pdf

[13]https://appliedmachinelearning.blog/2018/03/24/achieving-90-accuracy-in-object-recognition-task-on-cifar-10-dataset-with-keras-convolutional-neural-networks