Facial Expression detection of Emotions using Convolutional Neural Network

Submitted in the partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Submitted by:

NIKHIL KUMAR

20BCS6845

Under the Supervision of:

SHUBHANGI MISHRA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

APEX INSTITUE OF TECHNOLOGY

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413,

PUNJAB

November. 2022

DECLARATION

I, **'Nikhil Kumar'**, student of **'Bachelor of Engineering in Artificial Intelligence and Machine Learning'**, **session: 2020-2024**, Department of Computer Science and Engineering, Apex Institute of Technology, Chandigarh University, Punjab, hereby declare that the work presented in this Research Project Work entitled '**Facial Expression Detection of Emotions using Convolutional Neural Network'** is the outcome of our own bonafide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Date: 11-11-2022
Place: GHAURAN, PUNJAB

(NIKHIL KUMAR)
UID: 20BCS6845

BONAFIDE CERTIFICATE

Certified that this project report "FACIAL EXPRESSION OF EMOTIONS" is the bonafide work of "**NIKHIL KUMAR, MARAM LEELA KRISHNA SUBRAHMANYAM, ARYA CHAKRABORTY & KOPPULA PRAKASH**" who carried out the project work under my/our supervision.

<<Signature of the HoD>>

SIGNATURE

<<Name of the Head of the Department>>

HEAD OF THE DEPARTMENT

<<Department>>

<<Signature of the Supervisor>>

SIGNATURE

<<Name>>

SUPERVISOR

<<Academic Designation>>

<<Department>>

Submitted for the project viva-voce examination held on

INTERNAL EXAMINER

EXTERNAL EXAMI

TABLE OF CONTENTS

Abstract

Real time Facial Expression detection for Emotions may be a huge, spirited and sophisticated space of computer vision. If there may be a single object to be detected in a picture, it's called Image Localization and if there are multiple objects in a picture, then it's Object(Face) Detection. This detects the semantic objects of a category in digital images and videos. The applications of real time Facial Expression detection for Emotions are analyse customers' emotions, driver fatigur detection, monitor student's attention, detection of political attitudes,etc . Convolution Neural Networks may be a representative tool of Deep learning to find objects victimisation OpenCV(Opensource computer Vision), that may be a library of programming functions mainly geared toward realtime computer vision.

Keywords: Computer vision, Deep Learning, Convolution Neural Networks.

Acknowledgement

It gives us immense pleasure to express our deepest sense of gratitude and sincere thanks to our respected guide Amit Garg (Assistant Professor), CSE- Artificial Intelligence, Chandigarh University, Mohali for his valuable guidance, encouragement and help for completing this work. His useful suggestions for this whole work and co operative behaviour are sincerely acknowledged. We are also grateful to Dr. Shikha Gupta (Program Leader, CSE-AIML) for her constant support and guidance.

We also wish to express our indebtedness to our family members whose blessings and support always helped us to face the challenges ahead. We also wish to express thanks to all people who helped us in completion of this project.

NIKHIL KUMAR (20BCS6845)
MARAM LEELA KRISHNA SUBRAHMANYAM (21BCS8803)
ARYA CHAKRABORTY (21BCS8814)
KOPPULA PRAKASH (21BCS8824)

PLACE: GHARUAN, MOHALI
DATE: 11-11-2022

List of Tables

List of Figures

Chapter 1
INTRODUCTION

Introduction

We'll be using Open CV, an open source library for computer vision, written in C/C++, that has interfaces in C++, Python and Java. It supports Windows, Linux, Mac OS, iOS and Android. Some of our work will also require using D lib, a modern C++ toolkit containing machine learning algorithms and tools for creating complex software.

Problem definition

Human emotions and intentions are expressed through facial expressions and deriving an efficient and effective feature is the fundamental component of facial expression system. Facial expressions convey non-verbal cues, which play an important role in interpersonal relations. Automatic recognition of facial expressions can be an important component of natural human-machine interfaces; it may also be used in behavioral science and in clinical practice. An automatic Facial Expression Recognition system needs to solve the following problems: detection and location of faces in a cluttered scene, facial feature extraction, and facial expression classification.

Open cv Python program for Face Detection:

The objective of the program given is to detect object of interest (face) in real time and to keep tracking of the same object. This is a simple example of how to detect face in Python. You can try to use training samples of any other object of your choice to be detected by training the

classifier on required objects.

OpenCV:

OpenCV is the most popular library for computer vision. Originally written in C/C++, it now provides bindings for Python.

OpenCV uses machine learning algorithms to search for faces within a picture. Because faces are so complicated, there isn't one simple test that will tell you if it found a face or not. Instead, there are thousands of small patterns and features that must be matched. The algorithms break the task of identifying the face into thousands of smaller, bite-sized tasks, each of which is easy to solve. These tasks are also called classifiers.

For something like a face, you might have 6,000 or more classifiers, all of which must match for a face to be detected (within error limits, of course). But therein lies the problem: for face detection, the algorithm starts at the top left of a picture and moves down across small blocks of data, looking at each block, constantly asking, "Is this a face? … Is this a face? … Is this a face?" Since there are 6,000 or more tests per block, you might have millions of calculations to do, which will grind your computer to a halt.

To get around this, OpenCV uses cascades. What's a cascade? The best answer can be found in the dictionary: "a waterfall or series of waterfalls."

Like a series of waterfalls, the OpenCV cascade breaks the problem of detecting faces into multiple stages. For each block, it does a very rough and quick test. If that passes, it does a slightly more detailed test, and so on. The algorithm may have 30 to 50 of these stages or cascades, and it will only detect a face if all stages pass.

The advantage is that the majority of the picture will return a negative during the first few stages, which means the algorithm won't waste time testing all 6,000 features on it. Instead of taking hours, face detection can now be done in real time.

Haar Feature Selection:

There are some common features that we find on most common human faces :

- a dark eye region compared to upper-cheeks
- a bright nose bridge region compared to the eyes
- some specific location of eyes, mouth, nose…

The characteristics are called Haar Features. In this example, the first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature value is simply computed by summing the pixels in the black area and subtracting the pixels in the white area.

The integral image

Computing the rectangle features in a convolutional kernel style can be long, very long. For this reason, the authors, Viola and Jones, proposed an intermediate representation for the image: the integral image. The role of the integral image is to allow any rectangular sum to be computed simply, using only four values. We'll see how it works!

Cascades in Practice

Though the theory may sound complicated, in practice it is quite easy. The cascades themselves are just a bunch of XML files that contain OpenCV data used to detect objects. You initialize your code with the cascade you want, and then it does the work for you.

Since face detection is such a common case, OpenCV comes with a number of built-in cascades for detecting everything from faces to eyes to hands to legs. There are even cascades for non-human things. For example, if you run a banana shop and want to track people stealing bananas, this guy has built one for that!

Cascading Classifier

Although the process described above is quite efficient, a major issue remains. In an image, most of the image is a non-face region. Giving equal importance to each region of the image makes no sense, since we should mainly focus on the regions that are most likely to contain a picture. Viola and Jones achieved an increased detection rate while reducing computation time using Cascading Classifiers.

Requirements:

The first step is to install OpenCV, and Dlib. Run the following command:

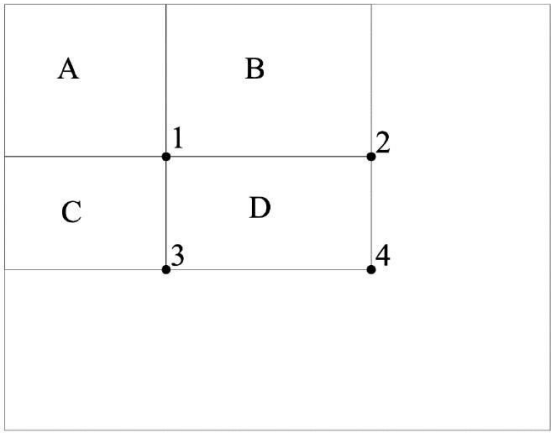
```
pip install opencv-python

pip install dlib
```

Imports and models path

We'll create a new Jupyter notebook / python file and start off with :

```
import cv2
import matplotlib.pyplot as plt
import dlib
from imutils import face_utils

font = cv2.FONT_HERSHEY_SIMPLEX
```

The integral image



One should simply be aware that rectangles are quite simple features in practice, but sufficient for face detection. Steerable filters tend to be more flexible when it comes to complex problem.

Imports

The next step simply is to locate the pre-trained weights. We will be using default pre-trained models to detect face, eyes and mouth. Depending on your version of Python, the files should be located somewhere over here :

```
/usr/local/lib/python3.7/site-packages/cv2/data
```

Once identified, we'll declare Cascade classifiers this way

```
cascPath = "/usr/local/lib/python3.7/site-
packages/cv2/data/haarcascade_frontalface_default.xml"
eyePath = "/usr/local/lib/python3.7/site-
packages/cv2/data/haarcascade_eye.xml"
smilePath = "/usr/local/lib/python3.7/site-
packages/cv2/data/haarcascade_smile.xml"

faceCascade = cv2.CascadeClassifier(cascPath)
eyeCascade = cv2.CascadeClassifier(eyePath)
smileCascade = cv2.CascadeClassifier(smilePath)
```

Detect face on an image

Before implementing the real time face detection algorithm, let's try a simple version on an image. We can start by loading a test image

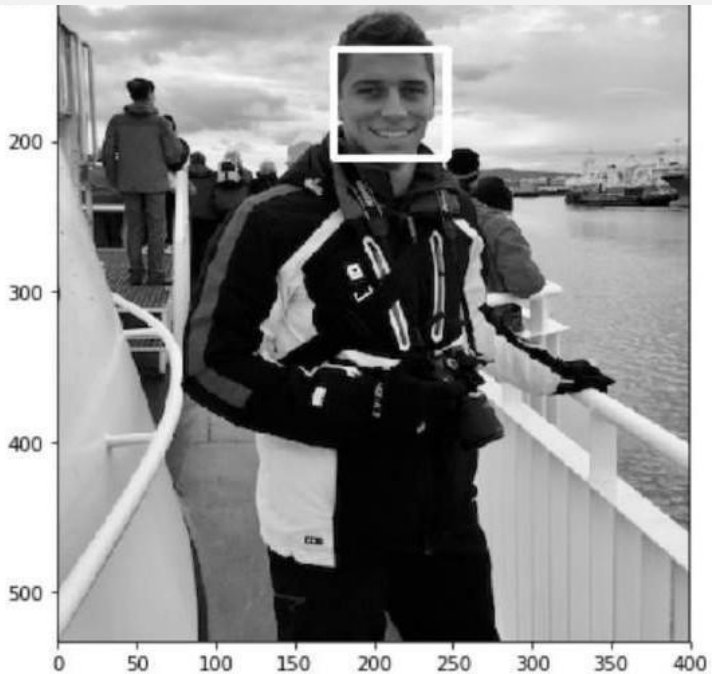
```
# Load the image
gray = cv2.imread('face_detect_test.jpeg', 0)


plt.figure(figsize=(12,8))
plt.imshow(gray, cmap='gray')
plt.show()
```

Then, we detect the face and we add a rectangle around it:

```
# Detect faces
faces = faceCascade.detectMultiScale(
gray,
scaleFactor=1.1,
minNeighbors=5,
flags=cv2.CASCADE_SCALE_IMAGE
)


# For each face
for (x, y, w, h) in faces:
    # Draw rectangle around the face
    cv2.rectangle(gray, (x, y), (x+w, y+h), (255, 255, 255), 3)
```

Finally, display the result :

```
plt.figure(figsize=(12,8))
plt.imshow(gray, cmap='gray')
plt.show()
```



Chapter - 2

Literature Survey

In recent years, facial emotion recognition has become a hot focus of research. To identify emotion from faces, most people utilize computer vision, machine learning, or deep learning technologies. This study [1] gives a brief overview of FER research done over the last few decades. The traditional FER techniques are presented first, followed by a description of the typical FER system types and their major algorithms. The authors next describe deep-learning-based FER methods that use deep networks to enable "end-to-end" learning. This paper also looks at a new hybrid deeplearning technique that employs a convolutional neural network (CNN) for spatial characteristics of a single frame and a long short-term memory (LSTM) for temporal data of several frames. A brief overview of publicly accessible evaluation metrics is provided in the latter half of this work, as well as a comparison with benchmark findings, which constitute a standard for a quantitative comparison of FER investigations. Instead of minimizing the crossentropy loss, learning reduces a margin-based loss. Study of multi-level features in a convolutional neural network for facial emotion identification by Hai-Duong Nguyen [2]. They offer a model based on the data that purposely combines a hierarchy of characteristics to better the categorization job. The model was tested on the FER2013 dataset and found to be similar to existing state-of-the-art approaches in terms of performance. Using a feedforward learning model, the authors in [3] developed an instructor's face expression recognition technique within a classroom. For successful high-level feature extraction, the face is first recognized from the obtained lecture videos and important frames are picked, removing all unnecessary frames. Then, using several convolution neural networks and parameter tweaking, deep features are retrieved and supplied to a classifier.

2.1 Existing System

Previous works are focused on eliciting results from unimodal systems. Machines used to predict emotion by only facial expressions or only vocal sounds . After a while, multimodal systems that use more than one features to predict emotion has more effective and gives more accurate results. So that, the combination of features such as audio-visual expressions, EEG, body gestures have been used since. More than one intelligent machine and neural networks are used to implement the emotion recognition system. Multimodal recognition method has proven more effective than unimodal systems by Shiqing et al. [1]. Research has demonstrated that deep neural networks can effectively generate discriminative features that approximate the complex non-linear dependencies between features in the original set. These deep generative models have been applied to speech and language processing, as well as emotion recognition tasks . Martin et al. [2] showed that bidirectional Long Short Term Memory(BLSTM) network is more effective that conventional SVM approach.; In speech processing, Ngiam et al. [3] proposed and evaluated deep networks to learn audio-visual features from spoken letters. In emotion recognition, Brueckner et al. [4] found that the use of a Restricted Boltzmann Machine (RBM) prior to a two-layer neural network with fine-tuning could significantly improve classification accuracy in the Interspeech automatic likability classification challenge. The work by Stuhlsatz et al. [5] took a different approach for learning acoustic features in speech emotion recognition using Generalized Discriminant Analysis (GerDA) based on Deep Neural Networks (DNNs).

Yelin et al. [6] showed three layered Deep Belief Networks(DBNs') give better performance than two layered DBNs' by using audio visual emotion recognition process.

Samira et al [7] used Recurrent neural network combined with Convoluted Neural Network(CNN) in an underlying CNN-RNN architecture to predict emotion in the video.Some noble methods and techniques also enriched this particular research. They are more accurate, stable and realistic. In terms of performance, accuracy, reasonability and precision these methods are the dominating solutions. Some of them are more accurate but some are more realistic. Some take much time and require greater computation power to produce the more accurate result but some compromises accuracy over performance. The idea of being successful might differ but these solutions are the best possible till now.

Yelin Kim and Emily Mower Provos [8] explore whether a subset of an utterance can be used for emotion inference and how the subset varies by classes of emotion and modalities. They propose a windowing method that identifies window configurations, window duration, and timing, for aggregating segment-level information for utterance-level emotion inference. The experimental results using the IEMOCAP and MSP IMPROV datasets show that the identified temporal window configurations demonstrate consistent patterns across speakers, specific to different classes of emotion and modalities. They compare their proposed windowing method to a baseline method that randomly selects window configurations and a traditional all-mean method that uses the full information within an utterance. This method shows a significantly higher performance in emotion recognition while the method only uses 40–80% of information within each utterance. The identified windows also show consistency across speakers, demonstrating how multimodal cues reveal emotion over time. These patterns also align with psychological findings. But after all achievement, the result is not consistent with this method.

A. Yao, D. Cai, P. Hu, S. Wang, L. Shan, and Y. Chen [9] used a well-designed Convolutional Neural Network (CNN) architecture regarding the video based emotion recognition . They proposed the method named as HOLONET has three critical considerations in network design. To reduce redundant filters and enhance the non-saturated non-linearity in the lower convolutional layers, they used modified Concatenated Rectified Linear Unit (CReLU) instead of ReLU.To enjoy the accuracy gain from considerably increased network depth and maintain efficiency, they combine residual structure and CReLU to construct the middle layers.To broaden network width and introduce multi-scale feature extraction property, the topper layers are designed as a variant of the inception-residual structure. This method more realistic than other methods here. It's focused on adaptability in real-time scenario than accuracy and theoretical performance. Though its accuracy is also impressive but only this method is applicable only in the video based emotion recognition. Other types of data rather than video, this method can't produce results .

Y. Fan, X. Lu, D. Li, and Y. Liu.[10] proposed a method for video-based emotion recognition in the wild. They used CNN-LSTM and C3D networks to simultaneously model video appearances and motions
. They found that the combination of the two kinds of networks can give impressive results, which demonstrated the effectiveness of the method. In their proposed method they used LSTM (Long Short Term Memory) - a special kind of RNN, C3D – A DirectSpatio-Temporal Model and Hybrid CNN-RNN and C3D Networks. This

method gives a great accuracy and performance is remarkable. But this method is much convoluted, time-consuming and less realistic. For this reason, efficiency is not that impressive .

Zixing Zhang, Fabien Ringeval, Eduardo Coutinho, Erik Marchi and Björn Schüller [11] proposed some improvement in SSL technique to improve the low performance of a classifier that can deliver on challenging recognition tasks reduces the trust ability of the automatically labeled data and gave solutions regarding the noise accumulation problem - instances that are misclassified by the system are still used to train it in future iterations. They exploited the complementarity between audio-visual features to improve the performance of the classifier during the supervised phase. Then, they iteratively re-evaluated the automatically labeled instances to correct possibly mislabeled data and this enhances the overall confidence of the system's predictions. This technique gives a best possible performance using SSL technique where labeled data is scarce and/or expensive to obtain but still, there are various inherent limitations that limit its performance in practical applications. This technique has been tested on a specific database with a limited type and number of data. The algorithm which has been used is not capable of processing physiological data alongside other types of data.

Wei-Long Zheng and Bao-Liang Lu proposed [12] EEG-based effective models without labeled target data using transfer learning techniques (TCA-based Subject Transfer) [18] which is very accurate in terms of positive emotion recognition than other techniques used before. Their method achieved 85.01% accuracy. They used to transfer learning and their method includes three pillars, TCA-based Subject Transfer, KPCA-based Subject Transfer and Transductive Parameter Transfer. For data preprocessing they used raw EEG signals processed with a bandpass filter between 1 Hz and 75 Hz and for feature extraction, they employed differential entropy (DE) features. For evaluation, they adopted a leave-one subject-out cross-validation method. Their experimental results demonstrated that the transductive parameter transfer approach significantly outperforms the other approaches in terms of the accuracies, and a 19.58% increase in recognition accuracy has been achieved.

Though this achievement is limited to the positive emotion recognition only. This method is limited in terms of negative and neutral emotion recognition. Yet a lot improvement needed to recognize negative and neutral emotion more accurately.

Table 1: Emotion recognition different approach and successes

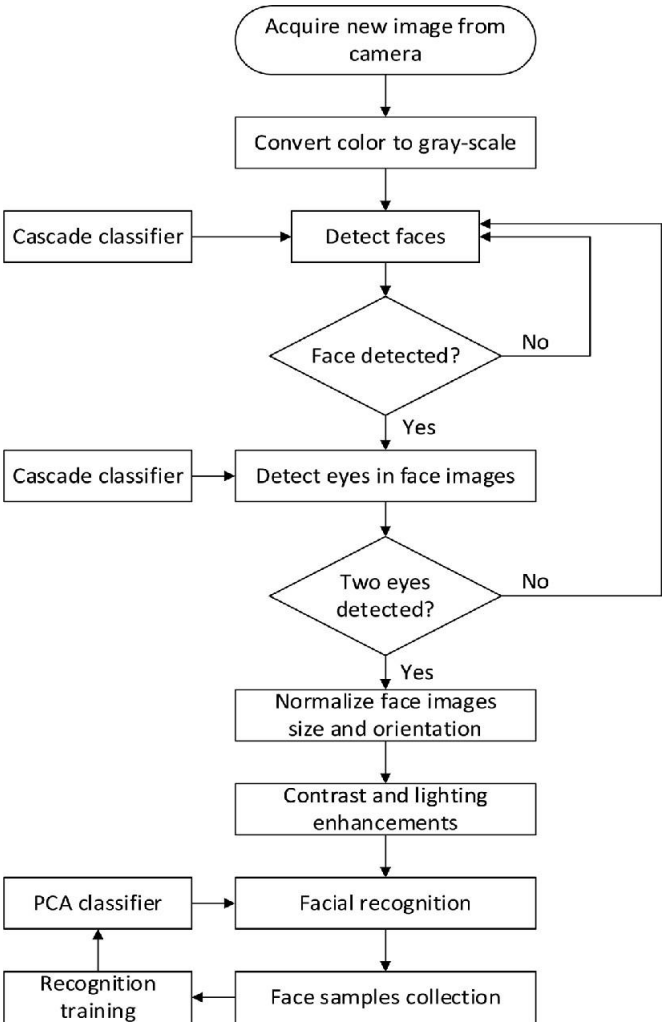
Reference and year	Approach and Method	Performance
Wei-Long Zheng and Bao-Liang Lu (2016)	EEG-based affective models without labeled target data using transfer learning techniques (TCA-based Subject Transfer)	Positive (85.01%) emotion recognition rate is higher than other approaches but neutral (25.76%) and negative (10.24%) emotions are often confused with each other.
Zixing Zhang, Fabien Ringeval, Fabien Ringeval, Eduardo Coutinho, Erik Marchi and Björn Schüller (2016)	Semi-Supervised Learning (SSL) technique	Delivers a strong performance in the classification of high/low emotional arousal (UAR = 76.5%), and significantly outperforms traditional SSL methods by at least 5.0% (absolute gain).
Y. Fan, X. Lu, D. Li, and Y. Liu. (2016)	Video-based Emotion Recognition Using CNN-RNN and C3D Hybrid Networks	Achieved accuracy 59.02% (without using any additional Emotion labeled video clips in training set) which is the best till now.
A. Yao, D. Cai, P. Hu, S. Wang, L. Shan and Y. Chen (2016)	HoloNet: towards robust emotion recognition in the wild	Achieved mean recognition rate of 57.84%.
Yelin Kim and Emily Mower Provos (2016)	Data driven framework to explore patterns (timings and durations) of emotion evidence, specific to individual emotion classes	Achieved 65.60% UW accuracy, 1.90% higher than the baseline.

2.2 Proposed System

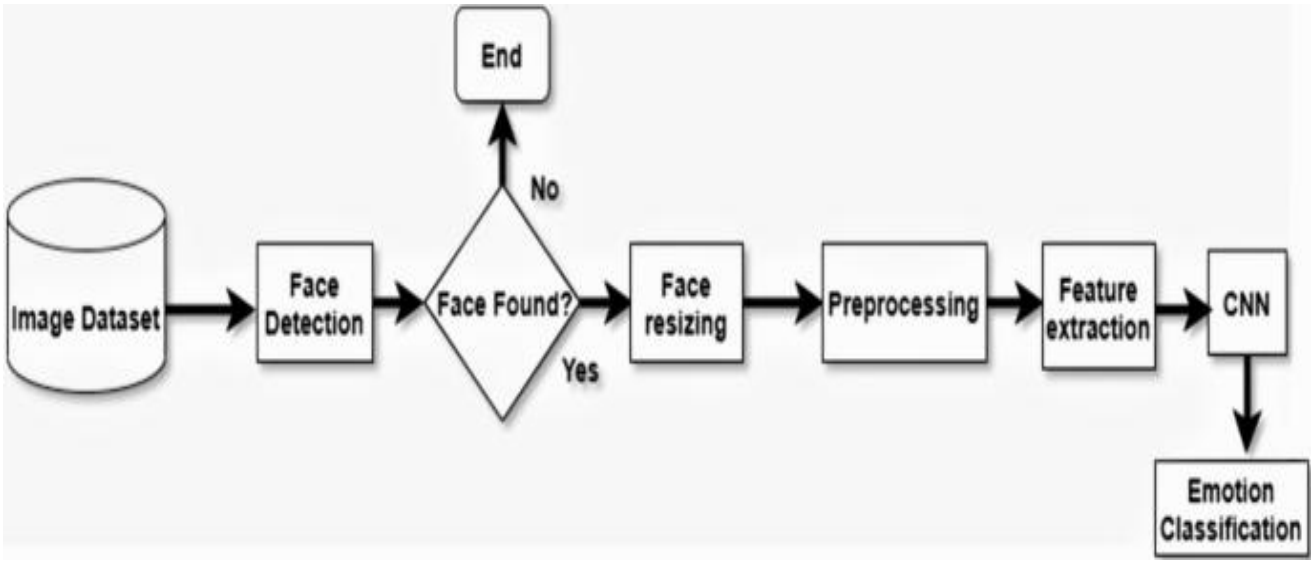
In this project, we approach the problem by taking deep-learning method of Convolutional Neural Networks (CNNs), which integrates the step of handcrafted feature extraction with training of classifier. This system is able to achieve the relatively most optimal solution through the process of backpropagation in which the algorithm learns the weights through modified stochastic gradient descent that can find the directions that best minimize the loss from the ground truth. The numerical result of the algorithm will show a probabilistic result of each labeled class. In order to reduce computational expense, the technique of fine-tuning is applied so that a pre-trained model can adapt the variance of our local dataset with benefit of reducing computational expense. As results, such method best resolves the issues of lighting variations and different orientation of object in the image and thus achieves a higher accuracy.

Chapter - 3

Design flow & Process



- *Above, the implementation of face detection.*



- *Above is the implementation of Emtion detection*

Techniques and tools:

We used Python syntax for this project. As a framework we used Keras, which is a high-level neural network API written in Python. But Keras can't work by itself, it needs a backend for low-level operations. Thus, we installed a dedicated software library — Google's TensorFlow**.** OpenCV was designed for computational efficiency, with a strong focus on real-time applications. So, it is perfect for real-time face recognition using a camera. We are using webcam for real-time detection otherwise if webcam is not used then a video will be used in that place.

As a development environment I used the PyCharm and Anaconda for base. I used Matplotlib for visualization.

The three main phases ofthe project will be:

To create a complete project on Face Recognition, we must work on 3 very distinct phases:

- Face Detection and Data Gathering
- Train the Recognizer
- Face Recognition

Theory:

Making use of the whole frontal face image and processing it in order to end up with the classifications of 6 universal facial expression prototypes: disgust, fear, joy, surprise, sad ness and anger; outlines the first approach. Here, it is assumed that each of the above-mentioned emotions have characteristic expressions on face and that's why recognition of them is necessary and sufficient. As expression is more related with subtle changes of some discrete features such as eyes, eyebrows and lip corners; these fine-grained changes are used for analyzing automated recognition.

There are two main methods that are used in both of the above explained approaches.

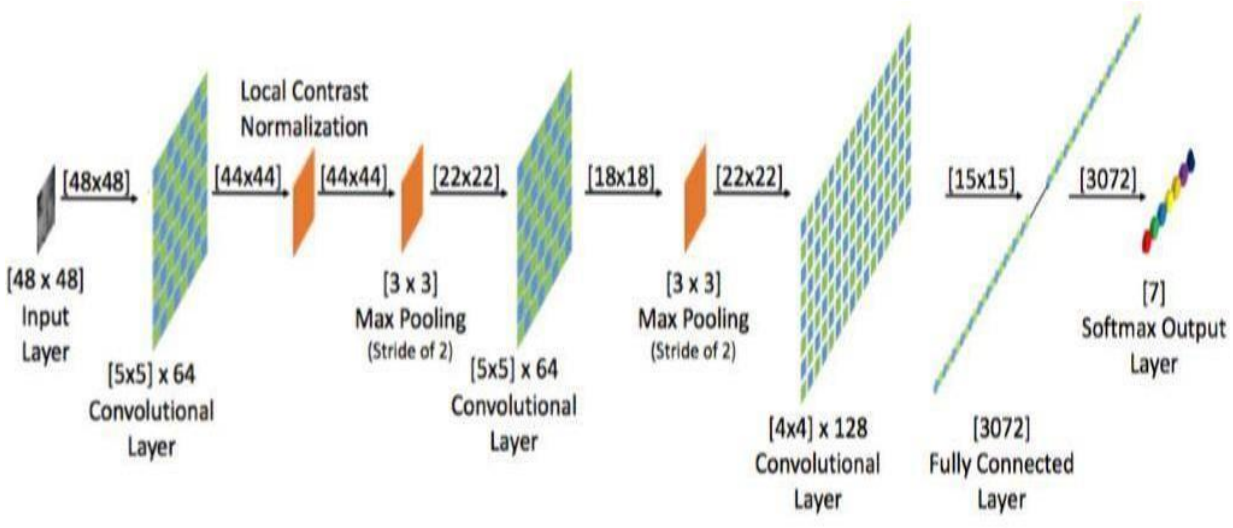
Geometric Based Parameterization is an old way which consists of tracking and processing the motions of some spots on image sequences, The disadvantages of this method are the contours of these features and components have to be adjusted manually in this frame, the problems of robustness and difficulties come out in cases of pose and illumination changes while the tracking is applied on images, as actions & expressions tend to change both in morphological and in dynamical senses, it becomes hard to estimate general parameters for movement and displacement. Therefore, ending up with robust decisions for facial actions under these varying conditions becomes to be difficult.

Rather than tracking spatial points and using positioning and movement parameters that vary within time, color (pixel) information of related regions of face are processed in Appearance Based Parameterizations; in order to obtain the parameters that are going to form the feature vectors .Different features such as Gabor, Haar wavelet coefficients, together with feature extraction and selection methods such as PCA, LDA, and Ad boost are used within this framework.

For classification problem, algorithms like Machine learning, Neural Network, Support Vector Machine, Deep learning, Naive Bayes are used.

Methodology:

The facial expression recognition system is implemented using convolutional neural network.

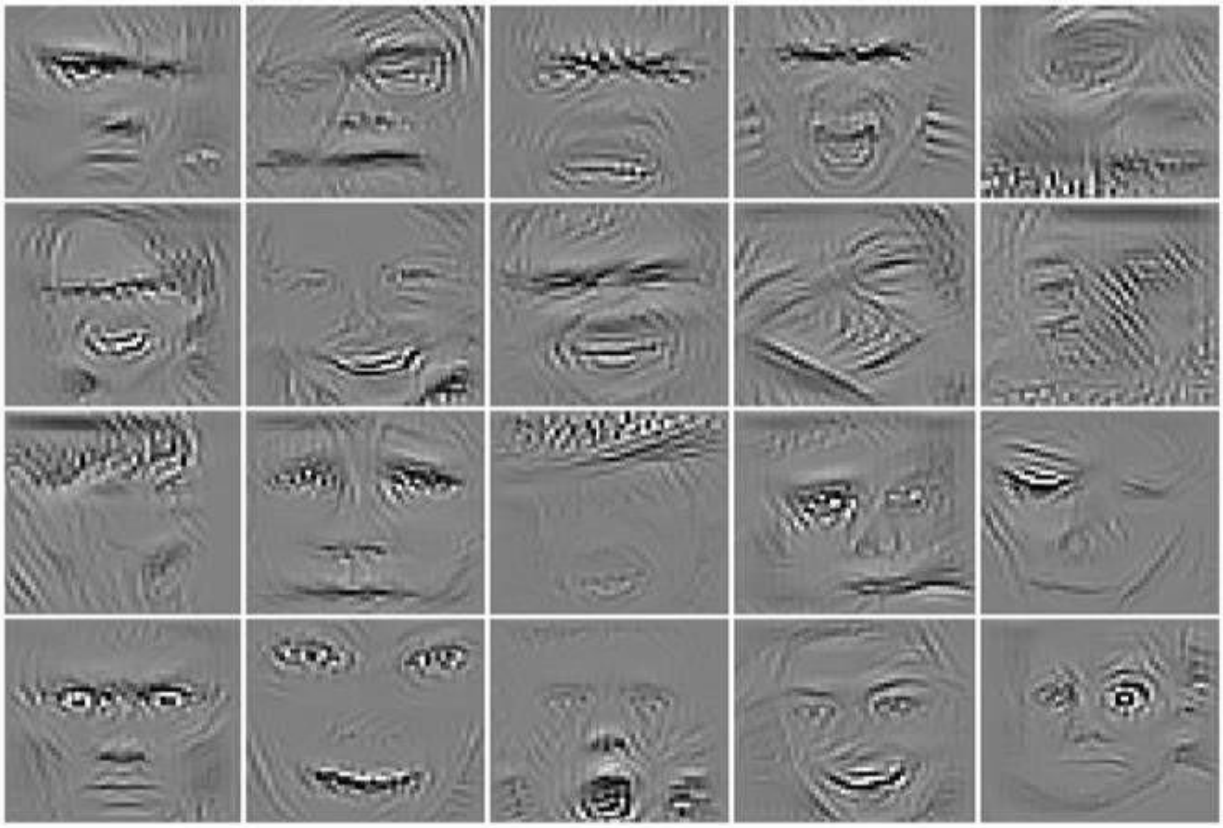


During training, the system received a training data comprising grayscale images of faces with their respective expression label and learns a set of weights for the network. The training step took as input an image with a face. Thereafter, an intensity normalization is applied to the image. The normalized images are used to train the Convolutional Network. To ensure that the training performance is not affected by the order of presentation of the examples, validation dataset is used to choose the final best set of weights out of a set of trainings performed with samples presented in different orders. The output of the training step is a set of weights that achieve the best result with the training data. During test, the system received a grayscale image of a face from test dataset, and output the predicted expression by using the final network weights learned during training. Its output is a single number that represents one of the seven basic expressions.

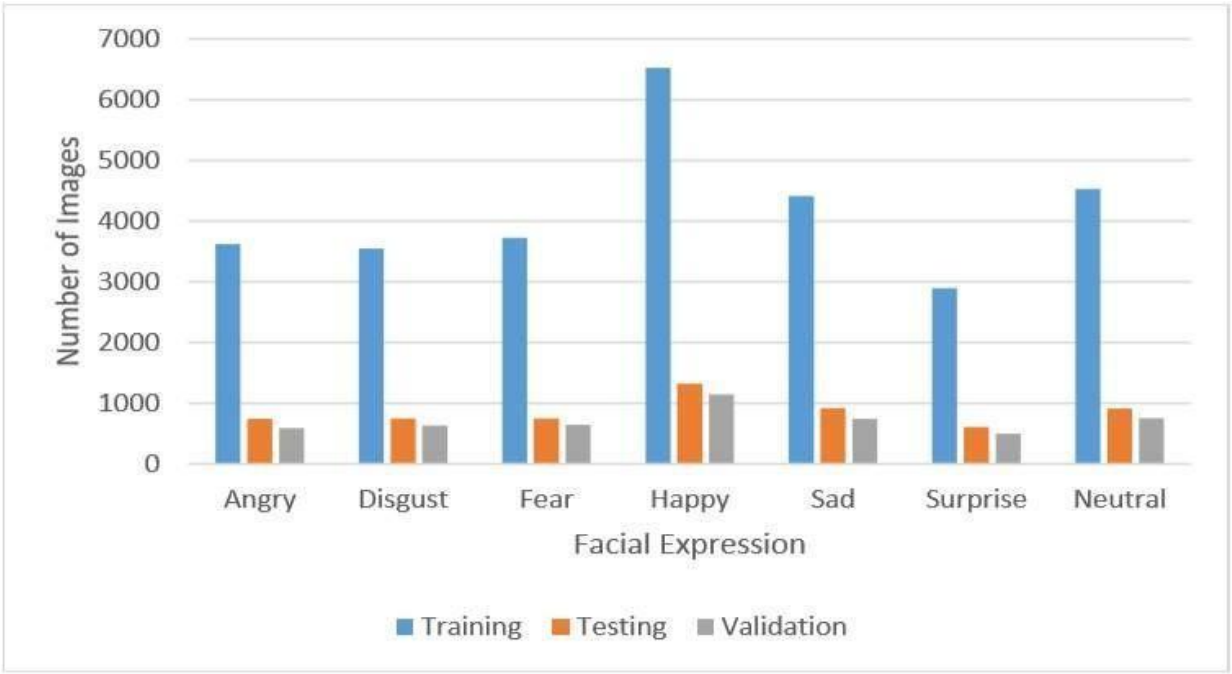
Dataset:

The dataset from a Kaggle Facial Expression Recognition Challenge (FER2013) is used for the training and testing. It comprises 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. Dataset has training set of 35887 facial images with facial expression labels.. The dataset has class imbalance issue, since some classes have large number of examples while some has few. The dataset is balanced using oversampling, by increasing numbers in minority classes. The balanced dataset contains 40263 images, from which 29263 images are used for training, 6000 images are used for testing, and 5000 images are used for validation.

(a) Samples from the FER-2013 dataset



(b) Guided back-propagation visualization of our sequential fully CNN.



Training, Testing and Validation Data distribution

Convolutional neural networks and image classification:

A typical architecture of a convolutional neural network contains an input layer, some convolutional layers, some fully-connected layers, and an output layer.

Convolutional neural networks (CNN) is a special architecture of artificial neural networks, proposed by Yann LeCun in 1988. CNN uses some features of the visual cortex. One of the most popular uses of this architecture is image classification.

Instead of the image, the computer sees an array of pixels. For example, if image size is 300 x 300. In this case, the size of the array will be 300x300x3. Where 300 is width, next 300 is height and 3 is RGB channel values. The computer is assigned a value from 0 to 255 to each of these numbers. This value describes the intensity of the pixel at eachpoint.

In more detail: the image is passed through a series of convolutional, nonlinear, pooling layers and fully connected layers, and then generates the output.

Process:

The detailed steps involved for the completion of project involve the basis of machine learning.

1. For reading data and converting it into arrays using numpy:

   File name: pre-processor.py

```
import numpy as np
from imageio import imread


def preprocess_input(x, v2=True):
    x = x.astype('float32')
    x = x / 255.0
    if v2:
        x = x - 0.
        x = x * 2.0
    return x

def _imread(image_name):
        return imread(image_name)

def _imresize(image_array, size):
    return imresize(image_array, size)

def to_categorical(integer_classes, num_classes=2):
    integer_classes = np.asarray(integer_classes, dtype='int')
    num_samples = integer_classes.shape[0]
    categorical = np.zeros((num_samples, num_classes))
    categorical[np.arange(num_samples), integer_classes] = 1
    return categorical
```

2. Loading the dataset:

In this we are creating program that can use two types of datasets: fer2013 dataset or imdb gender classification dataset. It is handy in either using of these two types of datasets.

File name: datasets.py

```python
from scipy.io import loadmat
import pandas as pd
import numpy as np
from random import shuffle
import cv2

class DataManager(object):
    def __init__(self, dataset_name='imdb', dataset_path=None, image_size=(48, 48)):

        self.dataset_name = dataset_name
        self.dataset_path = dataset_path
        self.image_size = image_size
        if self.dataset_path != None:
            self.dataset_path = dataset_path
        elif self.dataset_name == 'imdb':
            self.dataset_path = '../datasets/imdb_crop/imdb.mat'
        elif self.dataset_name == 'fer2013':
            self.dataset_path = '../datasets/fer2013/fer2013.csv'
        else:
            raise Exception('Incorrect dataset name, please input imdb or fer2013')

    def get_data(self):
        if self.dataset_name == 'imdb':
            ground_truth_data = self._load_imdb()
        elif self.dataset_name == 'fer2013':
            ground_truth_data = self._load_fer2013()
        return ground_truth_data
```

a) Loading imdb dataset;

```
def _load_imdb(self):
    face_score_treshold = 3
    dataset = loadmat(self.dataset_path)
    image_names_array = dataset['imdb']['full_path'][0, 0][0]
    gender_classes = dataset['imdb']['gender'][0, 0][0]
    face_score = dataset['imdb']['face_score'][0, 0][0]
    second_face_score = dataset['imdb']['second_face_score'][0, 0][0]
    face_score_mask = face_score > face_score_treshold
    second_face_score_mask = np.isnan(second_face_score)
    unknown_gender_mask = np.logical_not(np.isnan(gender_classes))
    mask = np.logical_and(face_score_mask, second_face_score_mask)
    mask = np.logical_and(mask, unknown_gender_mask)
    image_names_array = image_names_array[mask]
    gender_classes = gender_classes[mask].tolist()
    image_names = []
    for image_name_arg in range(image_names_array.shape[0]):
        image_name = image_names_array[image_name_arg][0]
        image_names.append(image_name)
    return dict(zip(image_names, gender_classes))
```

b) Loading fer2013 dataset;

```
def _load_fer2013(self):
    data = pd.read_csv(self.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'), self.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
```

3. Training the dataset:

```
def get_class_to_arg(dataset_name='fer2013'):
    if dataset_name == 'fer2013':
        return {'angry':0, 'disgust':1, 'fear':2, 'happy':3, 'sad':4,
                'surprise':5, 'neutral':6}
    elif dataset_name == 'imdb':
        return {'woman':0, 'man':1}
    else:
        raise Exception('Invalid dataset name')

def split_imdb_data(ground_truth_data, validation_split=.2, do_shuffle=False):
    ground_truth_keys = sorted(ground_truth_data.keys())
    if do_shuffle == True:
        shuffle(ground_truth_keys)
    training_split = 1 - validation_split
    num_train = int(training_split * len(ground_truth_keys))
    train_keys = ground_truth_keys[:num_train]
    validation_keys = ground_truth_keys[num_train:]
    return train_keys, validation_keys

def split_data(x, y, validation_split=.2):
    num_samples = len(x)
    num_train_samples = int((1 - validation_split)*num_samples)
    train_x = x[:num_train_samples]
    train_y = y[:num_train_samples]
    val_x = x[num_train_samples:]
    val_y = y[num_train_samples:]
    train_data = (train_x, train_y)
    val_data = (val_x, val_y)
    return train_data, val_data
```

For extra work like setting rectangular box and displaying of emotion nameon the output window:

File name: inference.py

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from keras.preprocessing import image

def load_image(image_path, grayscale=False, target_size=None):
    pil_image = image.load_img(image_path, grayscale, target_size)
    return image.img_to_array(pil_image)

def load_detection_model(model_path):
    detection_model = cv2.CascadeClassifier(model_path)
    return detection_model

def detect_faces(detection_model, gray_image_array):
    return detection_model.detectMultiScale(gray_image_array, 1.3, 5)
```

a) For boundary and applying offsets:

```
def draw_bounding_box(face_coordinates, image_array, color):
    x, y, w, h = face_coordinates
    cv2.rectangle(image_array, (x, y), (x + w, y + h), color, 2)

def apply_offsets(face_coordinates, offsets):
    x, y, width, height = face_coordinates
    x_off, y_off = offsets
    return (x - x_off, x + width + x_off, y - y_off, y + height + y_off)
```

b) For text on output window:

```
def draw_text(coordinates, image_array, text, color, x_offset=0, y_offset=0,
                                                    font_scale=2, thickness=2):
    x, y = coordinates[:2]
    cv2.putText(image_array, text, (x + x_offset, y + y_offset),
                cv2.FONT_HERSHEY_SIMPLEX,
                font_scale, color, thickness, cv2.LINE_AA)
```

c) For colour on text and box on output window:

```
def get_colors(num_classes):
    colors = plt.cm.hsv(np.linspace(0, 1, num_classes)).tolist()
    colors = np.asarray(colors) * 255
    return colors
```

4. For opening of webcam:

File name: emotions.py

a) For loading data and images:

```
emotion_model_path = './models/emotion_model.hdf5'
emotion_labels = get_labels('fer2013')
```

b) For loading models:

```
face_cascade = cv2.CascadeClassifier('./models/haarcascade_frontalface_default.xml')
emotion_classifier = load_model(emotion_model_path)
```

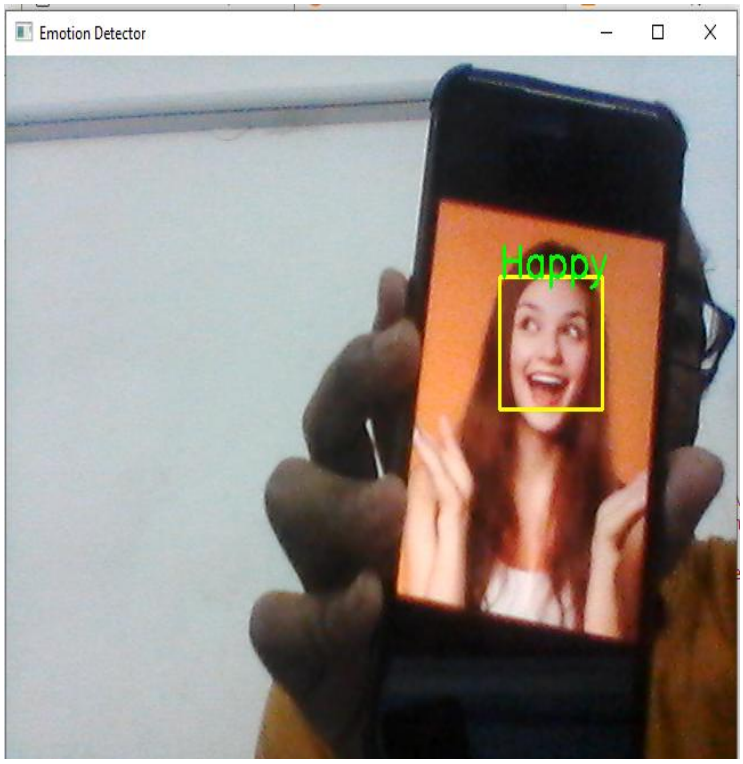
c) For starting webcam:

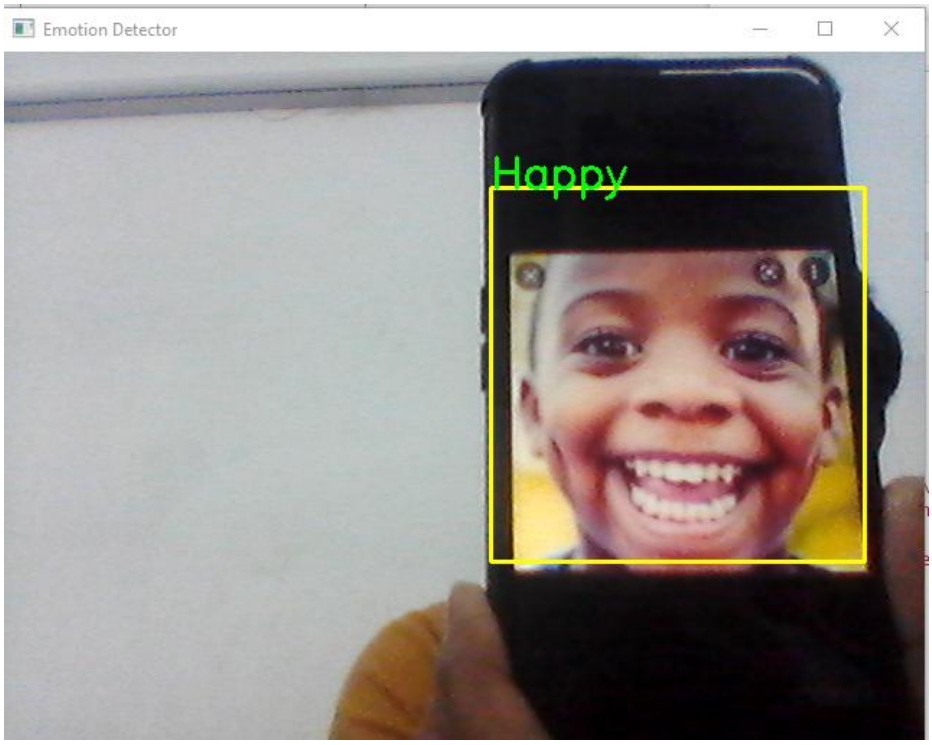
```
# starting video streaming
cv2.namedWindow('window_frame')
video_capture = cv2.VideoCapture(0)
# Select video or webcam feed
cap = None
if (USE_WEBCAM == True):
    cap = cv2.VideoCapture(0) # Webcam source
else:
    cap = cv2.VideoCapture('./demo/dinner.mp4') # Video file source
while cap.isOpened(): # True:
    ret, bgr_image = cap.read()
    #bgr_image = video_capture.read()[1]

    gray_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2GRAY)
    rgb_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2RGB)

    faces = face_cascade.detectMultiScale(gray_image, scaleFactor=1.1, minNeighbors=5,
                                          minSize=(30, 30), flags=cv2.CASCADE_SCALE_IMAGE)

    for face_coordinates in faces:

        x1, x2, y1, y2 = apply_offsets(face_coordinates, emotion_offsets)
        gray_face = gray_image[y1:y2, x1:x2]
        try:
            gray_face = cv2.resize(gray_face, (emotion_target_size))
        except:
            continue

        gray_face = preprocess_input(gray_face, True)
        gray_face = np.expand_dims(gray_face, 0)
```

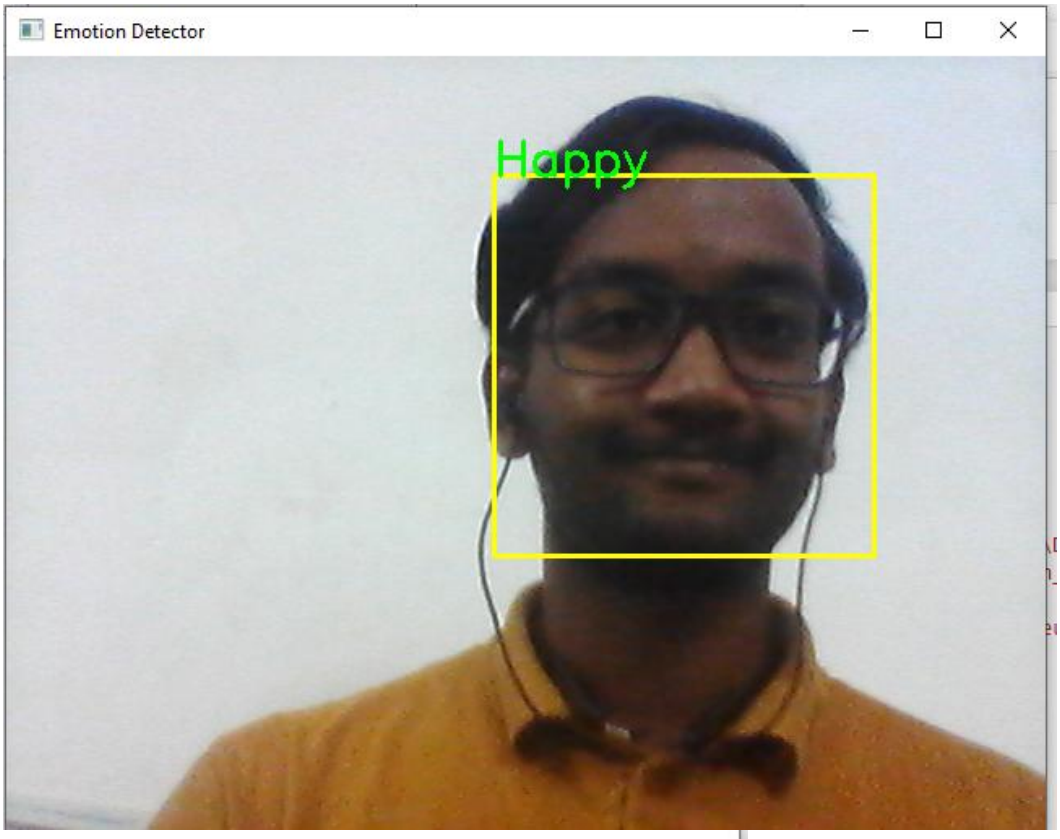
Chapter- 4

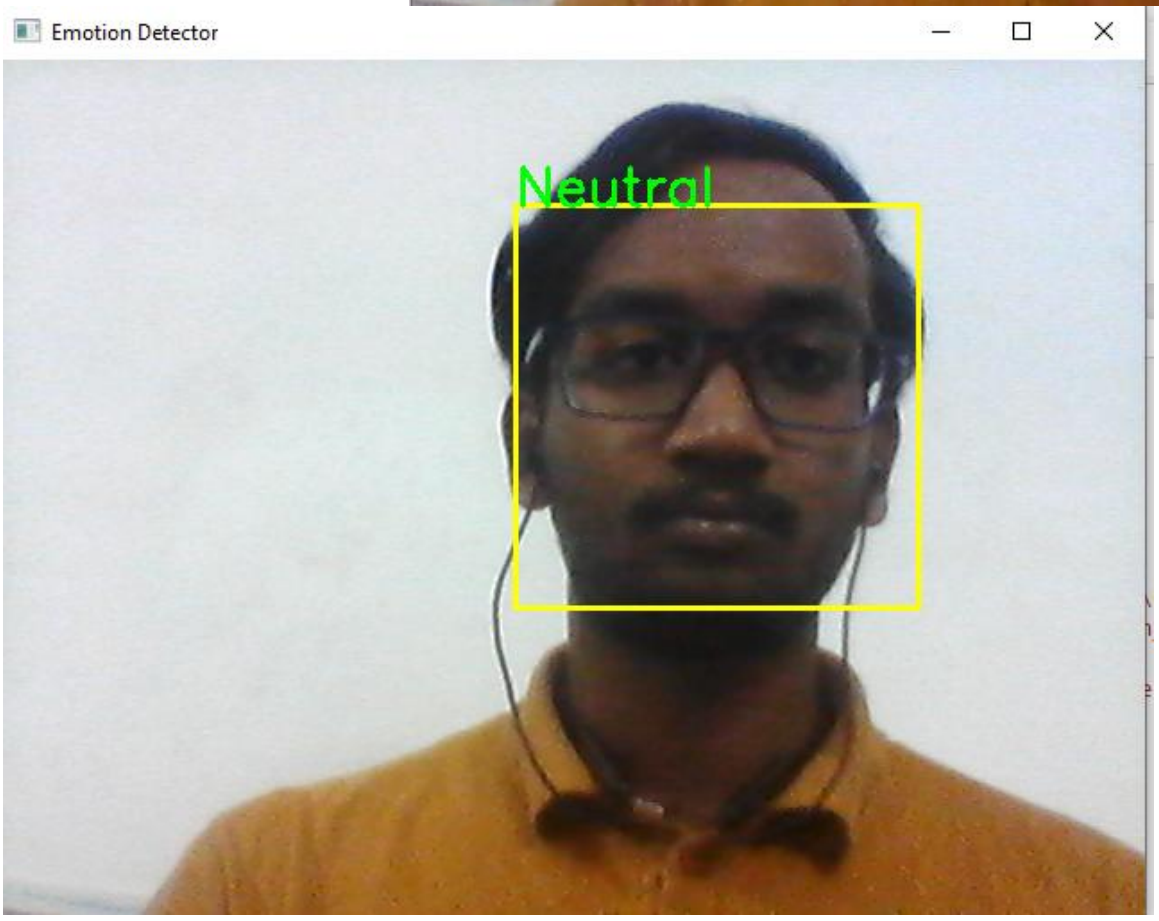
Result Analysis AND Validation

I. Output pictures:-

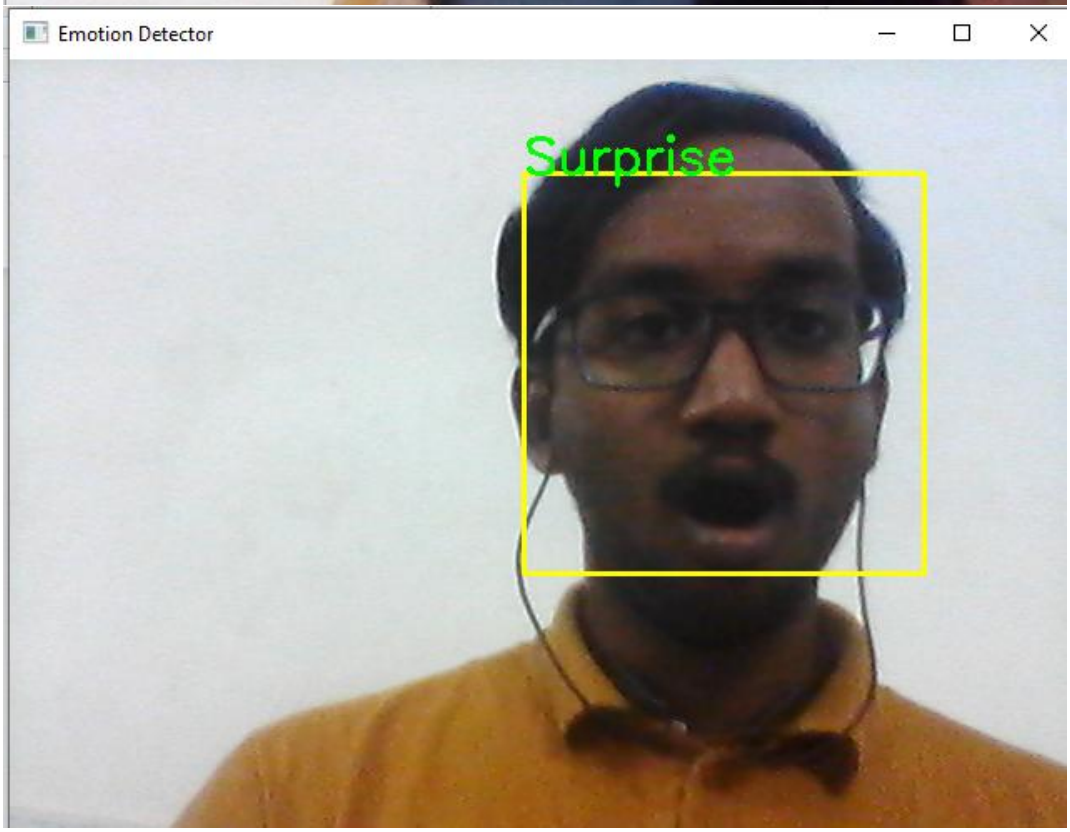


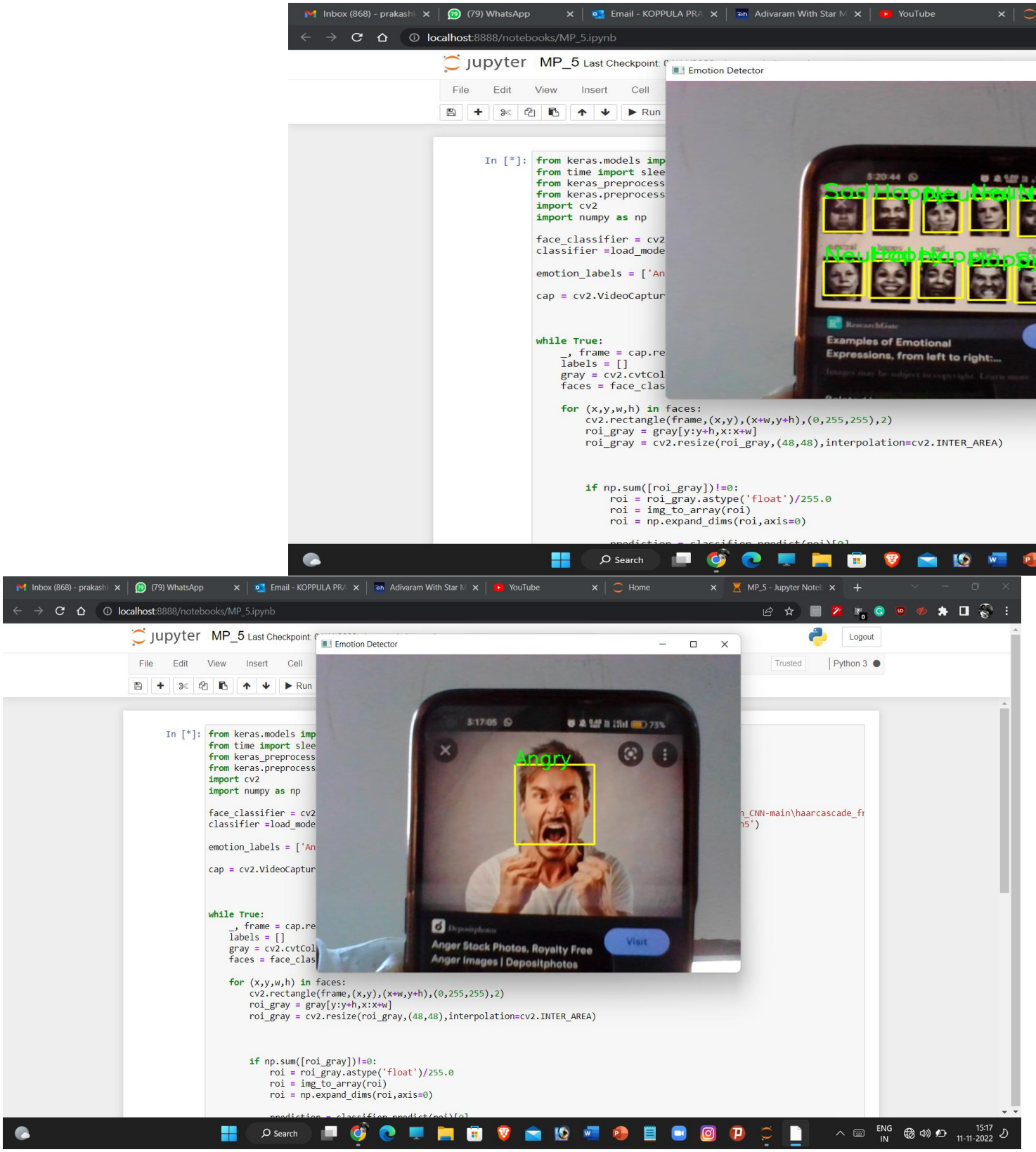



Emotion Detector
Happy

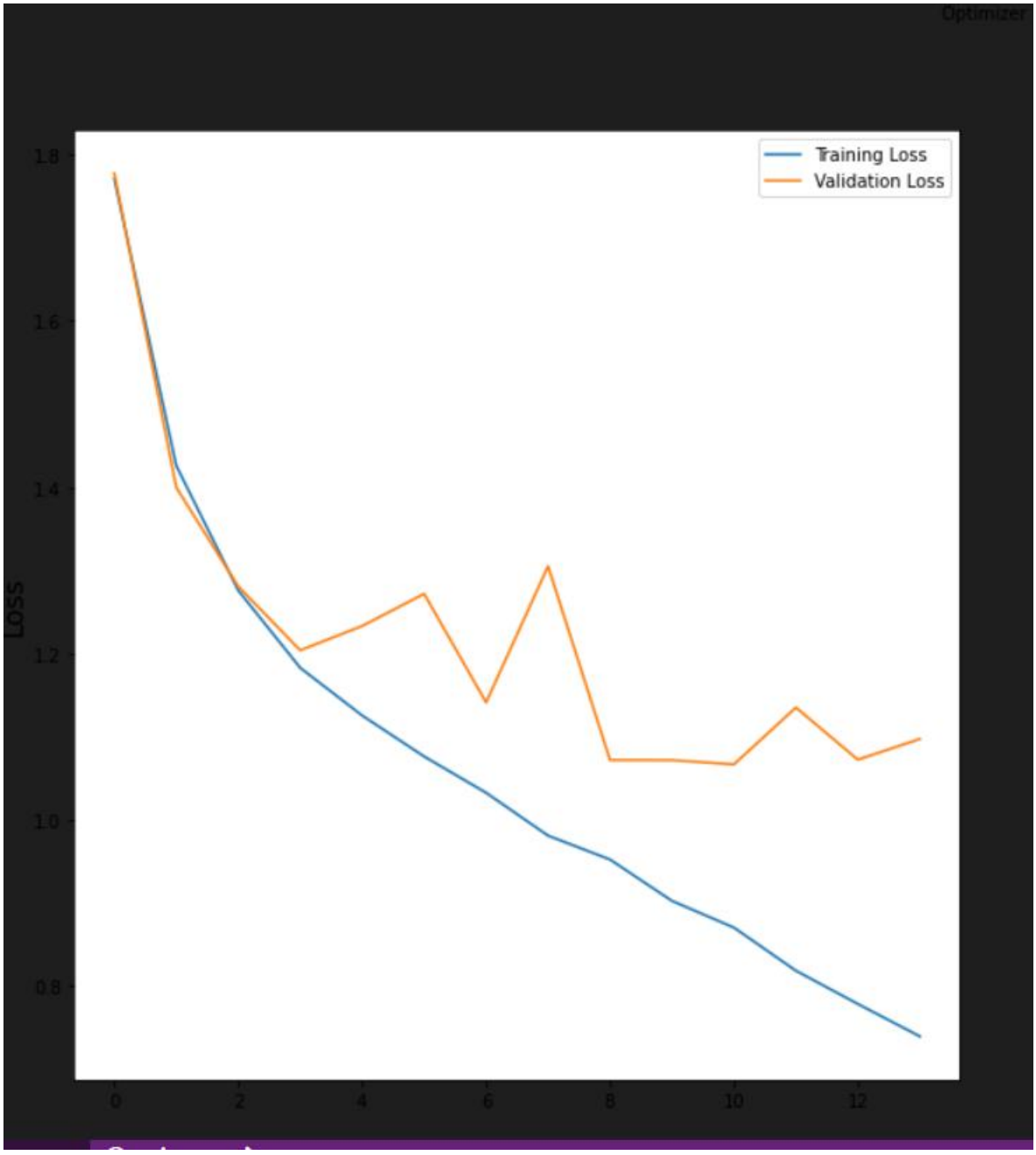

Emotion Detector
Neutral


Emotion Detector
Sad

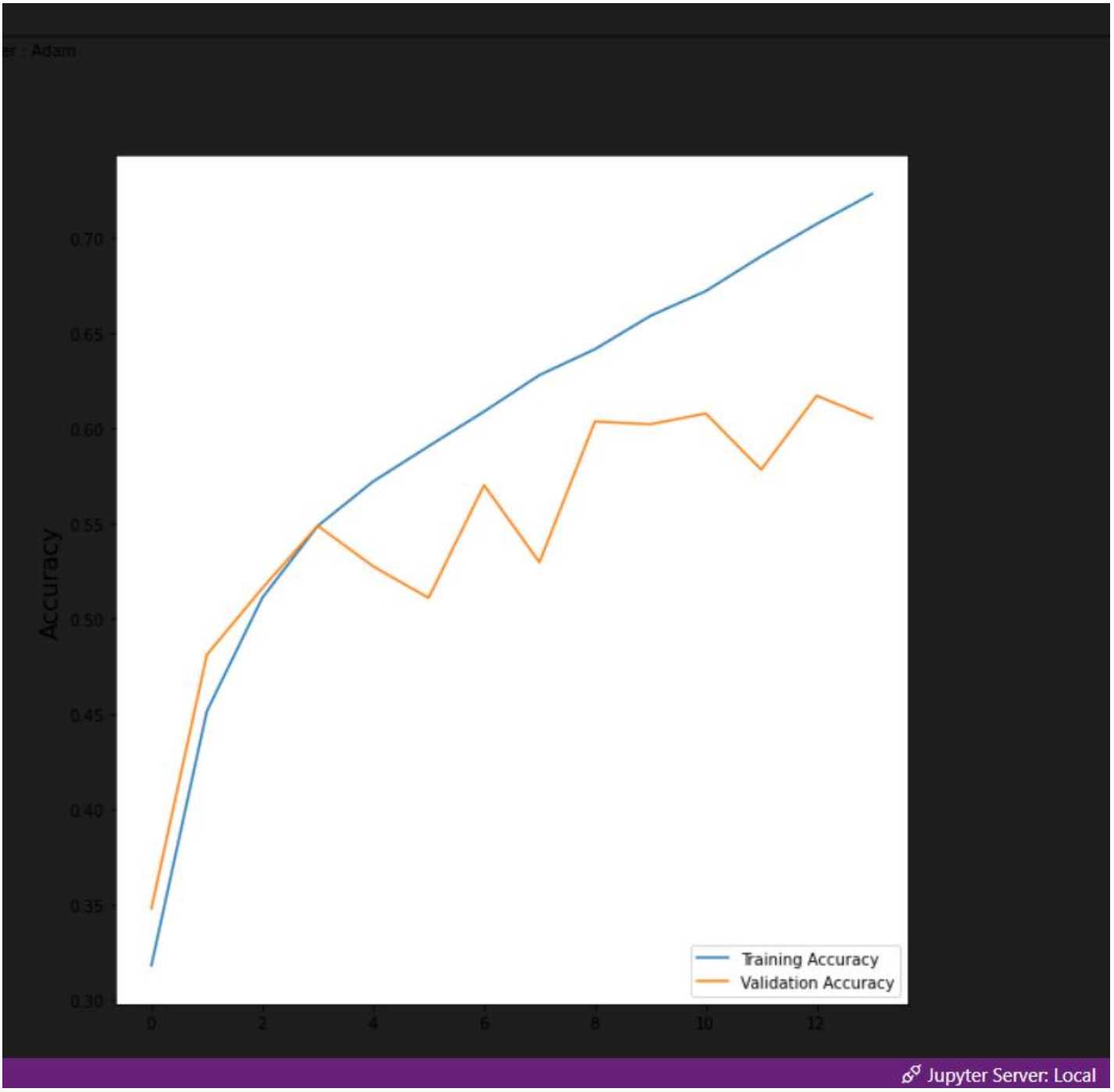

Emotion Detector
Surprise



II. Loss and Accuracy graph of our algorithm


Optimizer
Training Loss
Validation Loss
Loss
1.8
1.6
1.4
1.2
1.0
0.8
0
2
4
6
8
10
12

Loss graph



Accuracy graph

It can be clearly seen that at nearly about 15-17 iterations the accuracy of the model has crossed 75% and still the accuracy increases with increase in the number of iterations although the steepness is also decreasing. There are 48 iterations to make.

Chapter 5.

Conclusion:

In this Paper we discussed about the work done on emotion recognition and for achieving that all superior and novel approaches and methods. We have proposed a glimpse of a probable solution and method towards recognition the emotion. Work so far substantiate that emotion recognition using users EEG signal and audiovisual signal has the highest recognition rate and has highest performance.

Chapter 6.
References

[1] Guo, J., Zhou, S., Wu, J., Wan, J., Zhu, X., Lei, Z., & Li, S. Z. (2017). Multi-modality Network with Visual and Geometrical Information for Micro Emotion Recognition. 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), 814–819. https://doi.org/10.1109/FG.2017.103

[2] Liu, K., Zhang, M., & Pan, Z. (2016). Facial Expression Recognition with CNN Ensemble. Proceedings - 2016 International Conference on Cyberworlds, CW 2016, 163–166. https://doi.org/10.1109/CW.2016.34

[3] Nour, N., Elhebir, M., & Viriri, S. (2020). Face Expression Recognition using Convolution Neural Network (CNN) Models. International Journal of Grid Computing & Applications, 11(4), 1–11. https://doi.org/10.5121/ijgca.2020.11401

[4] Liu, Y., Yuan, X., Gong, X., Xie, Z., Fang, F., & Luo, Z. (2018). Conditional convolution neural network enhanced random forest for facial expression recognition. Pattern Recognition, 84, 251–261. https://doi.org/10.1016/j.patcog.2018.07.016

[5] Rajendra Kurup, A., Ajith, M., & Mart´ınez Ramon, M. (2019). Semi- ´ supervised facial expression recognition using reduced spatial features and Deep Belief Networks. Neurocomputing, 367, 188–197. https://doi.org/10.1016/j.neucom.2019.08.029