

NEURAL NETWORK TECHNIQUE FOR DETECTING FACIAL EXPRESSIONS OF EMOTION

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

BACHELOR OF ENGINEERING

IN

ARTIFICIAL INTELLIGENCE AND

MACHINE LEARNING

Submitted by:

KOPPULA PRAKASH – 21BCS8824

M L K SUBRAHMANYAM – 21BCS8803

ARYA CHACKRABORTHY – 21BCS8814

NIKHIL KUMAR - 20BCS6845

Under the Supervision of:

MS. SHUBHANGI MISHRA



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

APEX INSTITUTE OF TECHNOLOGY

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413,

PUNJAB

November. 2022

BONAFIDE CERTIFICATE

Certified that this project report “**NEURAL NETWORK TECHNIQUE FOR DETECTING FACIAL EXPRESSIONS OF EMOTIONS**” is the bonafide work of “**KOPPULA PRAKASH, M L K SUBRAHMANYAM, NIKHIL KUMAR, ARYA CHACKRABORTHY**” who carried out the project work under my/our supervision.

SIGNATURE OF THE HOD

AMAN KOUSHIK

(HEAD OF THE DEPARTMENT)

CSE - AIML

SIGNATURE OF THE SUPERVISOR

SHUBHANGI MISHRA

(SUPERVISOR)

(Asst.Professor)

(AIT – CSE)

Submitted for the project viva-voce examination held on

INTERNAL EXAMINER

EXTERNAL EXAMINER

TABLE OF CONTENTS

Title Page.....	i
Certificate.....	ii
List of figures.....	iv
List of Tables.....	iv
Abstract.....	1
Acknowledgement.....	2
Chapter 1. INTRODUCTION	3
Introduction	4
Problem definition.....	4
Software requirement.....	5
Hardware requirement.....	5
Chapter 2. LITERATURE SURVEY	6
Existing system	7
Proposed system.....	8
Chapter 3. DESIGN FLOW/PROCESS	9
Techniques & Tools.....	10
Theory.....	11
Datasets.....	14
Methodology.....	16
Chapter 4. IMPLEMENTATION SNAPSHOTS OF SOURCE CODE	18
Chapter 5. RESULT ANALYSIS AND VALIDATION.....	24
Chapter 6. CONCLUSION AND FUTURE SCOPE.....	29
Chapter 7. REFERECE.....	30

List of Figures:

Implementation of Face Detection	09
Implementation of Emotion Detection	10
Methodology	16
Dataset	14
Result Analysis & Validation	24
HAPPY.....	24
NEUTRAL.....	25
SAD.....	26
SURPRISED.....	26
ANGRY.....	27
All the emotions.....	27

List of Tables:

Table1. Emotion recognition different approach and successes	7
--	---

ABSTRACT

Emotion Recognition is a task to process a human facial expression and classify it into certain emotion categories. Such task typically requires the feature extractor to detect the feature, and the trained classifier produces the label based on the feature. The problem is that the extraction of feature may be distorted by variance of location of object and lighting condition in the image. In this project, we address the solution of the problem by using a deep learning algorithm called Convolutional Neural Network (CNN) to address the issues above. By using this algorithm, the feature of image can be extracted without user-defined feature-engineering, and classifier model is integrated with feature extractor to produce the result when input is given. In this way, such method produces a feature-location-invariant image classifier that achieves higher accuracy than traditional linear classifier when the variance such as lighting noise and background environment appears in the input image. The evaluation of the model shows that the accuracy of our lab condition testing data set is 84.63%, and for wild emotion detection it achieves only around 37% accuracy.

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 Ms. Shubhangi Mishra (Assistant Professor), CSE-Artificial Intelligence, Chandigarh University, Mohali for his valuable guidance, encouragement and help for completing this work. Her useful suggestions for this whole work and cooperative behavior 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

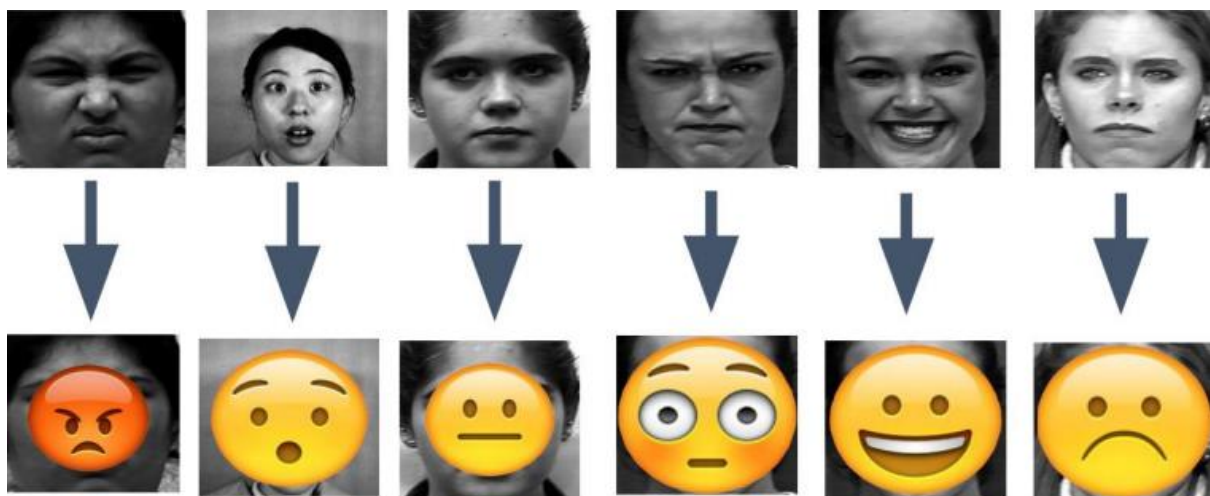
ARYA CHAKRABORTY 21BCS8814

KOPPULA PRAKASH 21BCS8824

Chapter – 1

INTRODUCTION

The facial expression of human emotion is one of the major topics in facial recognition, and it can generate both technical and everyday application beyond laboratory experiment. This project constructs a system of deep learning model to classify a given image of human facial emotion into one of the seven basic human emotions. The approach we take to build the model is through transfer learning of an existing pre-trained model, and the testing result will be evaluated based on accuracy of the model. The tasks in this project include preprocessing the image data, augmentation to enlarge the existing small dataset, test before training the model, training process, and predication with evaluation. The visual demonstration of the result will be similar to the figure below. The baseline of the test will be around 14.7% (1 in 7), and our objective is to achieve a result better than baseline accuracy.



A demonstration: The emojis are correctly imposed on their corresponding faces. There are seven different emotions: happy, angry, sad, fear, surprise, neutral, and disgust. The input will be raw image of the expression, and output will be shown as above. The demonstration does not include disgust as its emoji is like angry.

1.1 PROBLEM DEFINITION

In real life, people express their emotion on their face to show their psychological activities and attitudes in the interaction with other people. The primary focus of this project is to determine which emotion an input image that contains one facial emotion belongs to. Because human face is complex to interpret, emotion recognition can be specifically divided into classification of basic emotion and classification of compound emotion. For the goals of our project, the essential problem is to focus on the classification of 7 basic emotions.

1.2 PROJECT OVERVIEW

Facial Emotion Recognition typically has four steps. The first is to detect a face in an image and draw a rectangle around it and the next step is to detect landmarks in this face region. The third step is extracting spatial and temporal features from the facial components. The final step is to use a Feature Extraction (Facial Emotion) classifier and produce the recognition results using the extracted features. With the emotion recognition system, AI can detect the emotions of a person through their facial expressions. Detected emotions can fall into any of the six main data of emotions: happiness, sadness, fear, surprise, disgust, and anger. For example, a smile on a person can be easily identified by the AI as happiness. Human emotion recognition plays an important role in the interpersonal relationship. The automatic recognition of emotions has been an active research topic from early eras. Therefore, there are several advances made in this field. Emotions are reflected from speech, hand, and gestures of the body and through facial expressions. Hence extracting and understanding of emotion has a high importance of the interaction between human and machine communication. This paper describes the advances made in this field and the various approaches used for recognition of emotions. The main objective of the paper is to propose real time implementation of emotion recognition system.

1.3 TIMELINE

S.N	Strategies	1 st week	2 nd week	3 rd week	4 th week	5 th week	6 th week
1)	Problem Identification						
2)	Research & Analysis						
3)	Design						
4)	Coding						
5)	Implementation & testing						
6)	Project finalisation						
7)	Documentation						

1.4 HARDWARE AND SOFTWARE REQUIREMENTS

HARDWARE SPECIFICATIONS

- Personal computer with keyboard and mouse maintained with uninterrupted power supply.
- Processor: Intel® core™ i5
- Installed Memory (RAM): 8.00 GB

SOFTWARE SPECIFICATIONS

- Operating System: WINDOWS 7, 8.1,10,11
- Coding language: PYTHON
- Web Browser: GOOGLE CHROME

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. A regularized extreme learning machine (RELM) classifier is used to classify five various expressions of the teacher within the classroom for quick

learning and effective generalization of the method.

2.1 EXISTING SYSTEM

The existing conventional method requires a handcrafted feature extractor of facial Action Units (AUs) to extract feature from designated Facial Landmark regions, and these extracted AUs codes are processed through traditional machine learning algorithm such as Nearest Neighbors and SVM, which is a typical type of linear classifier. The problem with conventional method is that the lighting 9 variations and different position of object may corrupt the feature vector so that the accuracy is greatly reduced. Furthermore, it is typically difficult to conduct feature-engineering to fit the demand of facial recognition.

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

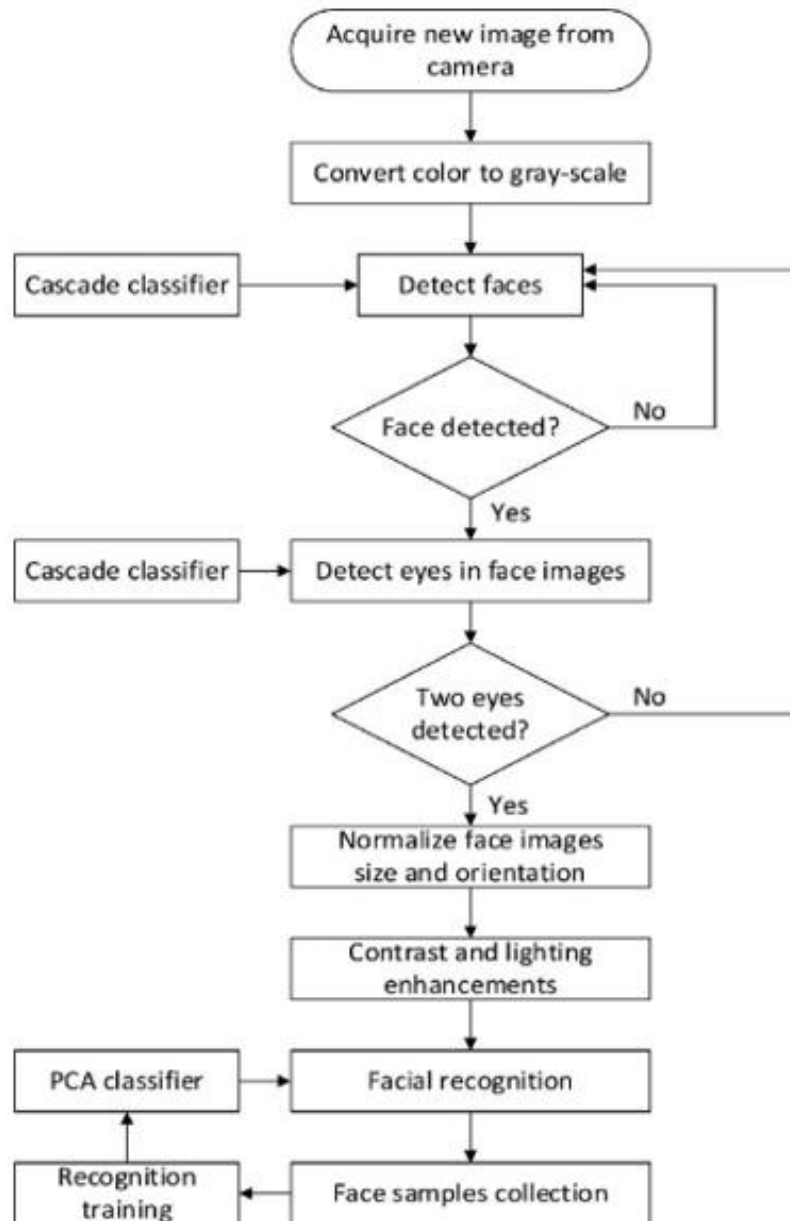


Fig: Above, the implementation of face detection.

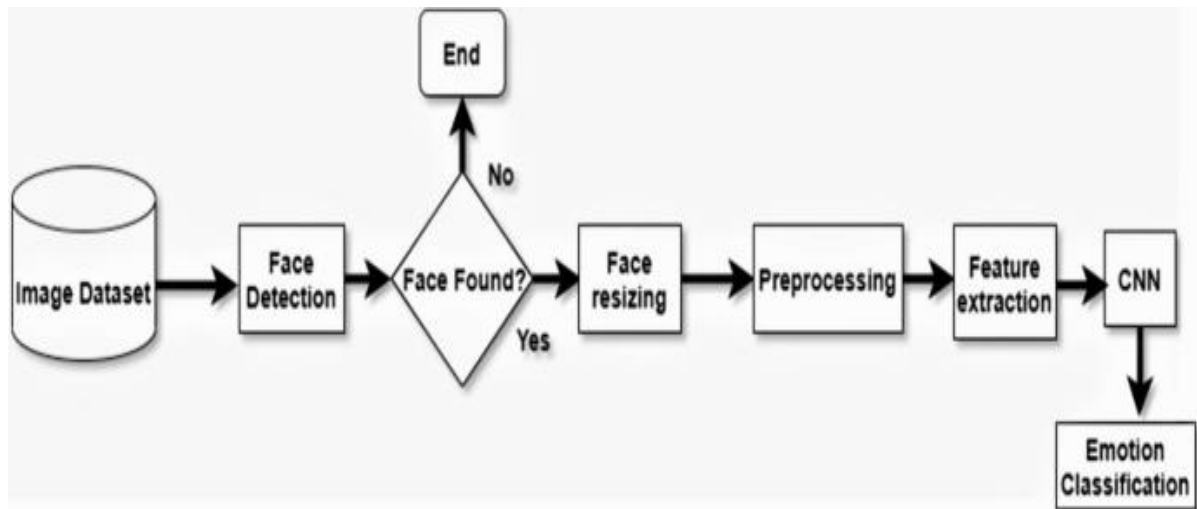


Fig: Above is the implementation of Emotion detection

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 —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 we used the PyCharm and Anaconda for base. We used Matplotlib for visualization.

The three main phases of the project will be:

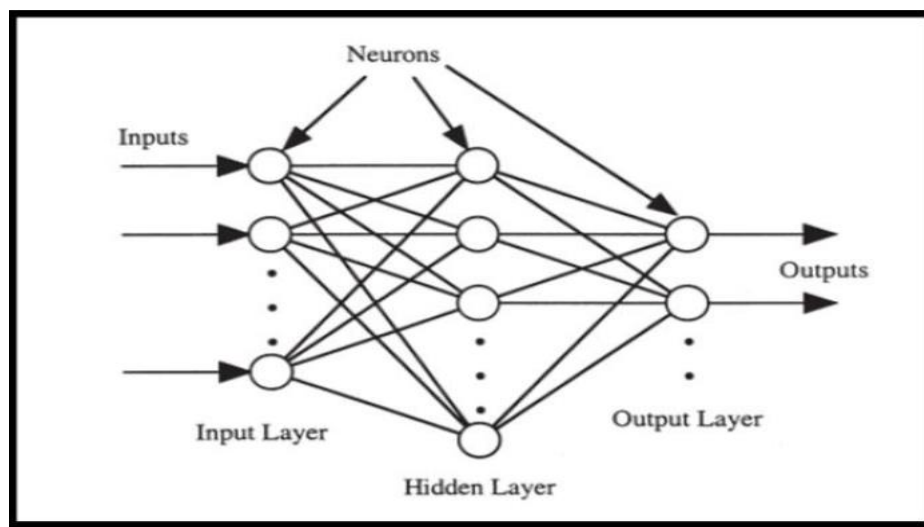
To create a complete project on Face Recognition, we must work on 3 very distinct phases:

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

The main modules used in this project are NumPy, Pandas, scikitplot, seaborn, sklearn, tensorflow, Keras layers and some CNN functions.

What is Neural Network?

Neural networks are designed by observing the functionality of neurons in a human brain. In a neural network, there are three layers: Input Layer, Hidden Layers, and Output layer. The input layer consists of the inputs or the independent X variable known as the predictors. These inputs are collected from external sources such as text data, images, audio, or video files.



A convolutional neural network (CNN or convnet) is a subset of machine learning. It is one of the various types of artificial neural networks which are used for different applications and data types. A CNN is a kind of network architecture for deep learning algorithms and is specifically used for image recognition and tasks that involve the processing of pixel data. There are other types of neural networks in deep learning, but for identifying and recognizing objects, CNNs are the network architecture of choice. This makes them highly suitable for computer vision (CV) tasks and for applications where object recognition is vital, such as self-driving cars and facial recognition.

LAYERS OF CNN:

1) Convolutional Layer

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size $M \times M$. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter ($M \times M$).

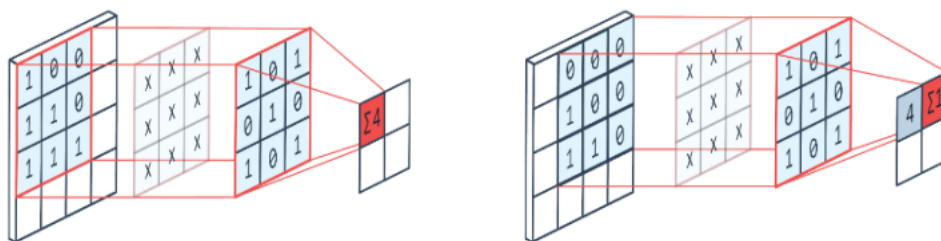


Fig: Convolutional Layer

2) Pooling Layer

In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer.

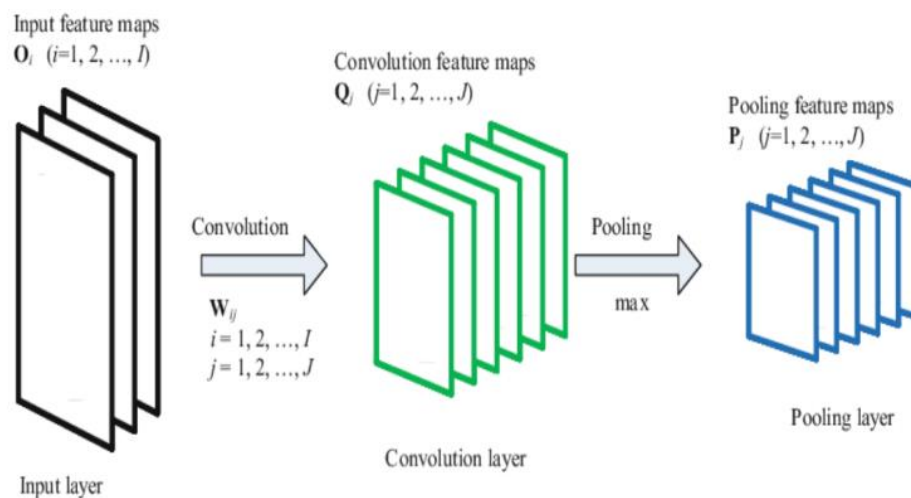


Fig: Pooling Layer

3) Fully-connected layer

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture. In this, the input image from the previous layers is flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place.

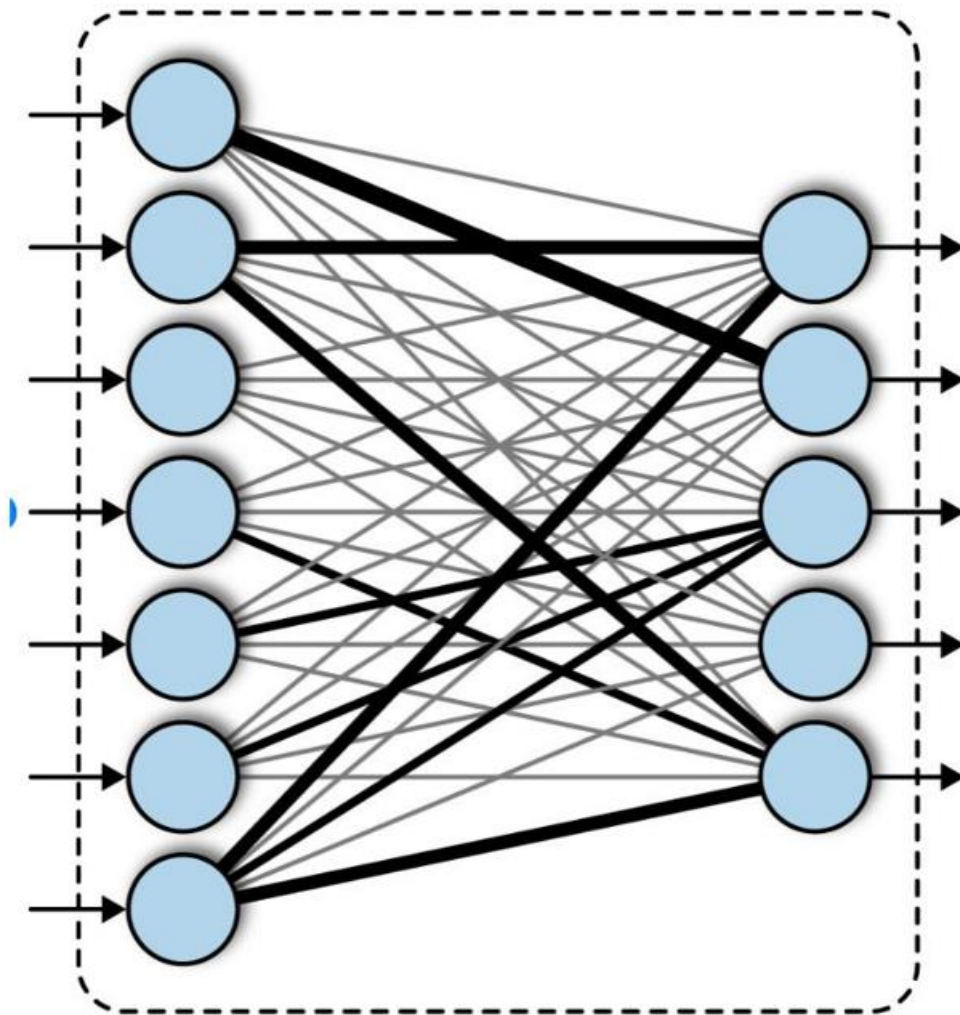


Fig: Fully connected Layer

4) Activation functions

Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network.

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.



Fig: Samples from the FER-2013 Dataset



Fig: Guided back-propagation visualization of our sequential fully CNN.

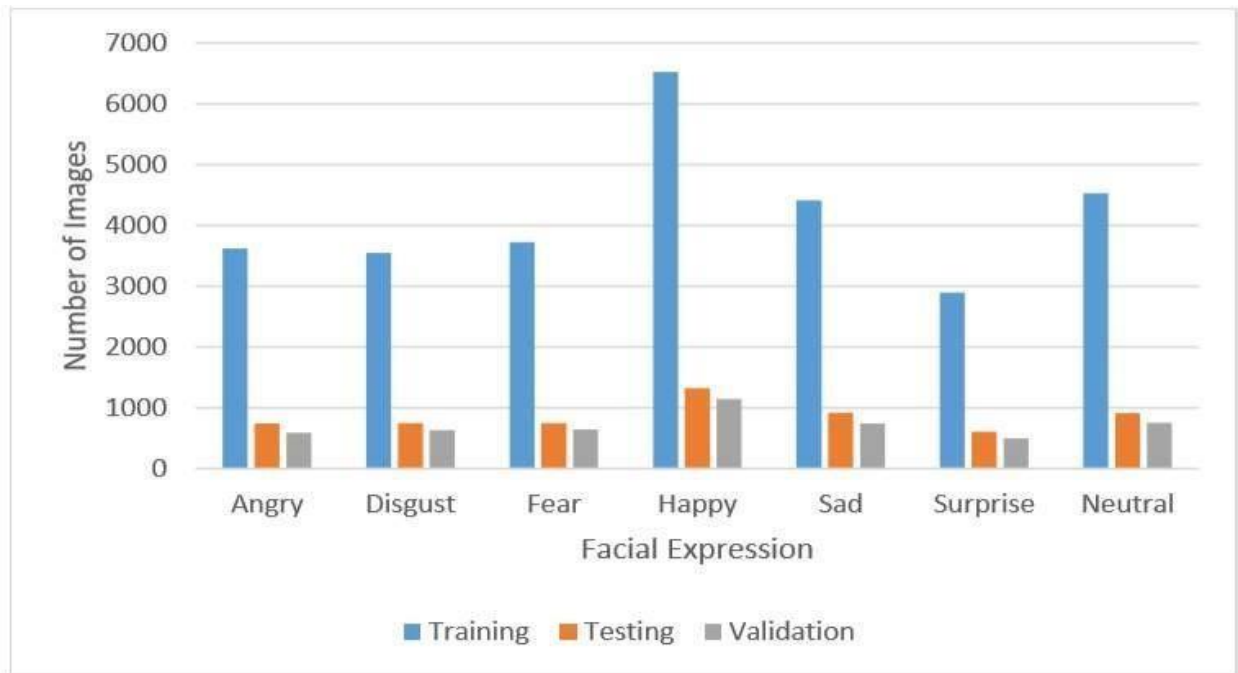


Fig: Training, Testing and Validation Data distribution

METHODOLOGY

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

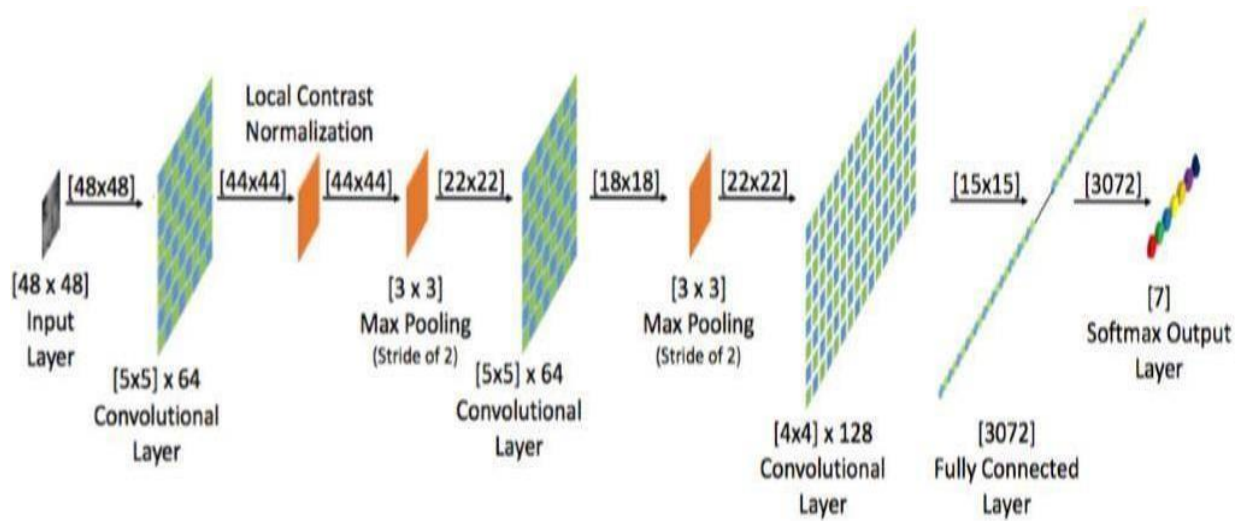


Fig: Layers in CNN

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.

Chapter – 4

IMPLEMENTATION SNAPSHOTS OF SOURCE CODE

Training a Model using FER-2013 Dataset

```
emotion-classification-cnn-using-keras.ipynb - Visual Studio Code
D:\5th Sem Project > Emotion_Detection_CNN-main > Emotion_Detection_CNN-main > emotion-classification-cnn-using-keras.ipynb > Importing Libraries
+ Code + Markdown | Outline ...

Importing Libraries

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import os

# Importing Deep Learning Libraries
from keras.preprocessing.image import load_img, img_to_array
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Dense, Input, Dropout, GlobalAveragePooling2D, Flatten, Conv2D, BatchNormalization, Activation, MaxPooling2D
from keras.models import Model, Sequential
from keras.optimizers import Adam, SGD, RMSprop

Python

Displaying Images

picture_size = 48
folder_path = "../input/face-expression-recognition-dataset/images/"

expression = 'disgust'

plt.figure(figsize=(12,12))


Python
```

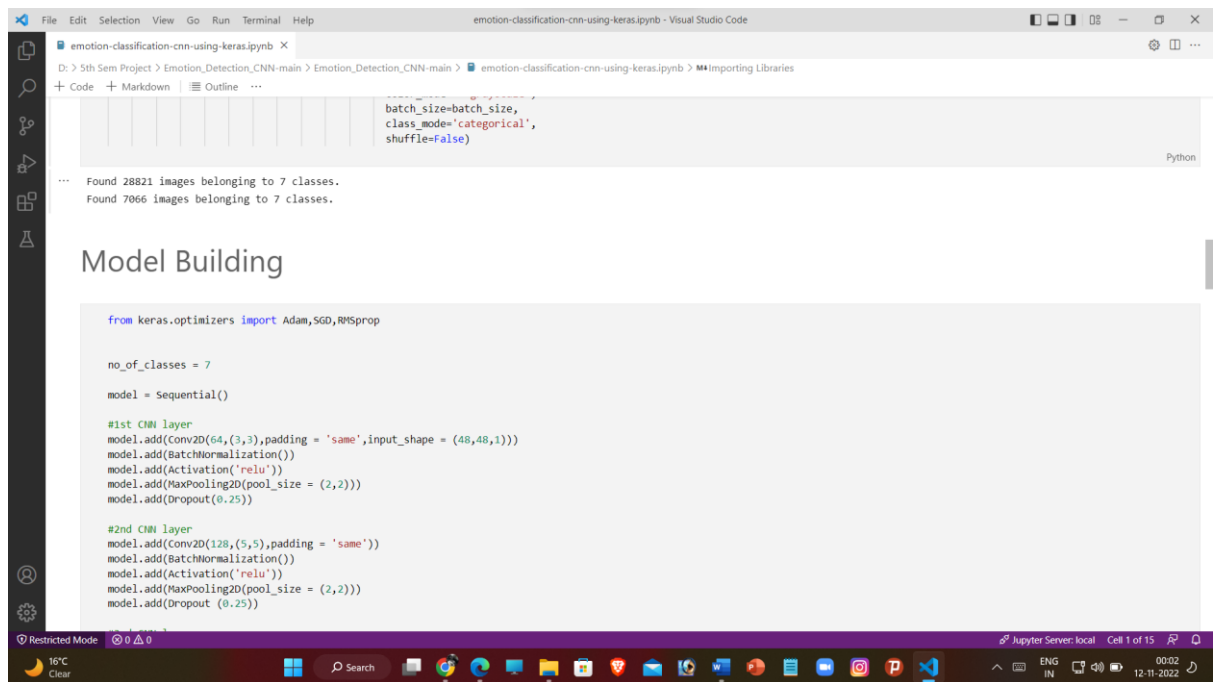
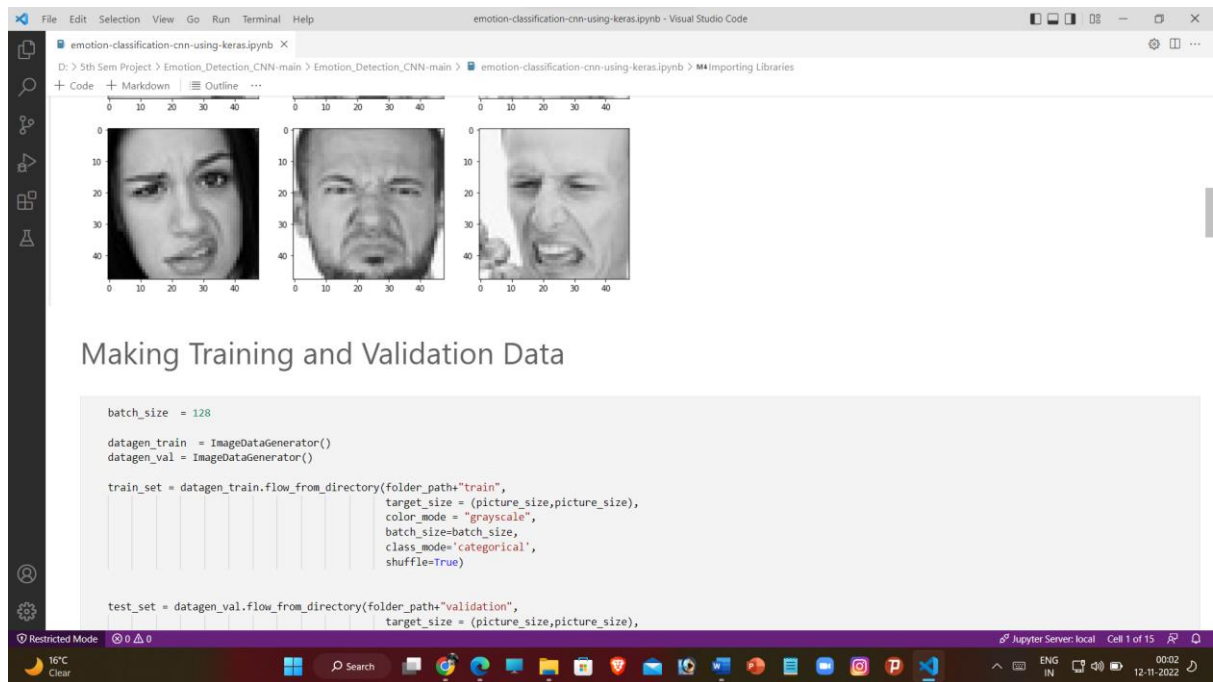
```
emotion-classification-cnn-using-keras.ipynb - Visual Studio Code
D:\5th Sem Project > Emotion_Detection_CNN-main > Emotion_Detection_CNN-main > emotion-classification-cnn-using-keras.ipynb > Importing Libraries
+ Code + Markdown | Outline ...

expression = 'disgust'

plt.figure(figsize=(12,12))
for i in range(1, 10, 1):
    plt.subplot(3,3,i)
    img = load_img(folder_path+"train/"+expression+"/"+
        os.listdir(folder_path+"train/"+expression)[i], target_size=(picture_size, picture_size))
    plt.imshow(img)
plt.show()

Python
```





```
emotion-classification-cnn-using-keras.ipynb - Visual Studio Code
D: > 5th Sem Project > Emotion_Detection_CNN-main > Emotion_Detection_CNN-main > emotion-classification-cnn-using-keras.ipynb > Importing Libraries

model.add(Dropout(0.25))

#3rd CNN Layer
model.add(Conv2D(512,(3,3),padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

#4th CNN Layer
model.add(Conv2D(512,(3,3),padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())

#Fully connected 1st layer
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

# Fully connected layer 2nd layer
model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

model.add(Dense(no_of_classes, activation='softmax'))

opt = Adam(lr = 0.0001)
model.compile(optimizer=opt,loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
emotion-classification-cnn-using-keras.ipynb - Visual Studio Code
D: > 5th Sem Project > Emotion_Detection_CNN-main > Emotion_Detection_CNN-main > emotion-classification-cnn-using-keras.ipynb > Importing Libraries

opt = Adam(lr = 0.0001)
model.compile(optimizer=opt,loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

Output exceeds the [size limit](#). Open the full output data [in a text editor](#)

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 64)	640
batch_normalization (Batch Normalization)	(None, 48, 48, 64)	256
activation (Activation)	(None, 48, 48, 64)	0
max_pooling2d (MaxPooling2D)	(None, 24, 24, 64)	0
dropout (Dropout)	(None, 24, 24, 64)	0
conv2d_1 (Conv2D)	(None, 24, 24, 128)	204928
batch_normalization_1 (Batch Normalization)	(None, 24, 24, 128)	512
activation_1 (Activation)	(None, 24, 24, 128)	0
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 128)	0
dropout_1 (Dropout)	(None, 12, 12, 128)	0

emotional-classification-cnn-using-keras.ipynb - Visual Studio Code

emotional-classification-cnn-using-keras.ipynb X

D:\5th Sem Project > Emotion_Detection_CNN-main > Emotion_Detection_CNN-main > emotional-classification-cnn-using-keras.ipynb > Importing Libraries

+ Code + Markdown | Outline ...

dropout_1 (Dropout)	(None, 12, 12, 128)	0
conv2d_2 (Conv2D)	(None, 12, 12, 512)	590336
...		
Total params: 4,478,727		
Trainable params: 4,474,759		
Non-trainable params: 3,968		

Fitting the Model with Training and Validation Data

```

from keras.optimizers import RMSprop, SGD, Adam
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau

checkpoint = ModelCheckpoint("./model.h5", monitor='val_acc', verbose=1, save_best_only=True, mode='max')

early_stopping = EarlyStopping(monitor='val_loss',
                                min_delta=0,
                                patience=3,
                                verbose=1,
                                restore_best_weights=True
                                )

reduce_learningrate = ReduceLROnPlateau(monitor='val_loss',
                                          factor=0.2,
                                          patience=3,
                                          verbose=1,
                                          min_delta=0.0001)

callbacks_list = [early_stopping, checkpoint, reduce_learningrate]

```

Restricted Mode 0 0 0 Jupyter Server: local Cell 1 of 15 16°C Clear 00:03 12-11-2022

emotional-classification-cnn-using-keras.ipynb - Visual Studio Code

emotional-classification-cnn-using-keras.ipynb X

D:\5th Sem Project > Emotion_Detection_CNN-main > Emotion_Detection_CNN-main > emotional-classification-cnn-using-keras.ipynb > Importing Libraries

+ Code + Markdown | Outline ...

```

min_delta=0.0001

callbacks_list = [early_stopping, checkpoint, reduce_learningrate]

epochs = 48

model.compile(loss='categorical_crossentropy',
              optimizer = Adam(lr=0.001),
              metrics=['accuracy'])

history = model.fit_generator(generator=train_set,
                              steps_per_epoch=train_set.n//train_set.batch_size,
                              epochs=epochs,
                              validation_data = test_set,
                              validation_steps = test_set.n//test_set.batch_size,
                              callbacks=callbacks_list
                              )

```

Output exceeds the size limit. Open the full output data in a text editor

```

Epoch 1/48
225/225 [=====] - 180s 799ms/step - loss: 1.7711 - accuracy: 0.3183 - val_loss: 1.7775 - val_accuracy: 0.3482
Epoch 2/48
225/225 [=====] - 22s 99ms/step - loss: 1.4261 - accuracy: 0.4517 - val_loss: 1.3999 - val_accuracy: 0.4814
Epoch 3/48
225/225 [=====] - 22s 97ms/step - loss: 1.2760 - accuracy: 0.5113 - val_loss: 1.2810 - val_accuracy: 0.5163
Epoch 4/48
225/225 [=====] - 22s 100ms/step - loss: 1.1831 - accuracy: 0.5490 - val_loss: 1.2041 - val_accuracy: 0.5491
Epoch 5/48
225/225 [=====] - 23s 100ms/step - loss: 1.1257 - accuracy: 0.5724 - val_loss: 1.2334 - val_accuracy: 0.5278
Epoch 6/48
225/225 [=====] - 22s 98ms/step - loss: 1.0765 - accuracy: 0.5909 - val_loss: 1.2720 - val_accuracy: 0.5112

```

Restricted Mode 0 0 0 Jupyter Server: local Cell 1 of 15 16°C Clear 00:03 12-11-2022

emotion-classification-cnn-using-keras.ipynb - Visual Studio Code

D: > 5th Sem Project > Emotion_Detection_CNN-main > Emotion_Detection_CNN-main > emotion-classification-cnn-using-keras.ipynb > MAImporting Libraries

225/225 [=====] - 23s 103ms/step - loss: 0.9813 - accuracy: 0.6282 - val_loss: 1.3052 - val_accuracy: 0.5300
 Epoch 9/48
 225/225 [=====] - 23s 103ms/step - loss: 0.9526 - accuracy: 0.6418 - val_loss: 1.0722 - val_accuracy: 0.6038
 Epoch 10/48
 225/225 [=====] - 27s 120ms/step - loss: 0.9028 - accuracy: 0.6593 - val_loss: 1.0720 - val_accuracy: 0.6024
 Epoch 11/48
 225/225 [=====] - 23s 104ms/step - loss: 0.8707 - accuracy: 0.6724 - val_loss: 1.0670 - val_accuracy: 0.6081
 Epoch 12/48
 225/225 [=====] - 22s 98ms/step - loss: 0.8188 - accuracy: 0.6906 - val_loss: 1.1353 - val_accuracy: 0.5786
 Epoch 13/48
 ...
 Epoch 00014: ReduceLROnPlateau reducing learning rate to 0.000200000000949949026.
 225/225 [=====] - 23s 102ms/step - loss: 0.7399 - accuracy: 0.7234 - val_loss: 1.0975 - val_accuracy: 0.6054
 Epoch 00014: early stopping

Plotting Accuracy & Loss

```
plt.style.use('dark_background')

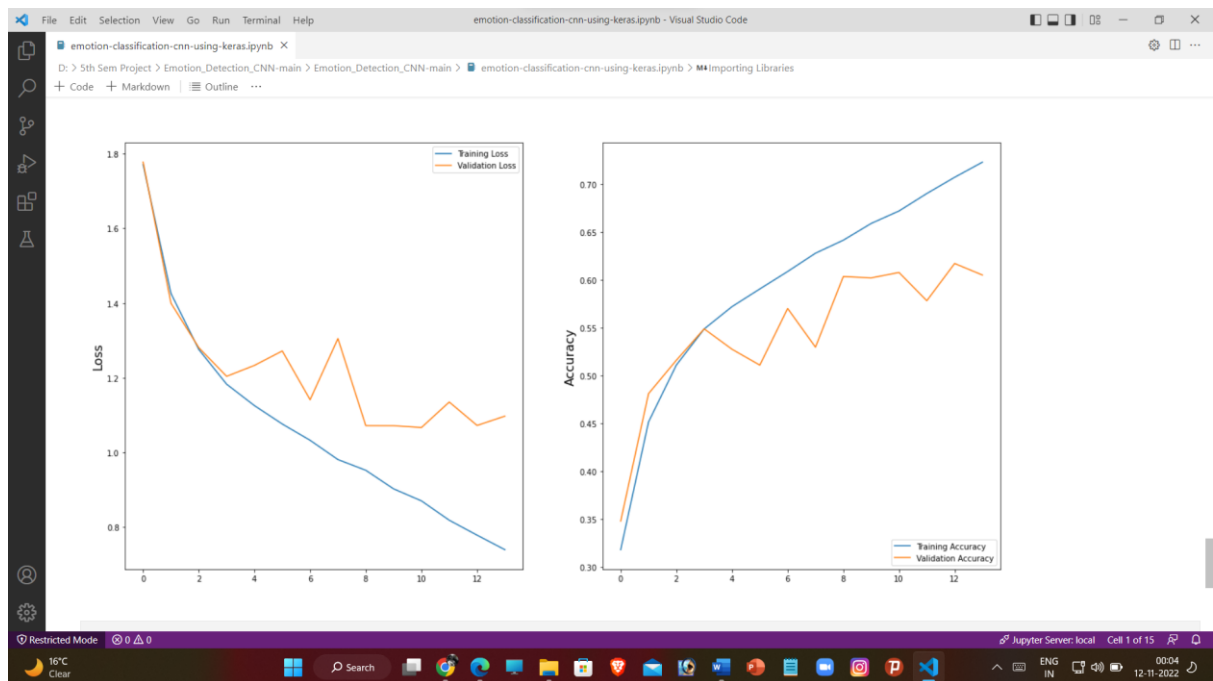
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : Adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
```

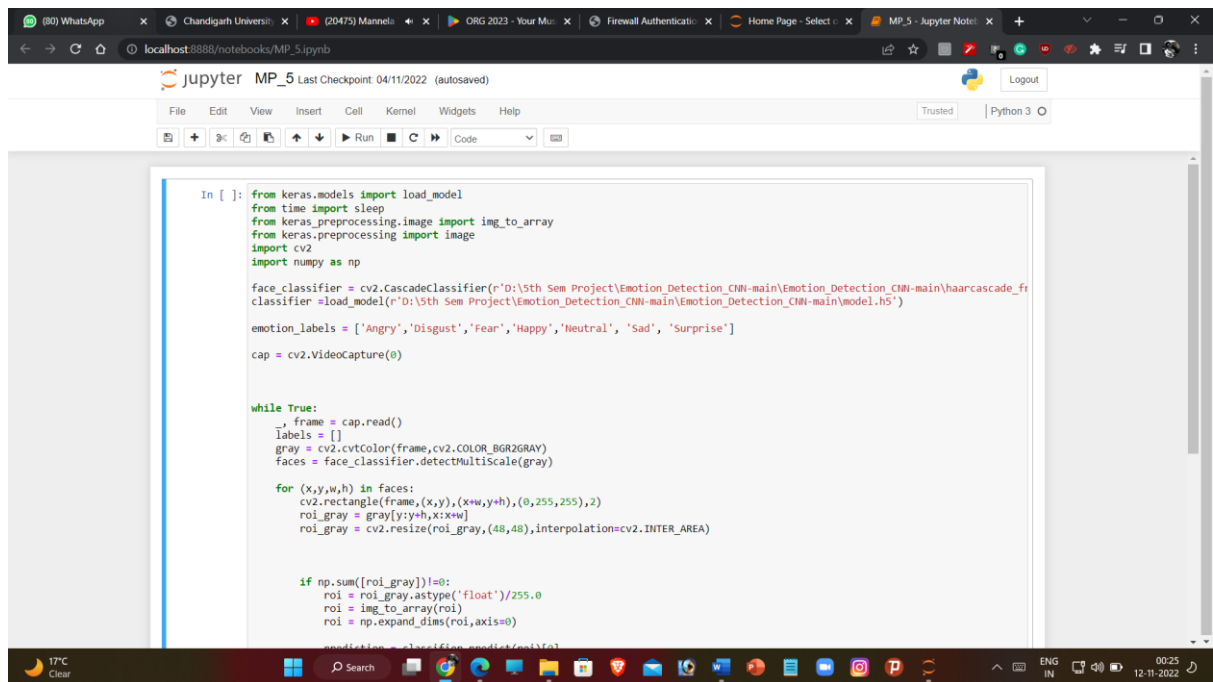
16°C Clear

Jupyter Server: local Cell 1 of 15

0004 12-11-2022



Testing of the Model



The screenshot shows a Jupyter Notebook titled 'MP_5' with a last checkpoint of '04/11/2022 (autosaved)'. The code in the first cell is as follows:

```
In [ ]: from keras.models import load_model
from time import sleep
from keras.preprocessing.image import img_to_array
from keras.preprocessing import image
import cv2
import numpy as np

face_classifier = cv2.CascadeClassifier(r'D:\5th Sem Project\Emotion_Detection_CNN-main\Emotion_Detection_CNN-main\haarcascade_f
classifier = load_model(r'D:\5th Sem Project\Emotion_Detection_CNN-main\Emotion_Detection_CNN-main\model.h5')

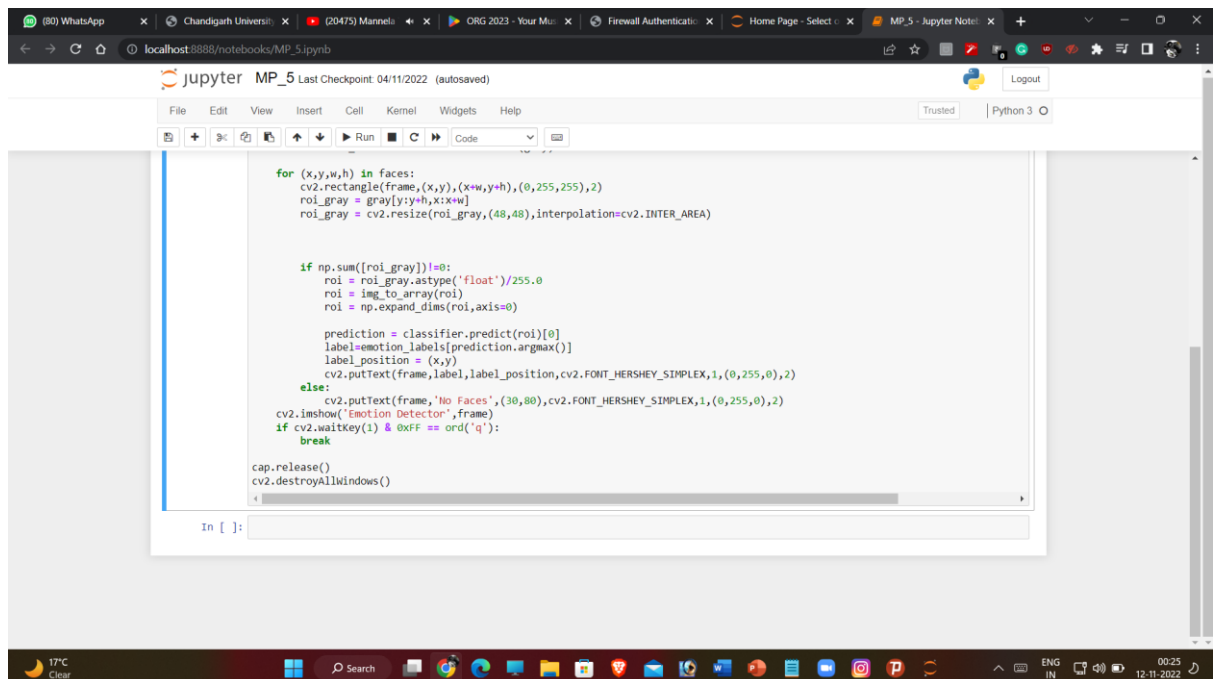
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

cap = cv2.VideoCapture(0)

while True:
    frame = cap.read()
    labels = []
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)
        roi_gray = gray[y:y+h,x:x+w]
        roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.INTER_AREA)

        if np.sum([roi_gray])!=0:
            roi = roi_gray.astype('float')/255.0
            roi = img_to_array(roi)
            roi = np.expand_dims(roi,axis=0)
```



The screenshot shows the continuation of the Jupyter Notebook. The code in the second cell is as follows:

```
        prediction = classifier.predict(roi)[0]
        label=emotion_labels[prediction.argmax()]
        label_position = (x,y)
        cv2.putText(frame,label,label_position,cv2.FONT_HERSHEY_SIMPLEX,1,(0,255,0),2)
    else:
        cv2.putText(frame,'No Faces',(30,80),cv2.FONT_HERSHEY_SIMPLEX,1,(0,255,0),2)
    cv2.imshow('Emotion Detector',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

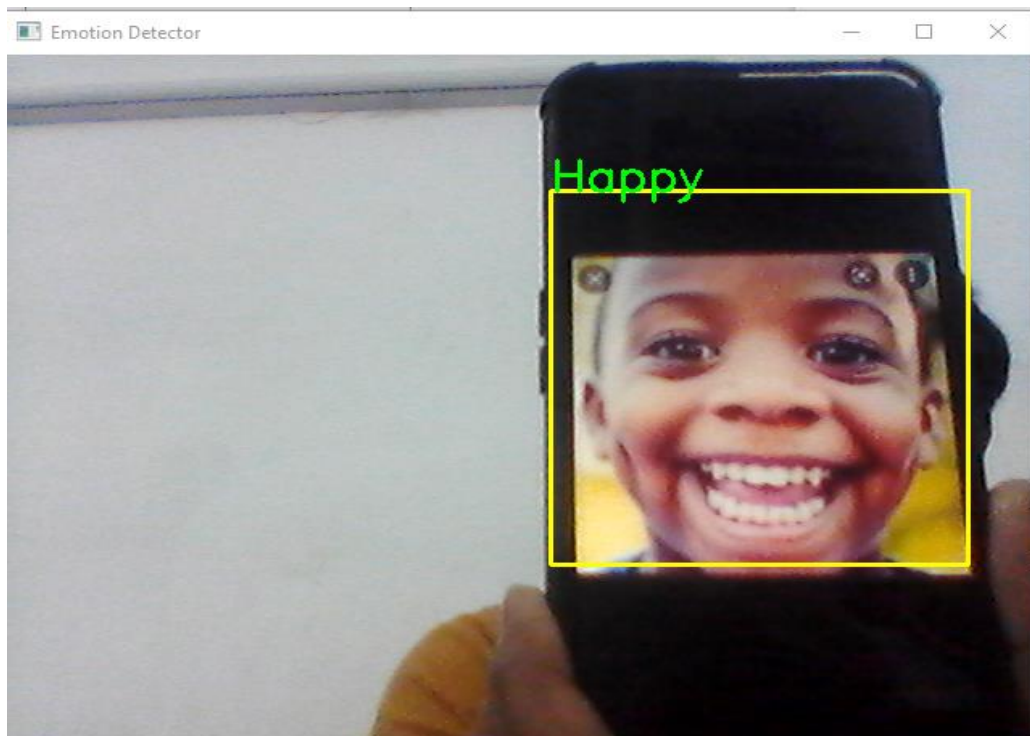
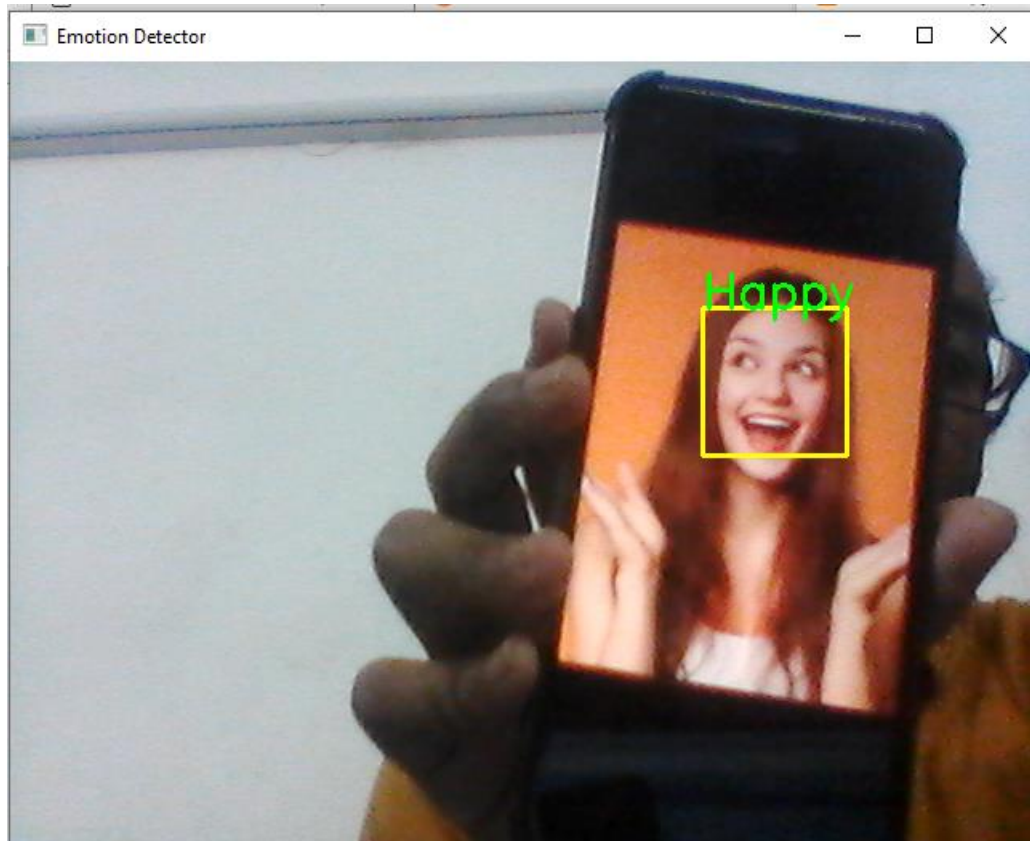
cap.release()
cv2.destroyAllWindows()

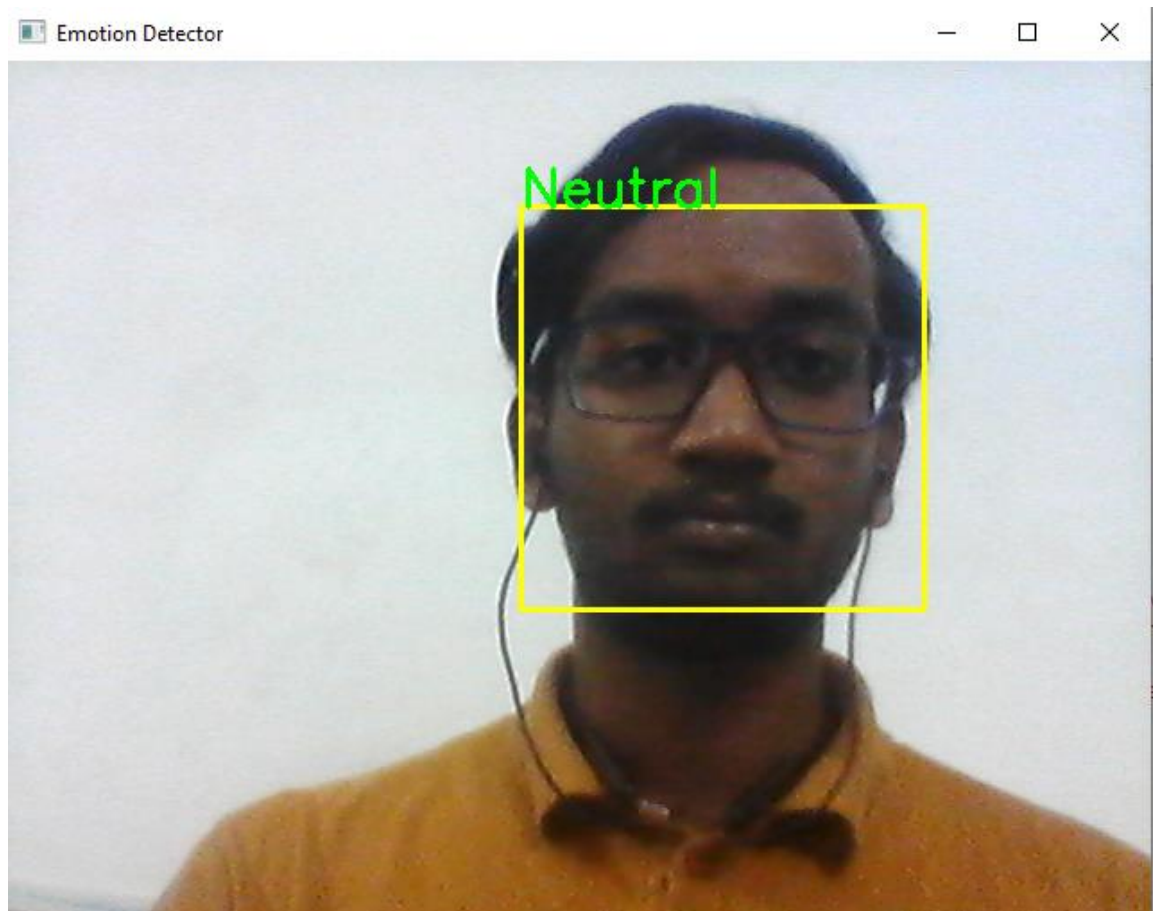
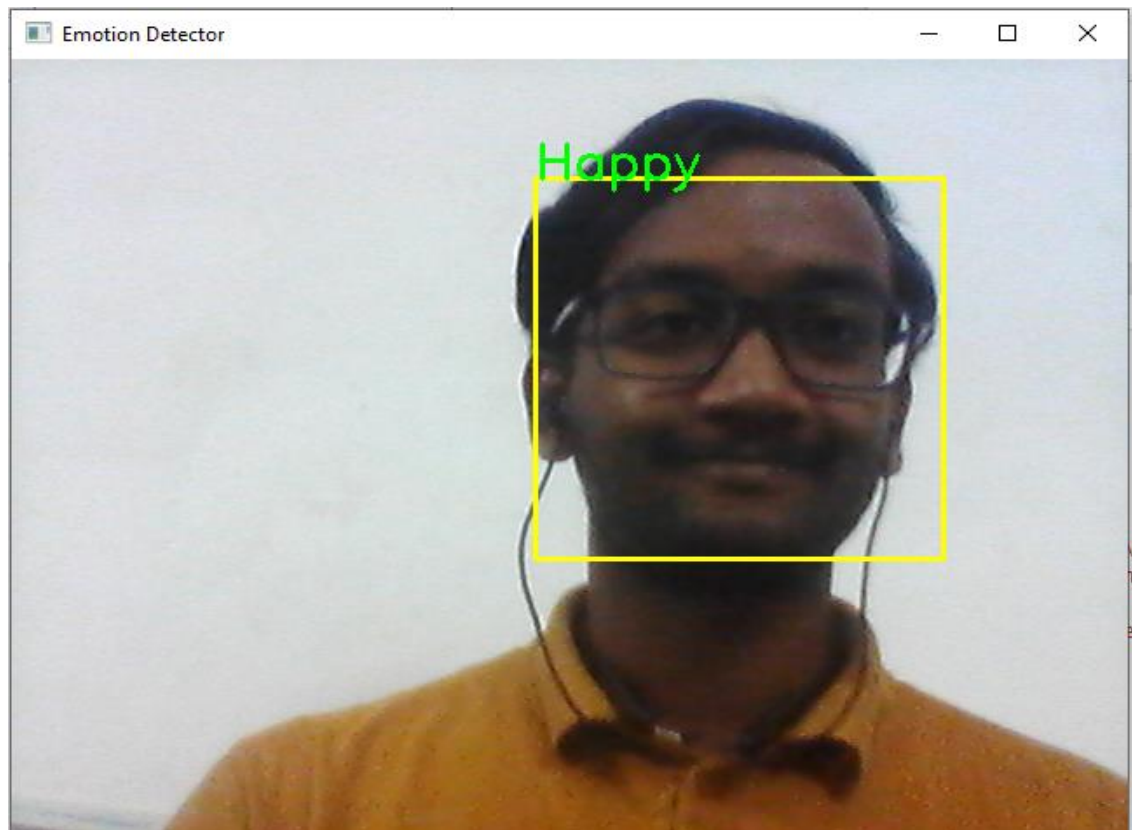
In [ ]:
```

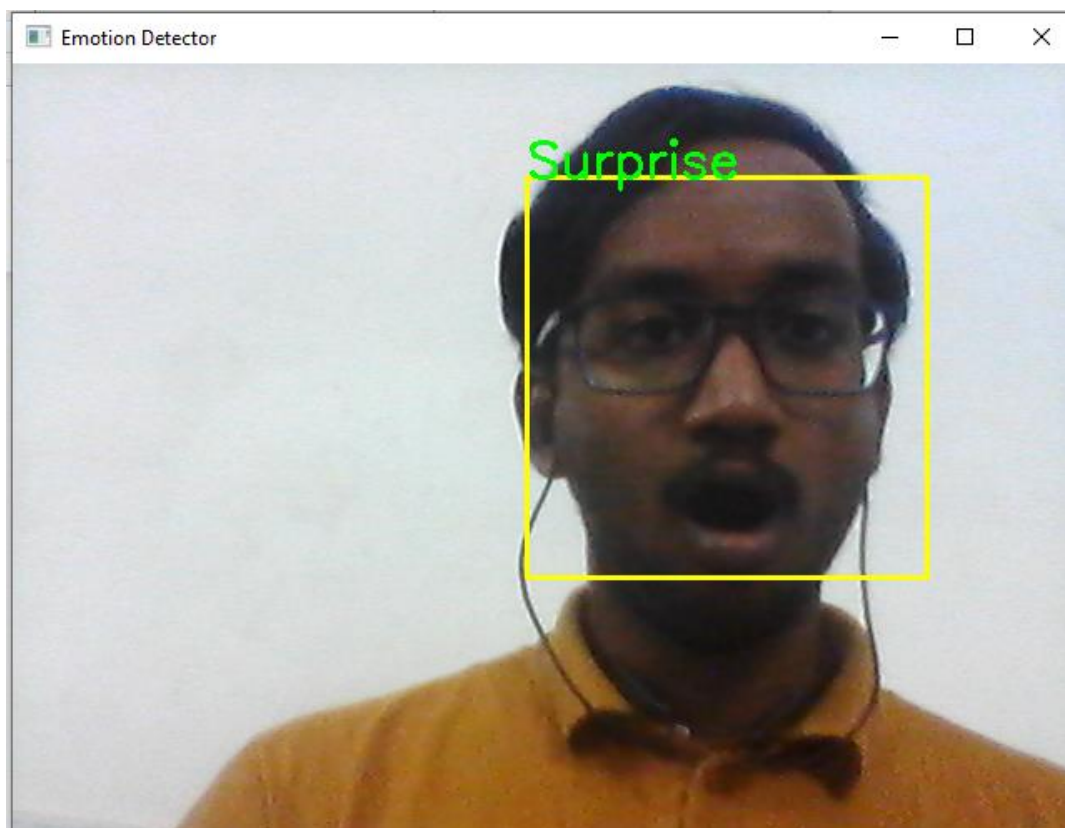
Chapter – 5

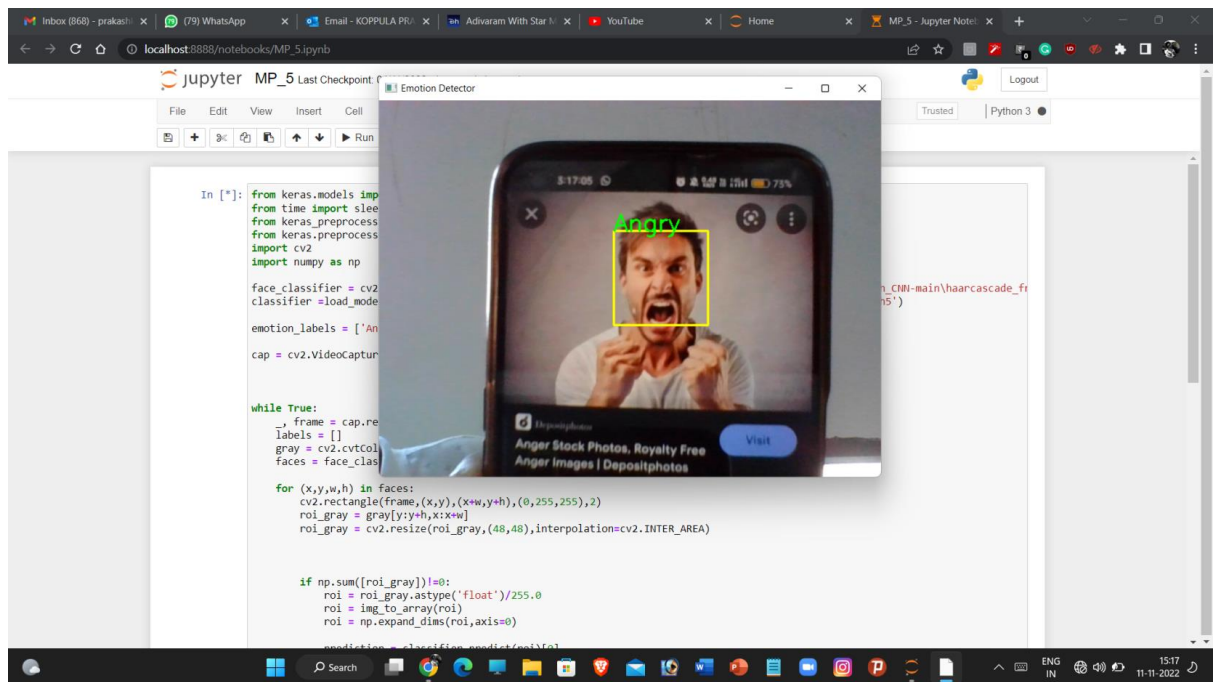
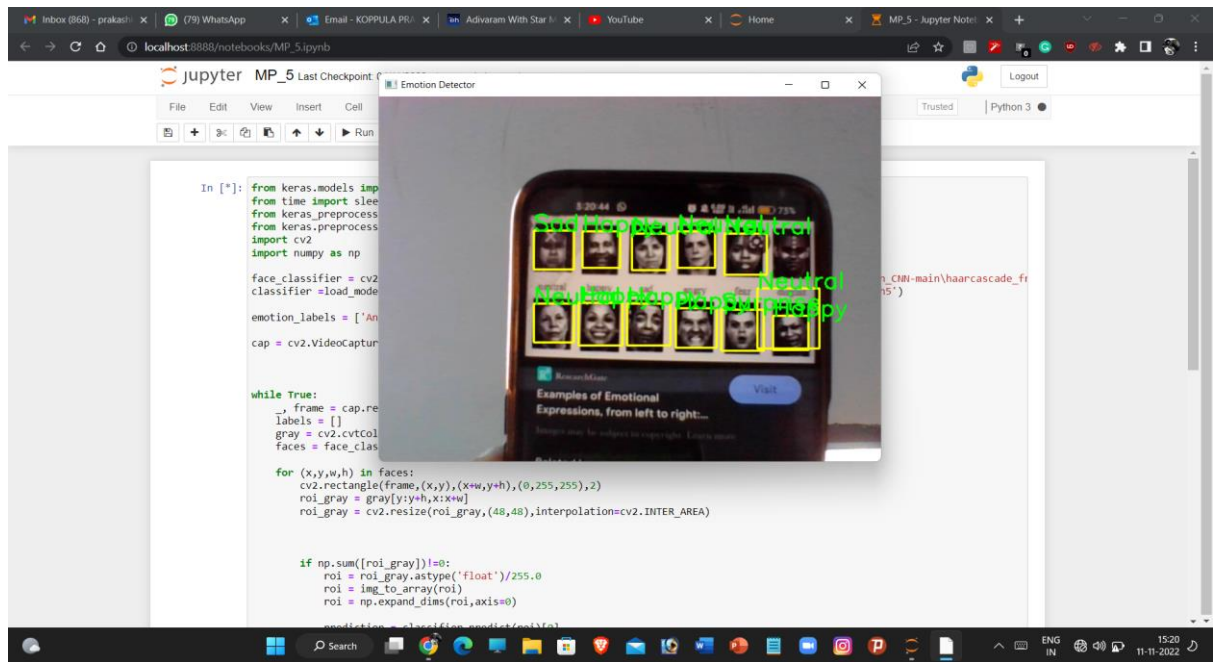
RESULT ANALYSIS AND VALIDATION

1. Test Samples

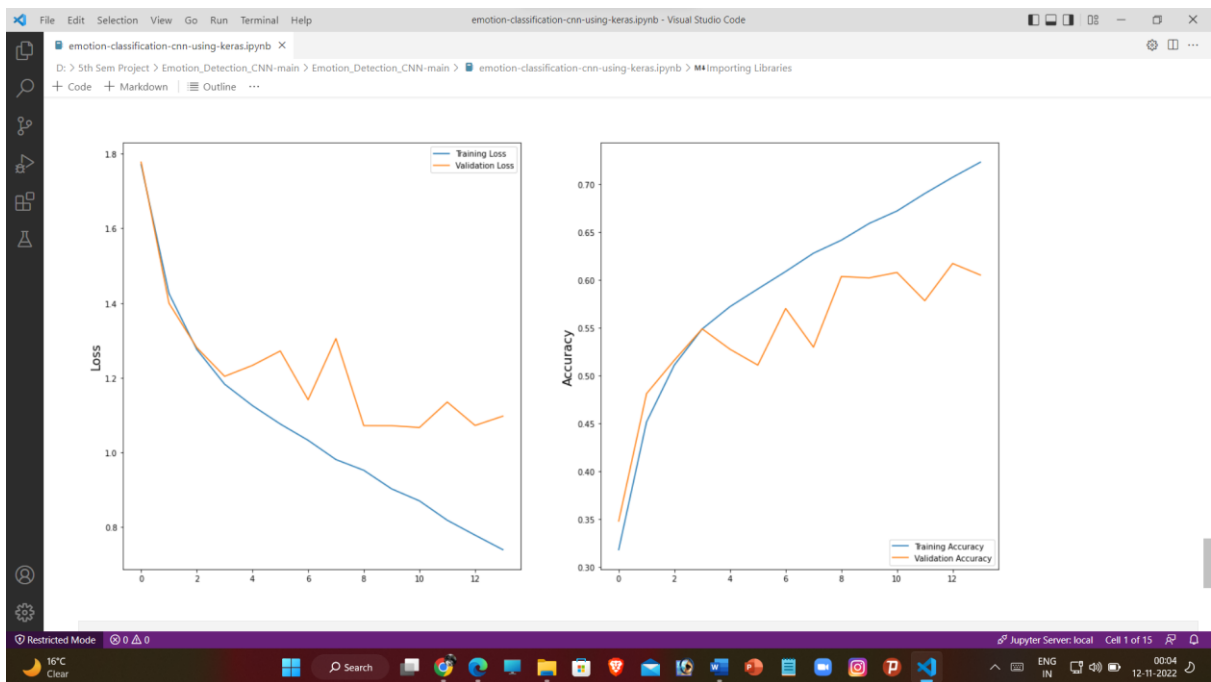








2. Loss and Accuracy Graph



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 – 6

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion:

In this project, we studied the what are the techniques used previously which mentioned in literature survey, what are the real time application for the proposed system which covered in objective and motivation.

By using the mentioned dataset, the neural network will be trained and by adding maximum hidden layer i.e., with deep layer in convolutional network the result will be calculated.

It covers the concept of facial expression recognition with aimed to classify images of faces into any of seven discrete emotion or face expression categories that represent universal human emotions.

5.2 Future Scope:

Facial emotion recognition is an emerging field so considering other NNs such as Recurrent Neural Networks (RNNs) may improve the accuracy.

The feature extraction is like pattern recognition which is used in intelligence, military and forensics for identification purposes. Thus, techniques such as the Capsnet algorithm for pattern recognition can be considered.

DL based approaches require a large labeled dataset, significant memory and long training and testing times which makes them difficult to implement on mobile and other platforms with limited resources.

Thus, simple solutions should be developed with lower data and memory requirements.

It is useful and important for security and healthcare purposes. Also, it is crucial for easy and simple detection of human feelings at a specific moment without asking them.

Chapter – 7

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., & Martínez 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>
- [6] https://www.ripublication.com/ijaer18/ijaerv13n8_119.pdf
- [7] <https://peltarion.com/knowledge-center/modeling-view/build-an-ai-model/blocks/2d-convolution>
- [8] <https://www.upgrad.com/blog/basic-cnn-architecture/>
- [9] https://www.researchgate.net/figure/Visualization-of-a-fully-connected-layer-Taken-from-Holleman-72_fig4_336607800
- [10] https://www.researchgate.net/figure/A-pair-of-convolution-and-pooling-layers-in-the-CNN-architecture_fig1_323233791
- [11] <https://en.wikipedia.org/wiki/NumPy>
- [12] [https://en.wikipedia.org/wiki/Pandas_\(software\)](https://en.wikipedia.org/wiki/Pandas_(software))
- [13] <https://en.wikipedia.org/wiki/TensorFlow>
- [14] <https://matplotlib.org/stable/gallery/misc/logos2.html>