PROJECT SYNOPSIS

# "Facial Expressions Detection System using AI Based Techniques"

University of Pune

SUBMITTED TO SAVITRIBAI PHULE PUNE UNIVERSITY
UNDER THE GUIDANCE OF

Prof. Kaklyani
Alishetty
Prof.Smita Maam

SUBMITTED BY

ADITYA RAWAT AND
EDWIN AUCHITE

# TABLE OF CONTENTS

| | Page No. |
|---|---|
| **Acknowledgment** | **ii** |
| **Candidate declaration** | **iii** |
| **Supervisor declaration** | **iv** |
| **List of Figures** | **vi** |
| **List of Tables** | **vii** |
| **Abstract** | **viii** |
| CHAPTER 1: INTRODUCTION | 1 |
| 1.1 NEED OF THE STUDY | 2 |
| 1.2 SCOPE OF THE STUDY | 3 |
| 1.3 OBJECTIVE OF THE STUDY | 3 |
| 1.4 PROBLEM OF DEFINITION | 4 |
| CHAPTER 2: LITERATURE REVIEW | 6 |
| 2.1 SURVEY | 6 |
| 2.2 APPROACHES | 8 |
| 2.3 TECHNOLOGY USED | 12 |
| 2.4 SOFTWARE REQUIREMENT | 13 |
| CHAPTER 3: IMPLEMENTATION DETAILS | 14 |
| 3.1 PLANNING | 14 |
| 3.2 DESIGN | 15 |
| 3.3 ALGORITHM | 21 |
| 3.4 IMPLEMENTATION | 22 |
| 3.5 CODING | 26 |
| CHAPTER 4: RESULTS | 36 |
| CHAPTER 5: CONCLUSIONS AND FUTURE SCOPE | 40 |
| REFERENCE | 42 |

# LIST OF FIGURES

| List of Figures | Title | Page No. |
|---|---|---|
| Figure 1 | Emotions Classification | 2 |
| Figure 2 | Expression Detection Need | 2 |
| Figure 3 | Deaf and Dumb | 3 |
| Figure 4 | System Flow Of Project | 5 |
| Figure 5 | Preprocessing Of Images | 6 |
| Figure 6 | Face Registration | 7 |
| Figure 7 | Facial Feature Extraction | 7 |
| Figure 8 | Emotion Classification | 8 |
| Figure 9 | Neural Network Example | 8 |
| Figure 10 | Gabor Filter | 9 |
| Figure 11 | Iterative Waterfall Model | 15 |
| Figure 12 | 7 Basic Facial Expression | 22 |
| Figure 13 | CNN Model | 24 |
| Figure 14 | Reading From CSV File | 36 |
| Figure 15 | Summary Of Model | 37 |
| Figure 16 | Dataset Trained Successfully | 37 |
| Figure 17 | Accuracy Check Of The Model & Json | 38 |

# LIST OF TABLES, DFD AND GRAPHS

| List | Title | Page No. |
|---|---|---|
| Table 1 | Accuracy of Database | 12 |
| DFD 1 | Level 0 | 18 |
| DFD 2 | Level 1 | 19 |
| DFD 3 | Level 2 | 20 |
| Graph 1 | FER 2013 Dataset | 22 |

## ABSTRACT

These Human facial expressions convey a lot of information visually rather than articulately. Facial expression recognition plays a crucial role in the area of human-machine interaction. Automatic facial expression recognition systems have many applications including, but not limited to, human behavior understanding, detection of mental disorders, and synthetic human expressions. Recognition of facial expression by computer with a high recognition rate is still a challenging task.

Two popular methods utilized mostly in the literature for the automatic FER systems are based on geometry and appearance. Facial Expression Recognition is usually performed in four stages consisting of pre-processing, face detection, feature extraction, and expression classification.

In this project, we applied various deep learning methods (convolutional neural networks) to identify the key seven human emotions: anger, disgust, fear, happiness, sadness, surprise, and neutrality.

# CHAPTER 1:

# INTRODUCTION

**“2018 is the year when machines learn to grasp human emotions” --Andrew Moore, the dean of computer science at Carnegie Mellon.**

With the advent of modern technology, our desires went high and there are no bounds. In the present era, huge research work is going on in the field of digital image and image processing. The way of progression has been exponential and it is ever increasing. Image Processing is a vast area of research in the present day world and its applications are very widespread. Image processing in the field of signal processing where both the input and output signals are images. One of the most important applications of Image processing is Facial expression recognition. Our emotion is revealed by the expressions on our faces. Facial Expressions play an important role in interpersonal communication. Facial expression is a non-verbal scientific gesture that gets expressed in our face as per our emotions. Automatic recognition of facial expression plays an important role in artificial intelligence and robotics and thus it is a need of the generation. Some applications related to this include Personal identification and Access control, Videophone and Teleconferencing, Forensic application, Human-Computer Interaction, Automated Surveillance, Cosmetology, and so on. The objective of this project is to develop an Automatic Facial Expression Recognition System which can take human facial images containing some expression as input and recognize and classify them into seven different expression classes such as :

I. Neutral

II. Angry

III. Disgust

IV. Fear

V. Happy

VI. Sadness

VII. Surprise

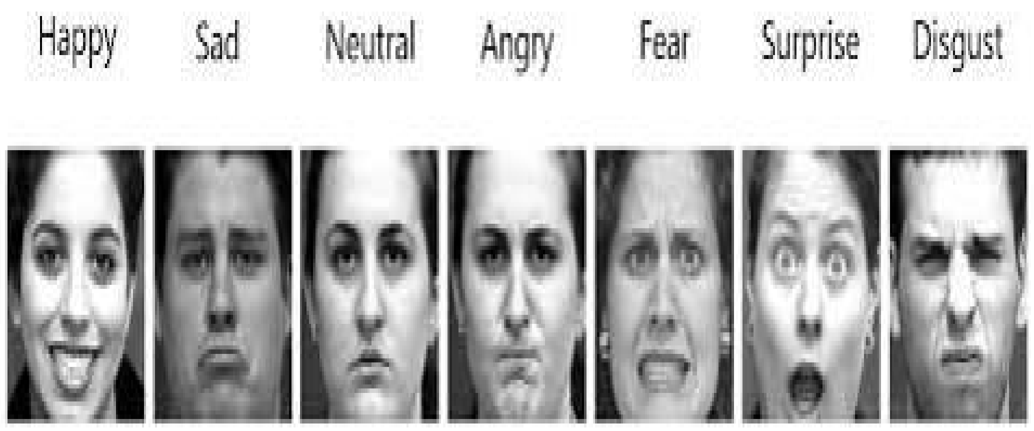


**Figure 1: Emotions Classification**

Several projects have already been done in this field and our goal will not only be to develop an Automatic Facial Expression Recognition System but also to improve the accuracy of this system compared to the other available systems.

## 1.1 NEED OF THE STUDY

Significant debate has risen in the past regarding the emotions portrayed in the world-famous masterpiece of the Mona Lisa. British Weekly „New Scientist" has stated that she is in fact a blend of many different emotions, 83%happy, 9% disgusted, 6% fearful, 2% angry.

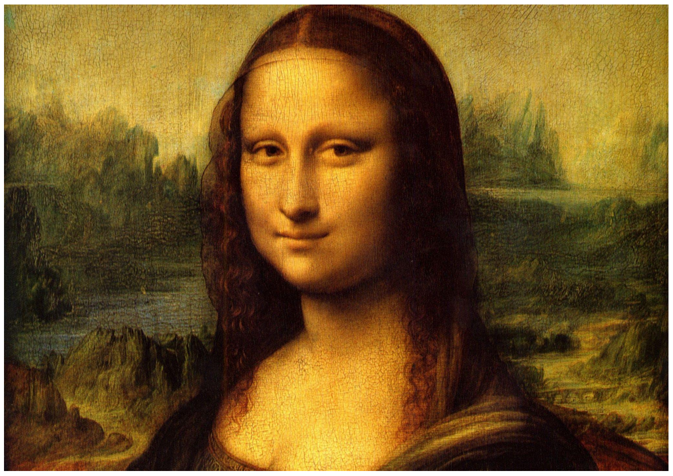


**Figure 2: Expression Detection Need**

We have also been motivated to observe the benefits of physically handicapped people like deaf and dumb. But if any normal human being or an automated system can understand their needs by observing their facial expression then it becomes a lot easier for them to make the fellow human or automated system understand their needs.

**Figure 3 : Deaf and Dumb**

## 1.2 SCOPE OF THE STUDY

In this project, facial expression recognition systems are implemented using convolution neural networks. Facial images are classified into seven facial expression categories namely Anger, Disgust, Fear, Happy, Sad, Surprise, and 'Neutral. Kaggle dataset is used to train and test the classifier.

## 1.3 OBJECTIVE OF THE STUDY

The primary goal of this research is to design, implement and evaluate a novel facial expression recognition system using various statistical learning techniques. This goal will be realized through the following objectives:

- Identify the emotion of a  human face. That is, given a face of a human the system has to automatically identify the type of emotion of the face as happy, anger, disgust, fear, sadness, and surprise.

- Identifying emotions of a human. The emotion can be captured from the face through the camera.

- Psychological characteristics such as heartbeat and blood pressure, speech, hand gestures, body movements, Facial expressions identify the emotions of a person.

- It can be used as a part of many interesting and useful applications like Monitoring security, treating patients in the medical field, marketing research, E-learning, etc.

- The ability of a computer to recognize human emotion has many highly valuable real-world applications. Consider the domain of therapy robots which are designed to provide care and comfort for infirm and disabled individuals.

## 1.4 PROBLEM DEFINITION

Human facial expressions can be easily classified into 7 basic emotions: happy, sad, surprise, fear, anger, disgust, and neutral. Our facial emotions are expressed through the activation of specific sets of facial muscles. These sometimes subtle, yet complex, signals in an expression often contain an abundant amount of information about our state of mind.

Through facial emotion recognition, we are able to measure the effects that content and services have on the audience/users through an easy and low-cost procedure. For example, retailers may use these metrics to evaluate customer interest. Healthcare providers can provide better service by using additional information about patients' emotional state during treatment. Entertainment producers can monitor audience engagement in events to consistently create desired content.

Humans are well-trained in reading the emotions of others, in fact, at just 14 months old, babies can already tell the difference between happy and sad.

But can computers do a better job than us in accessing emotional states? To answer the question, We designed a deep learning neural network that gives machines the ability to make inferences about our emotional states. In other words, we give them eyes to see what we can see.

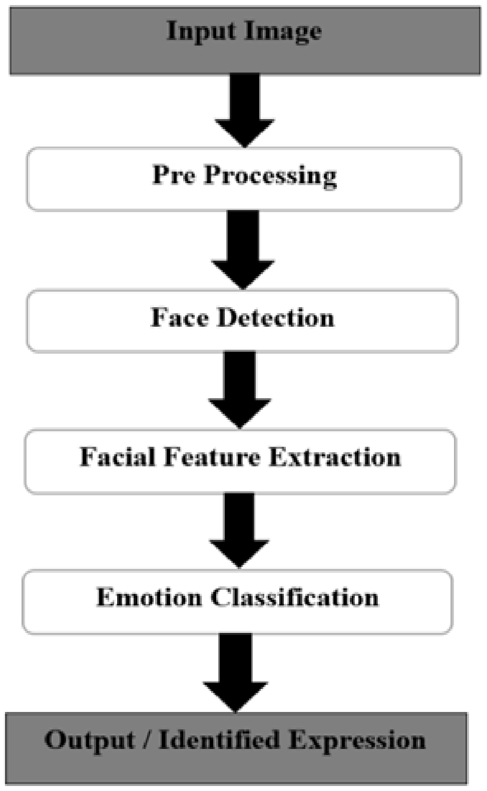


**Figure 4: System Flow Of Project**

# CHAPTER 2:

# LITERATURE REVIEW

## 2.1 SURVEY

As per various literature surveys, it is found that for implementing this project four basic steps are required to be performed.

i. Preprocessing

ii. Face registration

iii. Facial feature extraction

iv. Emotion classification

Description about all these processes are given below-

**Preprocessing**: Preprocessing is a common name for operations with images at the lowest level of abstraction; both input and output are intensity images. Most preprocessing steps that are implemented are –

a. Reduce the noise

b. Convert The Image To Binary/Grayscale.

c. Pixel Brightness Transformation.

d. Geometric Transformation

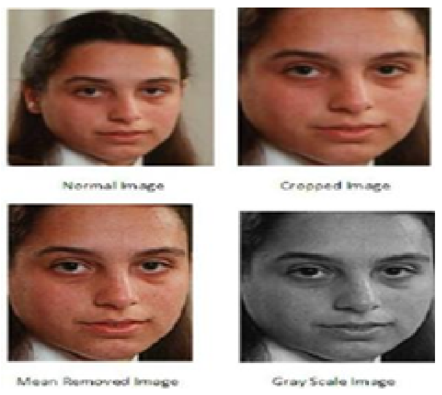


**Figure 5: Preprocessing Of Images**

**Face Registration:** Face Registration is a computer technology being used in a variety of applications that identifies human faces in digital images. In this face registration step, faces are first located in the image using some set of landmark points called “face localization” or “face detection”. These detected faces are then geometrically normalized to match some template image in a process called “face registration”.

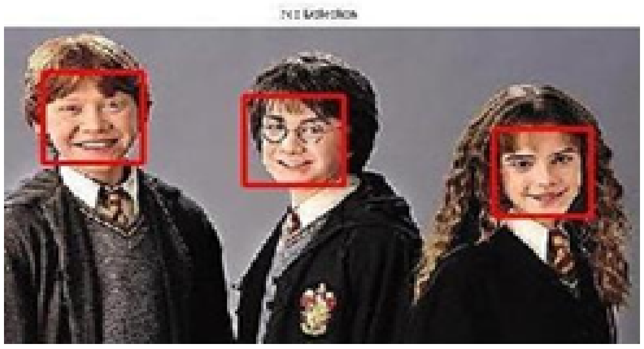


**Figure 6: Face Registration**

**Facial Feature Extraction :** Facial Features extraction is an important step in face recognition and is defined as the process of locating specific regions, points, landmarks, or curves/contours in a given 2-D image or a 3D range image. In this feature extraction step, a numerical feature vector is generated from the resulting registered image. Common features that can be extracted are-

a. Lips

b. Eyes

c. Eyebrows

d. Nose tip

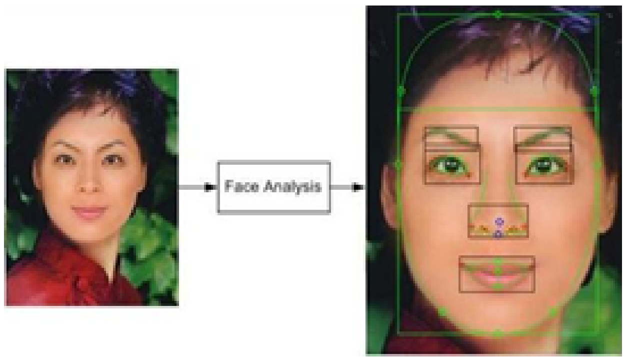


**Figure 7: Facial Feature Extraction**

**Emotion Classification:** In the third step, of classification, the algorithm attempts to classify the given faces portraying one of the seven basic emotions. 4.4. Paul Ekman (born February 15, 1934) is an American psychologist and professor emeritus at the University of California, San Francisco who is a pioneer in the study of emotions and their relation to facial expressions. He has created an "atlas of emotions" with more than ten thousand facial expressions.

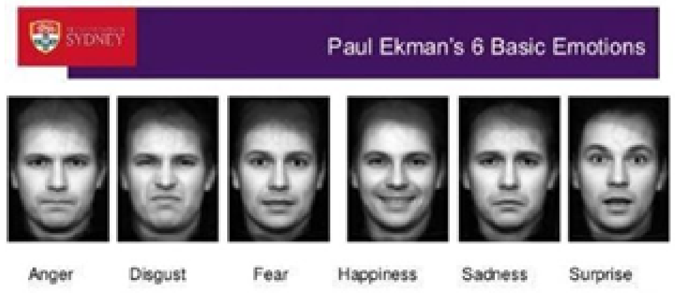


**Figure 8: Emotion Classification**

## 2.2 APPROACHES

Different approaches which are followed for Facial Expression Recognition:

- **Neural Network Approach:** The neural network contains a hidden layer with neurons. The approach is based on the assumption that a neutral face image corresponding to each image is available to the system. Each neural network is trained independently with the use of online backpropagation. Neural networks will be discussed later.

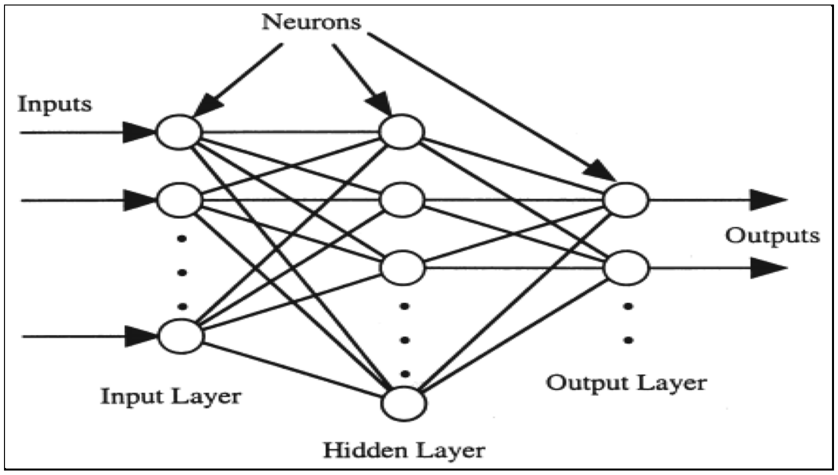


**Figure 9: Neural Network Example**

- **Principal of Component Analysis:** Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called Principal Components.

- **Gabor Filter:** In image processing, a Gabor filter, named after Dennis Gabor, is a linear filter used for texture analysis, which means that it basically analyses whether there is any specific frequency content in the image in specific directions in a localized region around the point or region of analysis. Frequency and orientation representations of Gabor filters are claimed by many contemporary vision scientists to be similar to those of the human visual system, though there is no empirical evidence and no functional rationale to support the idea. They have been found to be particularly appropriate for texture representation and discrimination. In the spatial domain, a 2D Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave.

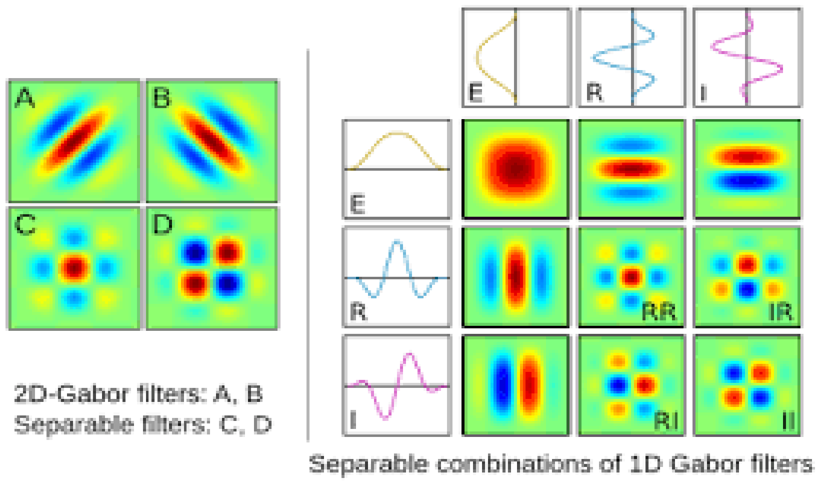


**Figure 10: Gabor Filter**

Gabor filters are directly related to Gabor wavelets since they can be designed for a number of dilations and rotations. However, in general, expansion is not applied for Gabor wavelets, since this requires the computation of bi-orthogonal wavelets, which may be very time-consuming. Therefore, usually, a filter bank consisting of Gabor filters with various scales and rotations is created. The filters are convolved with the signal, resulting in a so-called Gabor space. This process is closely related to processes in the primary visual cortex.

Jones and Palmer showed that the real part of the complex Gabor function is a good fit to the receptive field weight functions found in simple cells in a cat's striate cortex.

- **Support Vector Machine:** In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary model (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

  In addition to performing linear classification, SVMs can efficiently perform a nonlinear classification using what is called the kernel trick implicitly mapping their inputs into high-dimensional feature spaces.

  When data are not labeled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The support vector clustering algorithm created by Hava Siegelmann and Vladimir Vapnik, applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabeled data, and is one of the most widely used clustering algorithms in industrial applications.

- **Training & Testing Database:** In machine learning, the study and construction of algorithms that can learn from and make predictions on data is a common task. Such algorithms work by making data-driven predictions or decisions, through building a mathematical model from input data.

  The data used to build the final model usually comes from multiple datasets. In particular, three data sets are commonly used in different stages of the creation of the model. The model is initially fit on a training dataset, which is a set of examples used to fit the parameters (e.g. weights of connections between neurons in artificial neural networks) of the model.

  The model (e.g. a neural net or a naive Bayes classifier) is trained on the training dataset using a supervised learning method (e.g. gradient descent or stochastic gradient descent). In practice, the training dataset often consists of pairs of an input vector and the corresponding answer vector or scalar, which is commonly denoted as the target.

The current model runs with the training dataset and produces a result, which is then compared with the target, for each input vector in the training dataset. Based on the result of the comparison and the specific learning algorithm being used, the parameters of the model are adjusted. The model fitting can include both variable selection and parameter estimation.

Successively, the fitted model is used to predict the responses for the observations in a second dataset called the validation dataset. The validation dataset provides an unbiased evaluation of a model fit on the training dataset while tuning the model's hyperparameters (e.g. the number of hidden units in a neural network). Validation datasets can be used for regularization by early stopping: stop training when the error on the validation dataset increases, as this is a sign of overfitting to the training dataset. This simple procedure is complicated in practice by the fact that the validation dataset's error may fluctuate during training, producing multiple local minima. This complication has led to the creation of many ad-hoc rules for deciding when overfitting has truly begun.

Finally, the test dataset is a dataset used to provide an unbiased evaluation of a final model fit on the training dataset.

Various facial datasets available online are:

1. Japanese Female Facial Expression (JAFFE)

2. FER

3. CMU MultiPIE

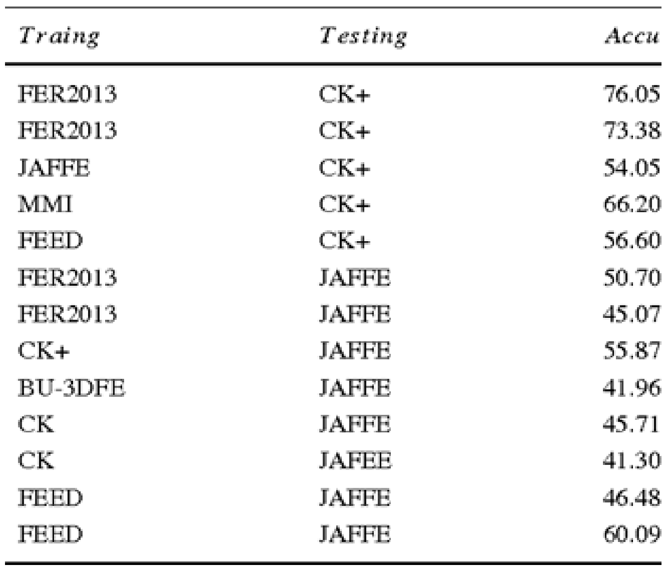
4. Lifespan

5. MMI

6. FEED

7. CK

Accuracy of various databases:



| *Traing* | *Testing* | *Accu* |
|---|---|---|
| FER2013 | CK+ | 76.05 |
| FER2013 | CK+ | 73.38 |
| JAFFE | CK+ | 54.05 |
| MMI | CK+ | 66.20 |
| FEED | CK+ | 56.60 |
| FER2013 | JAFFE | 50.70 |
| FER2013 | JAFFE | 45.07 |
| CK+ | JAFFE | 55.87 |
| BU-3DFE | JAFFE | 41.96 |
| CK | JAFFE | 45.71 |
| CK | JAFEE | 41.30 |
| FEED | JAFFE | 46.48 |
| FEED | JAFFE | 60.09 |

**Table 1: Accuracy of databases**

## 2.3 TECHNOLOGIES USED

**OpenCV:** OpenCV is the library we will be using for image transformation functions such as converting the image to grayscale. It is a complete package that can be used with other libraries to form a pipeline for any image extraction or detection framework.

**Python:** Python is a powerful scripting language and is very useful for solving statistical problems involving machine learning algorithms. It has various utility functions which help in preprocessing. It provides the pandas and NumPy framework which helps in the manipulation of data as per our needs. A good feature set can be created using the NumPy arrays which can have n-dimensional data.

**TensorFlow:** TensorFlow is an open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning.TensorFlow accepts data in the form of multi-dimensional arrays of higher dimensions called tensors. Multi-dimensional arrays are very handy in handling large amounts of data.TensorFlow works on the basis of data flow graphs that have nodes and edges. As the execution mechanism is in the form of graphs, it is much easier to execute TensorFlow code in a distributed manner across a cluster of computers while using GPUs.

**HaarCascade Classifier:** It is a machine learning-based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. Here we will work with face expression detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the popular data science packages suitable for Windows, Linux, and macOS.

**Jupyter Notebook** The Jupyter Notebook is an open-source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it classifier. Then we need to extract features from it. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting the sum of pixels under the white rectangle from the sum of pixels under the black rectangle

## 2.4 SOFTWARE REQUIREMENTS

As the project is developed in python, we have used Anaconda and Jupyter Notebook for Python 3.6.5.

**Anaconda:** It is a free and open-source distribution of the Python and R programming languages for data science and machine learning-related applications (large-scale data processing, predictive analytics, scientific computing), that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution is used by over 6 million users, and it includes more than 250 p supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

**Hardware Interfaces**

1. Processor: Intel CORE i5 processor with minimum 2.9 GHz speed.

2. RAM: Minimum 4 GB.

3. Hard Disk: Minimum 500 GB

**Software Interfaces**

1. Microsoft Word 2003

2. Database Storage: Microsoft Excel

3. Operating System: Windows10

# CHAPTER 3:

# IMPLEMENTATION DETAILS

## 3.1 PLANNING

The steps we followed while developing this project are:

1. Analysis of the problem statement.
2. Gathering of the requirement specification.
3. Analysation of the feasibility of the project.
4. Development of a general layout.
5. Going by the journals regarding the previous related works in this field.
6. Choosing the method for developing the algorithm.
7. Analyzing the various pros and cons.
8. Starting the development of the project.
9. Installation of software like ANACONDA.
10. Developing an algorithm.
11. Analysation of the algorithm by the guide.
12. Coding as per the developed algorithm in PYTHON.

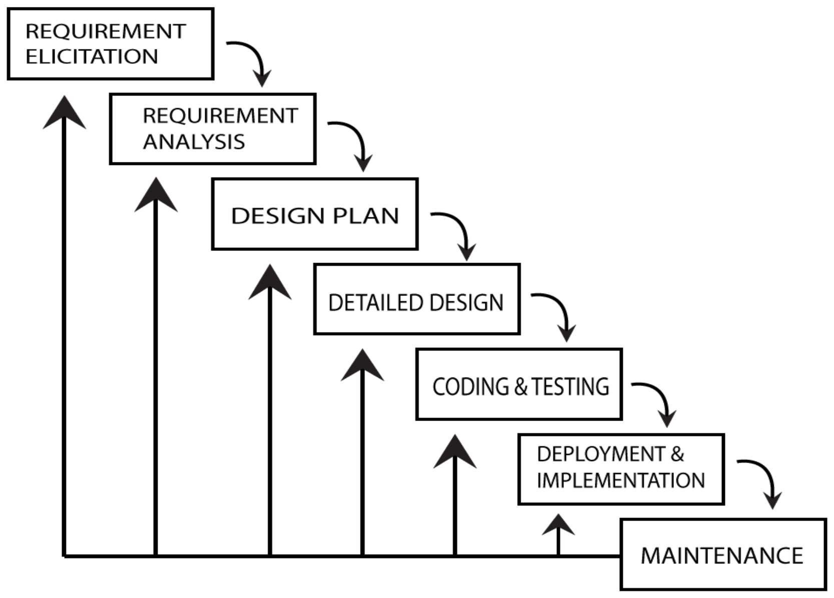


**Figure 11: Iterative Waterfall Model**

## 3.2 DESIGN

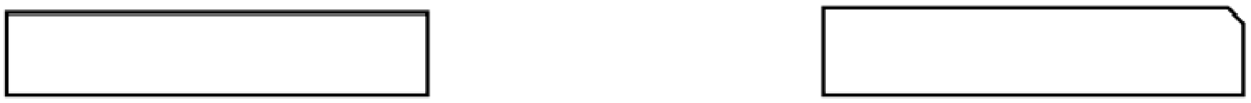
### DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditionally structured flowchart that focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model. Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top down approach to Systems Design.

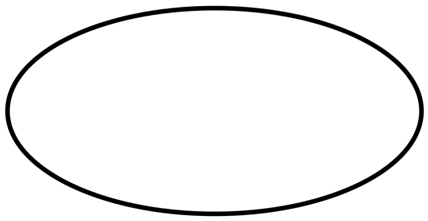
### Symbols and notations used in DFDs

Using any convention‟s DFD rules or guidelines, the symbols depict the four components of data flow diagrams -

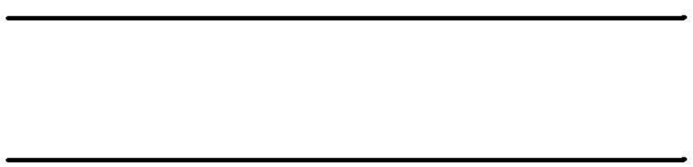
**External entity:** an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system, or a business system. They are also known as terminators, sources, and sinks or actors. They are typically drawn on the edges of the diagram.



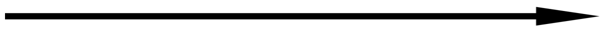
**Process:** any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules.



**Data store:** files or repositories that hold information for later use, such as a database table or a membership form.



**Data flow:** the route that data takes between the external entities, processes, and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data-name, like "Billing details."



**DFD levels and layers**

A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1, or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish.

**DFD Level 0** is also called a Context Diagram. It‟s a basic overview of the whole system or process being analyzed or modeled. It‟s designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts, and developers.

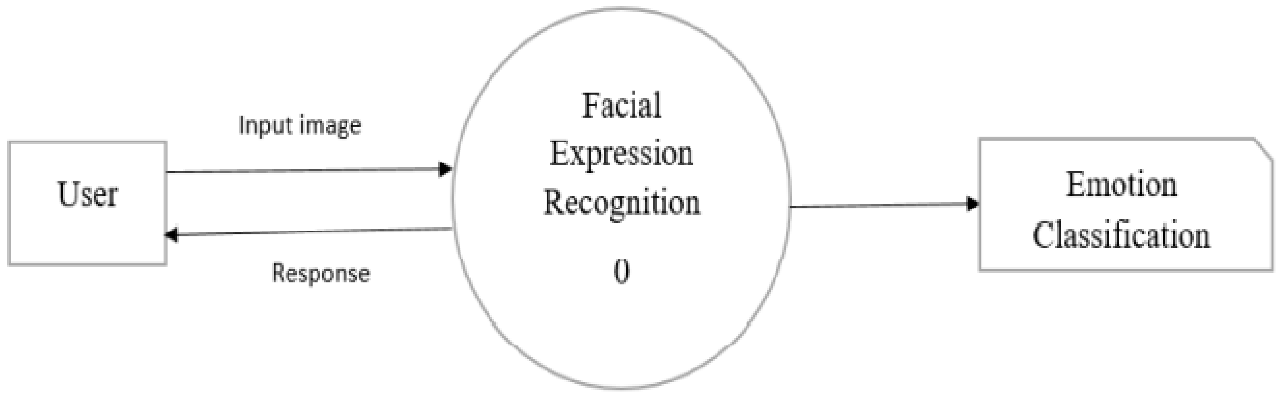
**DFD Level 1** provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its subprocesses.

**DFD Level 2** then goes one step deeper into parts of Level 1. It may require more text to reach the necessary level of detail about the system‟s functioning.

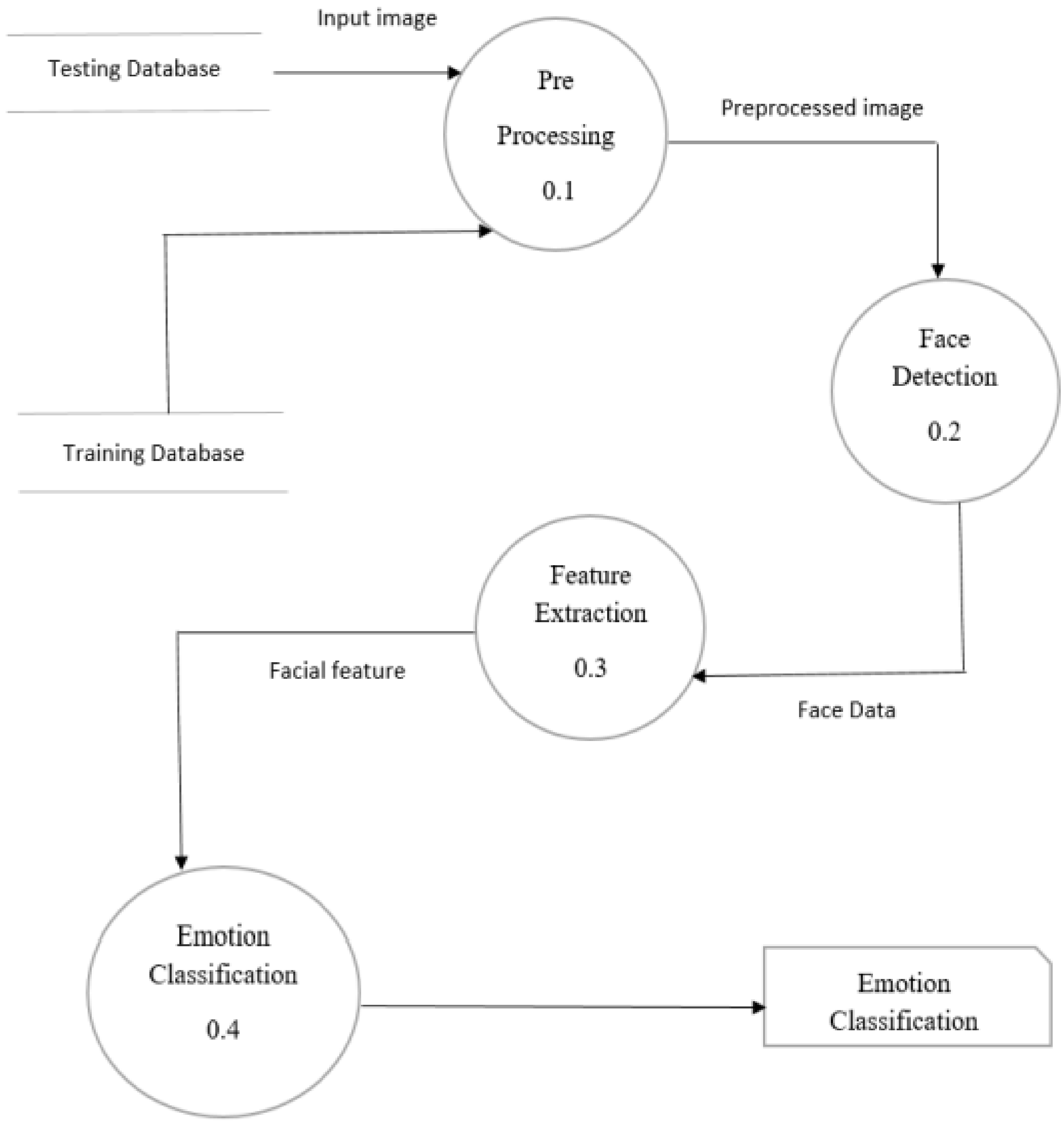
Progression to Levels 3, 4, and beyond is possible, but going beyond Level 3 is uncommon. Doing so can create complexity that makes it difficult to communicate, compare or model effectively.

Using DFD layers, the cascading levels can be nested directly in the diagram, providing a cleaner look with easy access to the deeper dive.

## DFD 1: Level 0

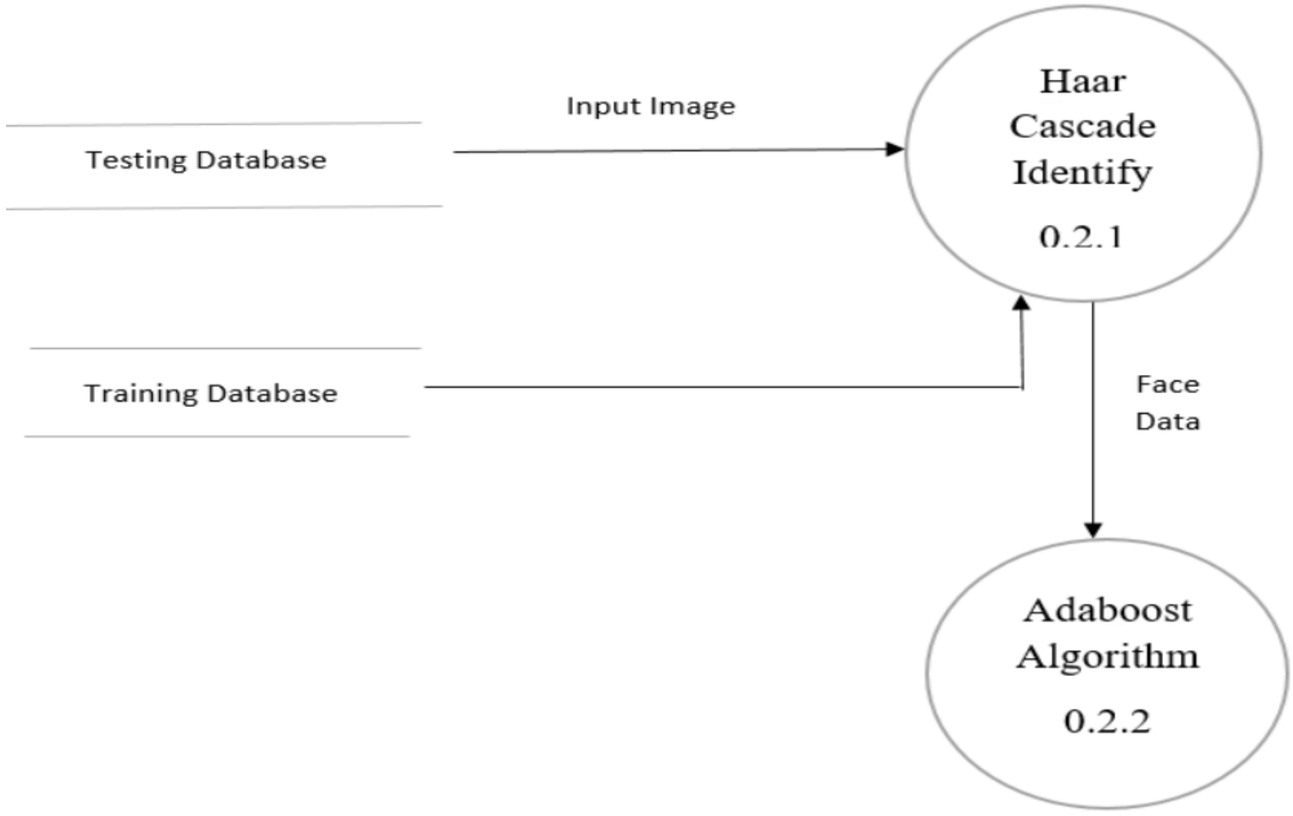


## DFD 2: Level 1

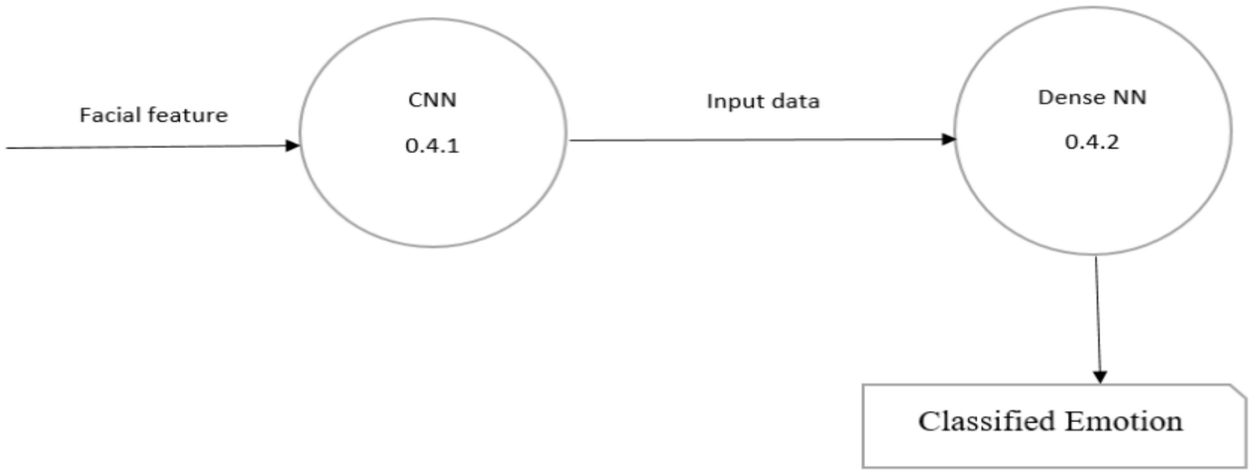

Input image
Testing Database
Pre
Processing
0.1
Preprocessed image
Training Database
Face
Detection
0.2
Feature
Extraction
0.3
Facial feature
Face Data
Emotion
Classification
0.4
Emotion
Classification

## DFD 3: Level 2

**Face Detection-**



**Emotion Classification-**



## 3.3 ALGORITHM

**Step 1:** Collection of a data set of images.

(pre-cropped, 48-by-48-pixel grayscale images of faces each labeled with one of the 7 emotion classes):

anger, disgust, fear, happiness, sadness, surprise, and neutral.

**Step 2:** Pre-processing of images.

**Step 3:** Detection of a face from each image.

**Step 4:** The cropped face is converted into grayscale images.

**Step 5:** The pipeline ensures every image can be fed into the input layer as a (1, 48, 48) NumPy array.

**Step 6:** The NumPy array gets passed into the Convolution2D layer.

**Step 7:** Convolution generates feature maps.

**Step 8:** Pooling method called AveragePooling2D and GlobalAveragePooling is applied across the feature map that only keeps the Average pixel value.

**Step 9:** During training, Neural network Forward propagation and backward propagation are performed on the pixel values.

**Step 10:** The Softmax function presents itself as a probability for each emotion class.

The model is able to show the detailed probability composition of the emotions in the face.

## 3.4 IMPLEMENTATION

### The Database

The dataset, used for training the model is from a Kaggle Facial Expression Recognition Challenge a few years back (FER2013). The data consists of 48x48 pixel grayscale 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 into one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

There are a total of 35,887 images. The training set consists of 28,709 examples. The public test set used for the leaderboard consists of 7,178 examples.

**Emotion labels in the dataset:**

**0:** -4953 images- **Angry**

**1:** -547 images- **Disgust**

**2:** -5121 images- **Fear**

**3:** -8989 images- **Happy**

**4:** -6077 images- **Sad**

**5:** -4002 images- **Surprise**

**6:** -6198 images- **Neutral**

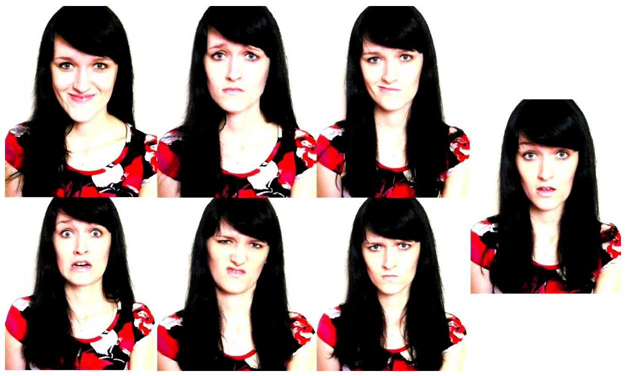
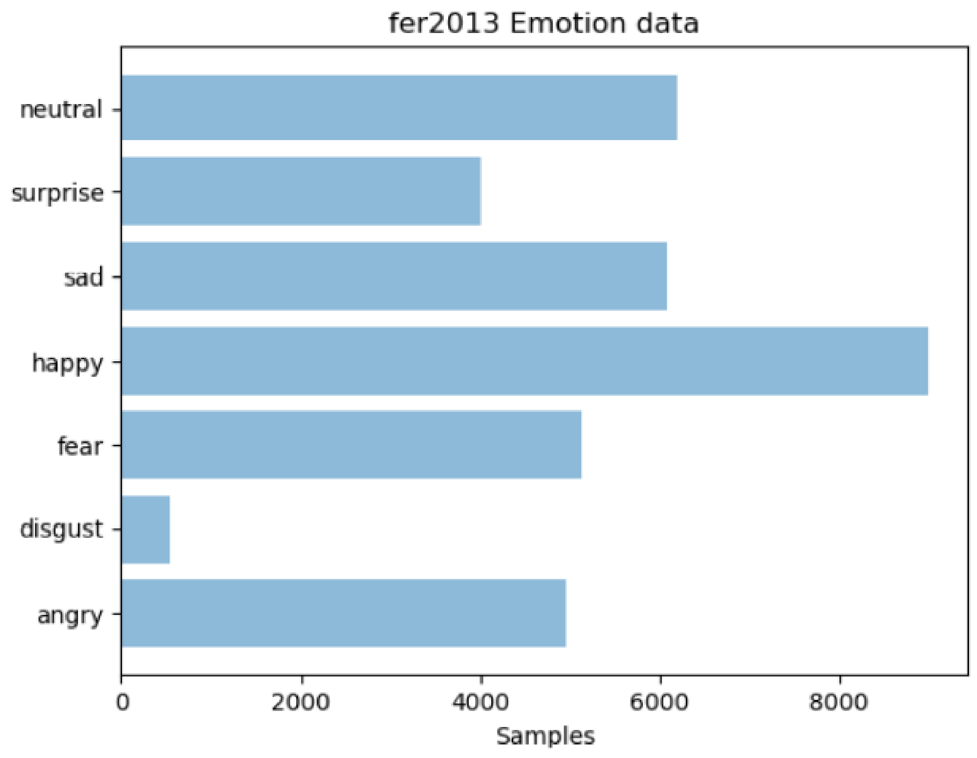


**Figure 12: 7 Basic Facial Expression**



**Graph 1: FER2013 Dataset**

**The Model**

Deep learning is a popular technique used in computer vision. We chose Convolutional Neural Network (CNN) layers as building blocks to create our model architecture. CNNs are known to imitate how the human brain works when analyzing visuals.

A typical architecture of a convolutional neural network contains an input layer, some convolutional layers, some dense layers (aka. fully-connected layers), and an output layer. These are linearly stacked layers ordered in sequence. In Keras, the model is created as Sequential() and more layers are added to build architecture.

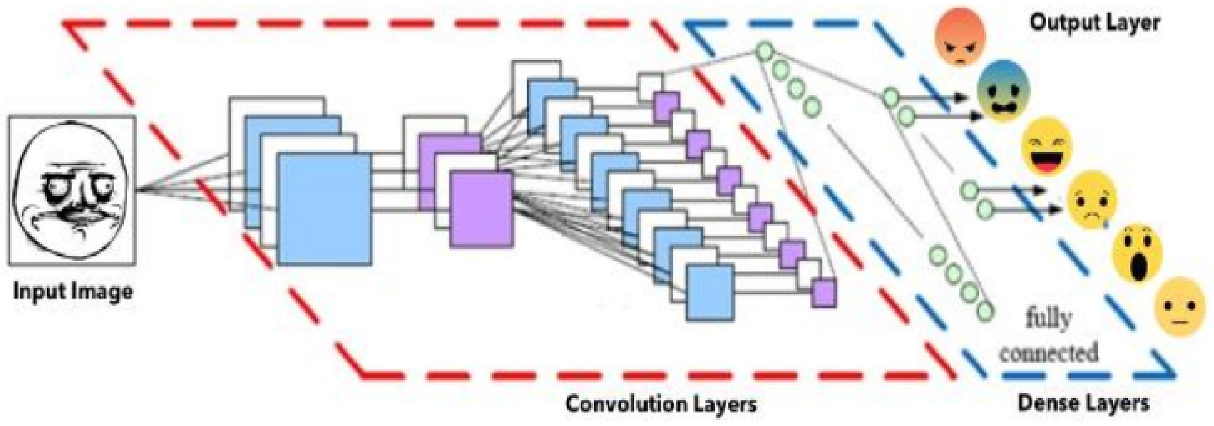


**Figure 13: CNN Model**

- **Input Layer:** The input layer has predetermined, fixed dimensions, so the image must be pre-processed before it can be fed into the layer. We used OpenCV, a computer vision library, for face detection in the image. The haarcascade_frontalface_default.xml in OpenCV contains pre-trained filters and uses Adaboost to quickly find and crop the face.

  The cropped face is then converted into grayscale using cv2.cvtColor and resized to 48-by-48 pixels with cv2.resize. This step greatly reduces the dimensions compared to the original RGB format with three color dimensions (3, 48, 48). The pipeline ensures every image can be fed into the input layer as a (1, 48, 48) NumPy array.

- **Convolutional Layer:** The NumPy array gets passed into the Convolution2D layer where we specify the number of filters as one of the hyperparameters. The set of filters(aka. kernel) are unique with randomly generated weights. Each filter, (3, 3) receptive field, slides across the original image with shared weights to create a feature map. Convolution generates feature maps that represent how pixel values are enhanced, for example, edge and pattern detection. A feature map is created by applying filter 1 across the entire image. Other filters are applied one after another creating a set of feature maps.

- **Pooling Layer:** Pooling is basically "downscaling" the image obtained from the previous layers. It can be compared to shrinking an image to reduce its pixel density. AveragePooling retains much information about the "less important" elements of a block, or pool. Another type of pooling layer is the GlobalAveragePooling layer. It is used to reduce the dimensionality of the feature maps output by some convolutional layer, to replace Flattening and sometimes even Dense layers in the classifier.

- **Normalization Layer:** This layer normalizes each feature so that they maintain the contribution of every feature, as some feature has higher numerical value than others. This way the network can be unbiased(to higher-value features).

  Batch Normalization focuses on standardizing the inputs to any particular layer(i.e. activations from previous layers). Standardizing the inputs mean that inputs to any layer in the network should have approximately zero mean and unit variance. Mathematically, this layer transforms each input in the current mini-batch by subtracting the input mean in the current mini-batch and dividing it by the standard deviation.

- **Dropout Layer:** The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting.

- **Output Layer:** Instead of using the sigmoid activation function, we used softmax at the output layer. This output presents itself as a probability for each emotion class. Therefore, the model is able to show the detailed probability composition of the emotions in the face. As later on, you will see that it is not efficient to classify human facial expression as only a single emotion. Our expressions are usually much complex and contain a mix of emotions that could be used to accurately describe a particular expression.

## Model Validation

Performance As it turns out, the final CNN had a validation accuracy of 64.2%. This actually makes a lot of sense. Because our expressions usually consist of a combination of emotions, and only using one label to represent an expression can be hard.

## 3.5 CODING

**Main Code:**

```
import numpy as np

import pandas as pd

import os

import cv2

from keras.preprocessing.image import ImageDataGenerator

from keras.models import Model, Sequential

from keras.layers import Input, Dropout, GlobalAveragePooling2D, AveragePooling2D,
Conv2D, BatchNormalization, Activation

from keras.optimizers import Adam

from keras.regularizers import l2

from keras.callbacks import CSVLogger, ModelCheckpoint, EarlyStopping

from keras.callbacks import ReduceLROnPlateau


# Reading from csv file

df = pd.read_csv('/content/drive/MyDrive/data2013.csv')

df

image_size=(48,48)
```

```
pixels = df['pixels'].tolist() # Converting the column element into a list for each row

width, height = 48, 48

faces = []

for pixel_sequence in pixels:
        face = [int(pixel) for pixel in pixel_sequence.split(' ')]
        #Splitting the string by space character as a list

        face = np.asarray(face).reshape(width, height)
         #converting the list to numpy array of size 48*48

        face = cv2.resize(face.astype('uint8'),image_size)
        #resize the image to have 48 cols (width) and 48 rows (height)

        faces.append(face.astype('float32'))
        #makes the list of each images of 48*48 and their pixels in numpyarray form
faces = np.asarray(faces) #converting the list into numpy array

faces = np.expand_dims(faces, -1)
#Expand the shape of an array -1=last dimension => means color space

emotions = pd.get_dummies(df['emotion']).to_numpy()
#converting categorical variable type i.e., emotions into dummy/indicator variables
x = faces.astype('float32')

x = x / 255.0 #Dividing the pixels by 255 for normalization  => range(0,1)



# Scaling the pixels value in range(-1,1)

x = x - 0.5

x = x * 2.0
```

```
num_samples, num_classes = emotions.shape

num_samples = len(x)

num_train_samples = int((1 - 0.2)*num_samples)


# Training data

train_x = x[:num_train_samples]

train_y = emotions[:num_train_samples]


# Validation data

val_x = x[num_train_samples:]

val_y = emotions[num_train_samples:]


train_data = (train_x, train_y)

val_data = (val_x, val_y)


##designing the cnn model

model = Sequential()

# 1st CNN layer

model.add(Conv2D(64, kernel_size=(3, 3), activation='relu', input_shape=(train_x.shape[1:])))

model.add(BatchNormalization())
model.add(Conv2D(16, kernel_size=(7, 7), padding='same'))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(AveragePooling2D(pool_size=(2, 2), padding='same'))

model.add(Dropout(.5))


# 2nd CNN layer

model.add(Conv2D(32, kernel_size=(5, 5), padding='same'))

model.add(BatchNormalization())

model.add(Conv2D(32, kernel_size=(5, 5), padding='same'))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(AveragePooling2D(pool_size=(2, 2), padding='same'))

model.add(Dropout(.5))


# 3rd CNN layer

model.add(Conv2D(64, kernel_size=(3, 3), padding='same'))

model.add(BatchNormalization())

model.add(Conv2D(filters=64, kernel_size=(3, 3), padding='same'))

model.add(BatchNormalization())
model.add(Activation('relu'))

model.add(AveragePooling2D(pool_size=(2, 2), padding='same'))

model.add(Dropout(.5))


# 4th CNN layer

model.add(Conv2D(128, kernel_size=(3, 3), padding='same'))

model.add(BatchNormalization())

model.add(Conv2D(128, kernel_size=(3, 3), padding='same'))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(AveragePooling2D(pool_size=(2, 2), padding='same'))

model.add(Dropout(.5))


# 5th CNN layer

model.add(Conv2D(256, kernel_size=(3, 3), padding='same'))

model.add(BatchNormalization())

model.add(Conv2D(filters=num_classes, kernel_size=(3, 3), padding='same'))

model.add(GlobalAveragePooling2D())

model.add(Activation('softmax',name='predictions'))
```

```
"""Image Data Augmentation is a technique that can be used to artificially expand the size of a
training dataset by creating modified versions of images in the dataset. """

# data generator Generate batches of tensor image data with real-time data augmentation

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/compilation

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

model.summary()


batch_size = 128

num_epochs = 500

verbose = 1

num_classes = 7

patience = 64

datasets = ['data2013']
```

```
base_path="/content"

for dataset_name in datasets:

        print('Training dataset:', dataset_name)

  #callbacks

  log_file_path = dataset_name + '_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 + dataset_name + 'simple_cnn'

  model_names = trained_models_path + '.{epoch:02d}-{val_loss:.2f}.hdf5'

  model_checkpoint = ModelCheckpoint(model_names, 'val_loss',
  verbose=1,save_best_only=True)

  my_callbacks = [model_checkpoint, csv_logger, early_stop, reduce_lr]



  # loading dataset

  train_faces, train_emotions = train_data

  history=model.fit(data_generator.flow(train_faces, train_emotions, batch_size),
 epochs=num_epochs, verbose=1, callbacks=my_callbacks,validation_data =val_data)


#evaluate() returns [loss,acc]

score = model.evaluate(val_x, val_y, verbose=1)

print('Test loss:', score[0])
print('Test accuracy:', score[1]*100)


#saving weights

model.save_weights("data.h5")

#saving architecture

model_json = model.to_json()

with open("data.json", "w") as json_file:

        json_file.write(model_json)

#model.save_weights("model.h5")

print("Saved model to disk")
```

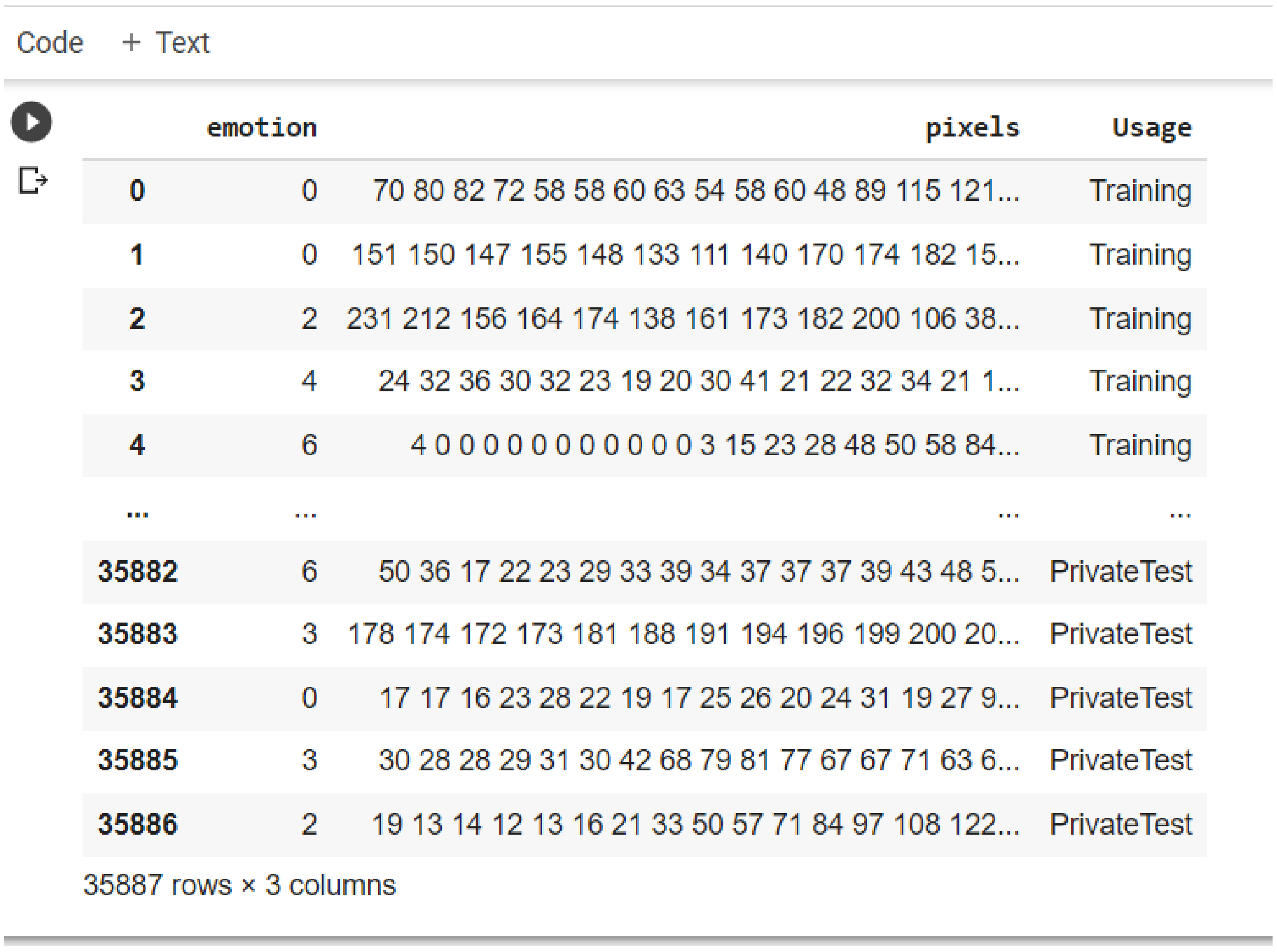
**Live Testing Code:**

```
import os

import cv2

import numpy as np

from keras.models import model_from_json

from keras.preprocessing import image
```

```
#load model

model = model_from_json(open('C:/Users/91945/Desktop/data.json', 'r').read())

#load weights

model.load_weights('C:/Users/91945/Desktop/data.h5')

face_haar_cascade =
cv2.CascadeClassifier('C:/Users/91945/Desktop/Emotion/haarcascade_frontalface_default.xml')

cap=cv2.VideoCapture(0)

while True:

    ret,test_img=cap.read()# captures frame and returns boolean value and captured image

    if not ret:

        continue

    gray_img= cv2.cvtColor(test_img, cv2.COLOR_BGR2GRAY)

    faces_detected = face_haar_cascade.detectMultiScale(gray_img, 1.32, 5)



    for (x,y,w,h) in faces_detected:

        cv2.rectangle(test_img,(x,y),(x+w,y+h),(255,0,0),thickness=3)

        roi_gray = gray_img[y:y+w,x:x+h]"""cropping region of interest i.e. face area
        from image"""

        roi_gray = cv2.resize(roi_gray,(48,48))

        img_pixels = image.img_to_array(roi_gray)

        img_pixels = np.expand_dims(img_pixels, axis = 0)

        img_pixels /= 255

        predictions = model.predict(img_pixels)
        #find max indexed array

        max_index = np.argmax(predictions[0])

        emotions = ('angry', 'disgust', 'fear', 'happy', 'sad', 'surprise', 'neutral')

        predicted_emotion = emotions[max_index]

        cv2.putText(test_img, predicted_emotion, (int(x), int(y)),
        cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)

        resized_img = cv2.resize(test_img, (1000, 700))

        cv2.imshow('Facial emotion analysis ',resized_img)

    if cv2.waitKey(10) == ord('q'): #wait until 'q' key is pressed

        break

cap.release()

cv2.destroyAllWindows
```

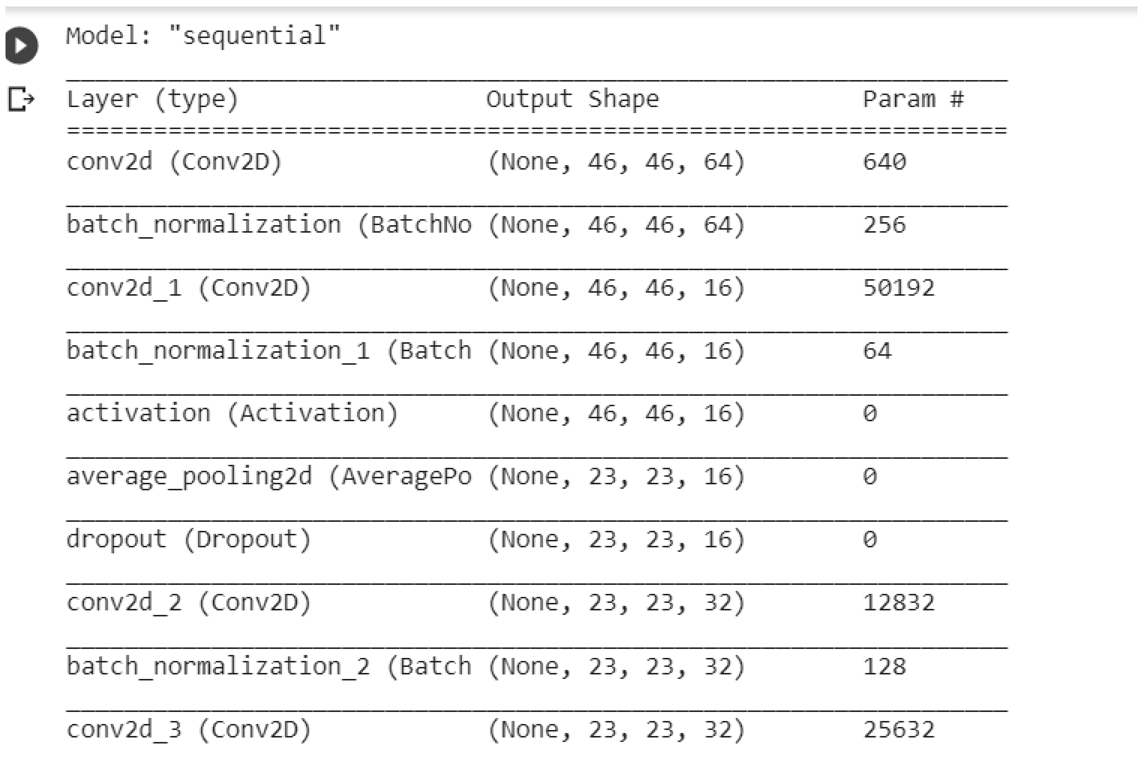
# CHAPTER 4:

# RESULTS



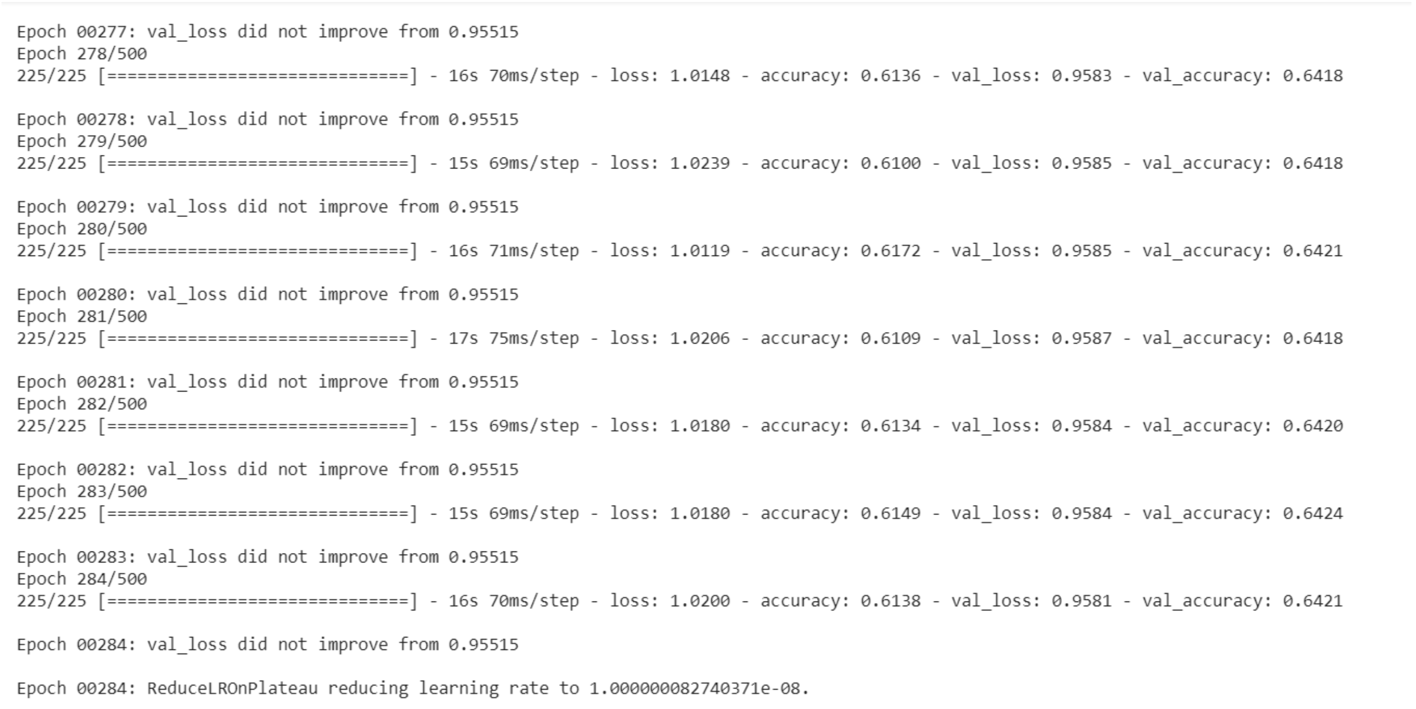
```
       emotion                                             pixels        Usage
0            0    70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...     Training
1            0  151 150 147 155 148 133 111 140 170 174 182 15...     Training
2            2  231 212 156 164 174 138 161 173 182 200 106 38...     Training
3            4     24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...     Training
4            6        4 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...     Training
...        ...                                                  ...          ...
35882        6     50 36 17 22 23 29 33 39 34 37 37 37 39 43 48 5...  PrivateTest
35883        3  178 174 172 173 181 188 191 194 196 199 200 20...  PrivateTest
35884        0     17 17 16 23 28 22 19 17 25 26 20 24 31 19 27 9...  PrivateTest
35885        3     30 28 28 29 31 30 42 68 79 81 77 67 67 71 63 6...  PrivateTest
35886        2    19 13 14 12 13 16 21 33 50 57 71 84 97 108 122...  PrivateTest

35887 rows × 3 columns
```

**Figure 14: Reading From CSV File**



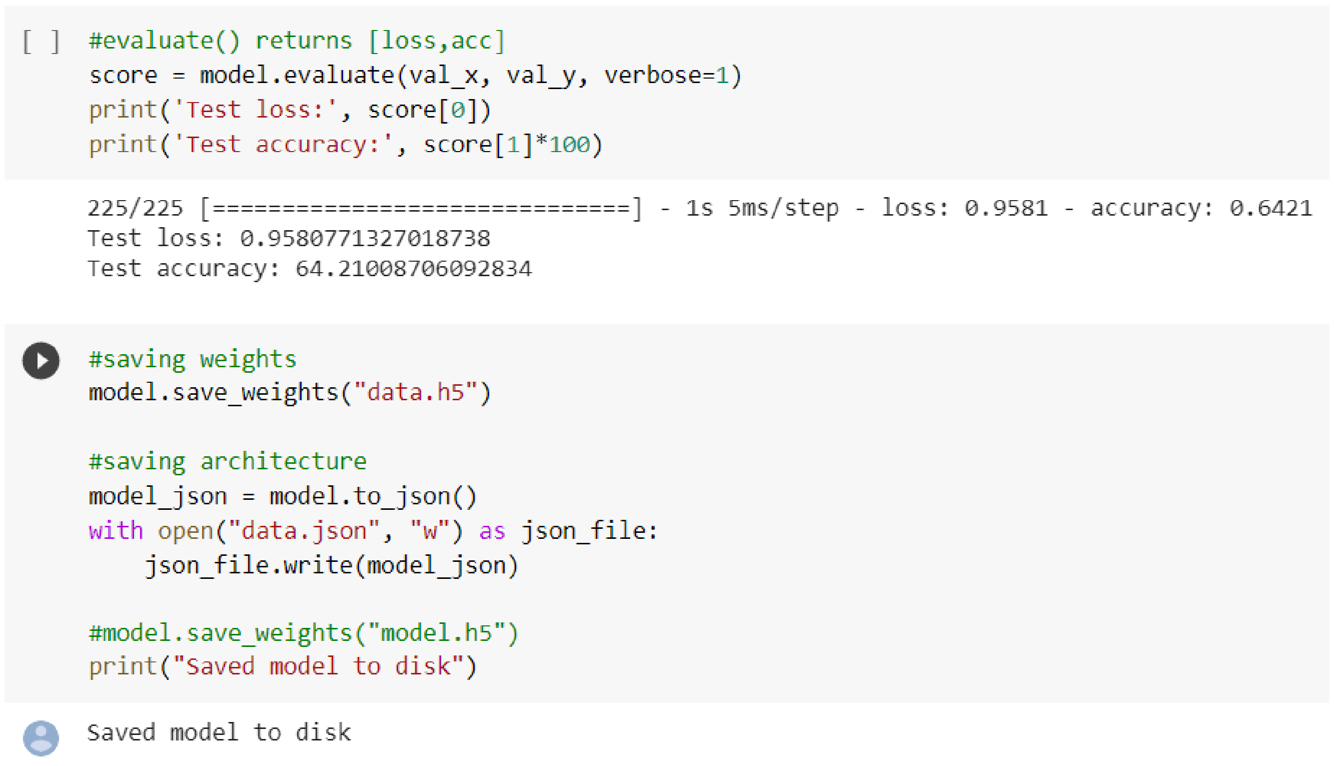
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 46, 46, 64)        640
_________________________________________________________________
batch_normalization (BatchNo (None, 46, 46, 64)        256
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 46, 46, 16)        50192
_________________________________________________________________
batch_normalization_1 (Batch (None, 46, 46, 16)        64
_________________________________________________________________
activation (Activation)      (None, 46, 46, 16)        0
_________________________________________________________________
average_pooling2d (AveragePo (None, 23, 23, 16)        0
_________________________________________________________________
dropout (Dropout)            (None, 23, 23, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 23, 23, 32)        12832
_________________________________________________________________
batch_normalization_2 (Batch (None, 23, 23, 32)        128
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 23, 23, 32)        25632
```

**Figure 15: Summary Of Model**



```
Epoch 00277: val_loss did not improve from 0.95515
Epoch 278/500
225/225 [==============================] - 16s 70ms/step - loss: 1.0148 - accuracy: 0.6136 - val_loss: 0.9583 - val_accuracy: 0.6418

Epoch 00278: val_loss did not improve from 0.95515
Epoch 279/500
225/225 [==============================] - 15s 69ms/step - loss: 1.0239 - accuracy: 0.6100 - val_loss: 0.9585 - val_accuracy: 0.6418

Epoch 00279: val_loss did not improve from 0.95515
Epoch 280/500
225/225 [==============================] - 16s 71ms/step - loss: 1.0119 - accuracy: 0.6172 - val_loss: 0.9585 - val_accuracy: 0.6421

Epoch 00280: val_loss did not improve from 0.95515
Epoch 281/500
225/225 [==============================] - 17s 75ms/step - loss: 1.0206 - accuracy: 0.6109 - val_loss: 0.9587 - val_accuracy: 0.6418

Epoch 00281: val_loss did not improve from 0.95515
Epoch 282/500
225/225 [==============================] - 15s 69ms/step - loss: 1.0180 - accuracy: 0.6134 - val_loss: 0.9584 - val_accuracy: 0.6420

Epoch 00282: val_loss did not improve from 0.95515
Epoch 283/500
225/225 [==============================] - 15s 69ms/step - loss: 1.0180 - accuracy: 0.6149 - val_loss: 0.9584 - val_accuracy: 0.6424

Epoch 00283: val_loss did not improve from 0.95515
Epoch 284/500
225/225 [==============================] - 16s 70ms/step - loss: 1.0200 - accuracy: 0.6138 - val_loss: 0.9581 - val_accuracy: 0.6421

Epoch 00284: val_loss did not improve from 0.95515

Epoch 00284: ReduceLROnPlateau reducing learning rate to 1.000000082740371e-08.
```

**Figure 16: Dataset Trained Successfully**



```
[ ] #evaluate() returns [loss,acc]
    score = model.evaluate(val_x, val_y, verbose=1)
    print('Test loss:', score[0])
    print('Test accuracy:', score[1]*100)

    225/225 [==============================] - 1s 5ms/step - loss: 0.9581 - accuracy: 0.6421
    Test loss: 0.9580771327018738
    Test accuracy: 64.21008706092834

    #saving weights
    model.save_weights("data.h5")

    #saving architecture
    model_json = model.to_json()
    with open("data.json", "w") as json_file:
        json_file.write(model_json)

    #model.save_weights("model.h5")
    print("Saved model to disk")

    Saved model to disk
```

**Figure 17: Accuracy Check Of The Model & Json Model Created Successfully**

# CHAPTER 5:

# CONCLUSIONS

In this case, when the model predicts incorrectly, the correct label is often the second most likely emotion. The facial expression recognition system presented in this research work contributes a resilient face recognition model based on the mapping of behavioral characteristics with the physiological biometric characteristics. The physiological characteristics of the human face with relevance to various expressions such as happiness, sadness, fear, anger, surprise, and disgust are associated with geometrical structures which are restored as base matching templates for the recognition system. The behavioral aspect of this system relates to the attitude behind different expressions as property bases. The property bases are alienated as exposed and hidden categories in genetic algorithmic genes. The gene training set evaluates the expressional uniqueness of individual faces and provides a resilient expressional recognition model in the field of biometric security. The design of a novel asymmetric cryptosystem based on biometrics having features like hierarchical group security eliminates the use of passwords and smart cards as opposed to earlier cryptosystems. It requires special hardware support like all other biometrics systems. This research work promises a new direction of research in the field of asymmetric biometric cryptosystems which is highly desirable in order to get rid of passwords and smart cards completely. Experimental analysis and study show that the hierarchical security structures are effective in geometric shape identification for physiological traits.

## FUTURE SCOPE

It is important to note that there is no specific formula to build a neural network that would guarantee to work well. Different problems would require different network architecture and a lot of trial and error to produce desirable validation accuracy. This is the reason why neural nets are often perceived as "black-box algorithms.". In this project, we got an accuracy of almost 64.2% which is not bad at all compared to all the previous models. But we need to improve in specific areas like-

- number and configuration of convolutional layers
- number and configuration of dense layers
- dropout percentage in dense layers

But due to the lack of a highly configured system, we could not go deeper into a dense neural network as the system gets very slow and we will try to improve in these areas in the future. We would also like to train more databases into the system to make the model more and more accurate but again resources become a hindrance in the path and we also need to improve in several areas in the future to resolve the errors and improve the accuracy. Having examined techniques to cope with expression variation, in the future it may be investigated in more depth about the face classification problem and optimal fusion of color and depth information. Further study can be laid down in the direction of the allele of gene matching to the geometric factors of the facial expressions. The genetic property evaluation framework for facial expression systems can be studied to suit the requirement of different security models such as criminal detection, governmental confidential security breaches, etc.

## REFERENCES

1. A literature survey on Facial Expression Recognition using Global Features by Vaibhav Kumar, J. Mistry and Mahesh M. Goyani, International Journal of Engineering and Advanced Technology (IJEAT), April 2013

[http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.645.5162&rep=rep1&type=pdf]

2. Convolutional Neural Networks (CNN) With TensorFlow by Sourav from Edureka [https://www.youtube.com/watch?v=umGJ30-15_A]

3. Deep Learning Simplified by Sourav from Edureka [https://www.youtube.com/watch?v=dafuAz_CV7Q&list=PL9ooVrP1hQOEX8BKDplfG86ky8s7Oxbzg]

4. "Robust Real-Time Face Detection", International Journal of Computer Vision 57(2), 137–154, 2004.

5. VIOLA JONES FACE DETECTION EXPLANATION by Rahul Patil.

6.Project_Report_On_FACIAL_EXPRESSION_RECOGNITION_USING_IMAGE_PROCESSING

7. Facial Expression Detection Techniques: Based on Viola and Jones algorithm and Principal Component Analysis by Samiksha Agrawal and Pallavi Khatri, ITM University Gwalior(M.P.), India, 2014.

8. Online publication on Convolutional Neural Networks: An Intro Tutorial by Derrick Mwiti.

[https://heartbeat.fritz.ai/a-beginners-guide-to-convolutional-neural-networks-cnn-cf26c5ee17ed]