

REAL-TIME EMOTION RECOGNITION FROM FACIAL EXPRESSION

A MINI PROJECT REPORT

ABDULLAH AMBERKHANI (311619104002)

ANUSH VENKATESH. V.G (311619104009)

HASHWANT SRIRAM (311619104026)

in partial fulfilment for the award of degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



MISRIMAL NAVAJEE MUNOTH JAIN ENGINEERING COLLEGE

THORAIPAKKAM, CHENNAI-600097

ANNA UNIVERSITY: CHENNAI 600025

JUNE 2022

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report “REAL-TIME EMOTION

RECOGNITION FROM FACIAL EXPRESSION “

is the bonafide work of “**ABDULLAH AMBERKHANI, ANUSH VENKATESH. V.G, HASHWANT SRIRAM** “who carried out the project work under my supervision.

SIGNATURE

Dr. N.Savaranan B.E., ME., Ph.D.

HEAD OF THE DEPARTMENT

Department of Computer Science

and Engineering

Misrimal Navajee Munoth Jain

Engineering College,

Thoraipakkam,

Chennai-600097

SIGNATURE

Ms. A.Jeyanthi M.E., (Ph.D.)

SUPERVISOR

ASSOCIATE PROFESSOR

Department of Computer Science

and Engineering

Misrimal Navajee Munoth Jain

Engineering College,

Thoraipakkam,

Chennai-600097

Submitted for the Mini Project Viva-Voce Examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

MISRIMAL NAVAJEE MUNOTH JAIN ENGINEERING COLLEGE, CHENNAI – 97

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

- **VISION**

Producing competent Computer Engineers with a strong background in the latest trends and technology to achieve academic excellence and to become pioneer in software and hardware products with an ethical approach to serve the society.

- **MISSION**

To provide quality education in Computer Science and Engineering with the state of the art facilities. To provide the learning audience that helps the students to enhance problem solvingskills and to inculcate in them the habit of continuous learning in their domain of interest. To serve the society by providing insight solutions to the real world problems by employing the latest trends of computing technology with strict adherence to professional and ethical responsibilities.

ACKNOWLEDGEMENT

We express our sincere gratitude to our honorable Secretary (Administration) **Dr. Harish L Metha** and Secretary (Academic) **Shri L. Jaswant S Munoth** for providing the infrastructure facilities to do this project during our course period.

We thank our **Principal, Dr. C. C. CHRISTOPHER, M.E., Ph.D.**, for his valuable suggestions and guidance for the development and completion of this project.

We express profound sense of gratitude and thanks to our **HEAD OF THE DEPARTMENT, Dr. N. SARAVANAN, B.E., M.E., Ph.D.** for her valuable suggestions and guidance for the development and completion of this project.

We thank our supervisor **Ms. A.Jeyanthi, M.E., (Ph.D.), Associate Professor** for her constant help and valuable suggestions. We thank her for her wholehearted guidance right from the inception to the completion of our project.

ABSTRACT

Facial Expression conveys non-verbal cues, which plays an important role in interpersonal relations. The Facial Expression Recognition system is the process of identifying the emotional state of a person. In this system captured image is compared with the trained dataset available in database and then emotional state of the image will be displayed. This system is based on image processing and machine learning.

We built several models capable of recognizing seven basic emotions (happy, sad, angry, afraid, surprise, disgust, and neutral) from facial expressions. Using the FER-2013 dataset of labelled headshots, and 66.67% using a CNN. We then transferred the skills learned on static images into a real-time emotion recognition system, which continuously detects faces from a video feed and classifies the predominant emotion of the individual.

Though the real-time system can reliably classify some of the seven emotions, more work is necessary to build a robust real-time system that performs outside of laboratory conditions.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	iv
1	INTRODUCTION	1
2	REQUIREMENT ANALYSIS	2
	2.1 SOFTWARE REQUIREMENTS	2
	2.2 HARDWARE REQUIREMENTS	2
3	TECHNIQUES USED	3
	3.1 CONVOLUTIONAL NEURAL NETWORK (CNN)	3
	3.2 COLABORATORY	4
	3.3 TENSORFLOW	6
	3.4 NUMPY	7
	3.5 ARTIFICIAL INTELLIGENCE	8
	3.6 MACHINE LEARNING	9
	3.7 COMPUTER VISION	9
4	SYSTEM ARCHITECTURE	10
5	IMPLEMENTATION MODDULES	11
	5.1 DATA PREPROCESSING	11
	5.2 CNN	12
	5.3 TRAIN	13
	5.4 REAL-TIME CLASSIFICATION	14

6	CONCLUSION	15
	APPENDICES	16
	APPENDICE 1-SAMPLE CODING	16
	APPENDICE 2-SCREENSHOT	21

CHAPTER 1

INTRODUCTION

A Facial expression is the visible manifestation of the affective state, cognitive activity, intention, personality and psychopathology of a person and plays a communicative role in interpersonal relations. It has been studied for a long period of time and obtaining the progress recent decades. Though much progress has been made, recognizing facial expression with a high accuracy remains to be difficult due to the complexity and varieties of facial expressions. Generally human beings can convey intentions and emotions through nonverbal ways such as gestures, facial expressions and involuntary languages. This system can be significantly useful, nonverbal way for people to communicate with each other. The important thing is how fluently the system detects or extracts the facial expression from image. The system is growing attention because this could be widely used in many fields like lie detection, medical assessment and human computer interface. The Facial Action Coding System (FACS), which was proposed in 1978 by Ekman and refined in 2002, is a very popular facial expression analysis tool.

We tackled the problem of recognizing the emotion of a person from an image of their facial expression. First, we built models capable of recognizing seven emotions (Happy, sad, angry, afraid, surprise, disgust, and neutral). Given static, cropped headshots, our model would output a probability distribution over emotions of the pictured individual. Next, we transferred the skills learned on static datasets to implement a real-time emotion classifier. Using a webcam video feed, we built a system to continuously detect faces, extract, crop, and grayscale the face region, and classify the emotion of the person.

CHAPTER 2

REQUIREMENT ANALYSIS

The requirement specification is a technical specification of requirements for the software products. The purpose of the software requirement specification is to provide a detailed overview of the software project, its parameter and goal. It describes the project target audience and its user interface, hardware and software requirements.

2.1 SOFTWARE REQUIREMENT

The software requirements may a detailed description of the system and all its features.

- Python
- Colaboratory
- TensorFlow
- NumPy

2.2 HARDWARE REQUIREMENT

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete engineer as the starting point for the system design.

- Ram: 8GB Ram or more
- Processor: Any Intel Processor
- GPU: 4GB or more
- Hard Disk: 10GB or more
- Speed: 1.4GHZ or more

CHAPTER 3

TECHNIQUES USED

3.1 CONVOLUTIONAL NEURAL NETWORK (CNN)

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of artificial neural network (ANN), most commonly applied to analyse visual imagery. CNNs are also known as Shift Invariant or Space Invariant Artificial Neural Networks (SIANN), based on the shared-weight architecture of the convolution kernels or filters that slide along input features and provide translation-equivariant responses known as feature maps. Counter-intuitively, most convolutional neural networks are not invariant to translation, due to the down sampling operation they apply to the input. They have applications in image and video recognition, recommender systems, image classification, image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial time series.

CNNs are regularized versions of multilayer perceptron's. Multilayer perceptron's usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "full connectivity" of these networks makes them prone to overfitting data. Typical ways of regularization, or preventing overfitting, include penalizing parameters during training (such as weight decay) or trimming connectivity (skipped connections, dropout, etc.) CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in their filters. Therefore, on a scale of connectivity and complexity, CNNs are on the lower extreme.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns to optimize the filters (or kernels) through automated learning, whereas in traditional algorithms these filters are hand-engineered. This independence from prior knowledge and human intervention in feature extraction is a major advantage.

3.2 COLABORATORY

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs. Colab resources are not guaranteed and not unlimited, and the usage limits sometimes fluctuate. This is necessary for Colab to be able to provide resources for free.

Colab notebooks are stored in Google Drive or can be loaded from GitHub. Colab notebooks can be shared just as you would with Google Docs or Sheets. Simply click the Share button at the top right of any Colab notebook or follow these Google Drive file sharing instructions. Code is executed in a virtual machine private to your account. Virtual machines are deleted when idle for a while, and have a maximum lifetime enforced by the Colab service.

Google Drive imposes a limit on how much data can be stored in it by each user. If Drive operations are failing with Input/output error and a notification says storage quota has been exceeded, delete some files using drive.google.com and Empty your Trash to reclaim the space. It might take a little while for the reclaimed space to be available in Colab.

In order to be able to offer computational resources for free, Colab needs to maintain the flexibility to adjust usage limits and hardware availability on the fly. Resources available in Colab vary over time to accommodate fluctuations in demand, as well as to accommodate overall growth and other factors.

Some users want to be able to do more in Colab than the resource limits allow. We have heard from many users who want faster GPUs, longer running 18 notebooks and more memory, as well as usage limits that are higher and don't fluctuate as much.

Colab can provide free resources in part by having dynamic usage limits that sometimes fluctuate, and by not providing guaranteed or unlimited resources. This means that overall usage limits as well as idle timeout periods, maximum VM lifetime, GPU types available, and other factors vary over time. Colab does not publish these limits, in part because they can (and sometimes do) vary quickly. The amount of memory available in Colab virtual machines varies over time (but is stable for the lifetime of the VM).

3.3 TENSORFLOW

TensorFlow is an open-source end-to-end platform for creating Machine Learning applications. It is a symbolic math library that uses dataflow and differentiable programming to perform various tasks focused on training and inference of deep neural networks. It allows developers to create machine learning applications using various tools, libraries, and community resources.

It enables you to build dataflow graphs and structures to define how data moves through a graph by taking inputs as a multi-dimensional array called Tensor. It allows you to construct a flowchart of operations that can be performed on these inputs, which goes at one end and comes at the other end as output.

It is called Tensorflow because it takes input as a multi-dimensional array, also known as tensors. You can construct a sort of flowchart of operations (called a Graph) that you want to perform on that input. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output. Deep learning relies on a lot of matrix multiplication.

TensorFlow is very fast at computing the matrix multiplication because it is written in C++. Although it is implemented in C++, TensorFlow can be accessed and controlled by other languages mainly, Python.

Finally, a significant feature of TensorFlow is the TensorBoard. The TensorBoard enables to monitor graphically and visually what TensorFlow is doing.

3.4 NUMPY

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project, and you can use it freely. NumPy stands for Numerical Python.

In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently. This behaviour is called locality of reference in computer science. This is the main reason why NumPy is faster than lists. Also, it is optimized to work with latest CPU architectures. NumPy is a Python library and is written partially in Python, but most of the parts that require fast computation are written in C or C++.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents.

NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops, using NumPy.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs if most operations work on arrays or matrices instead of scalars.

The core functionality of NumPy is its "ndarray", for n-dimensional array, data structure. These arrays are strided views on memory. In contrast to Python's built-in list data structure, these arrays are homogeneously typed: all elements of a single array must be of the same type.

3.5 ARTIFICIAL INTELLIGENCE

Artificial intelligence is a branch of computer science that aims to create intelligent machines. It is the simulation of human intelligence processes by machines, especially computer systems. These processes include learning (the acquisition of information and rules for using the information), reasoning (using rules to reach approximate or definite conclusions) and self-correction.

Artificial intelligence (AI) makes it possible for machines to learn from experience, adjust to new inputs and perform human-like tasks. Most AI examples that you hear about today – from chess-playing computers to self-health evaluation system – rely heavily on machine learning and deep learning. Using these technologies, computers can be trained to accomplish specific tasks by processing large amount of data and recognizing patterns in the data.

3.6 MACHINE LEARNING

Machine learning (ML) is the scientific study of algorithms and statistical models that computer uses to perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning (ML) algorithms build a mathematical model based on sample data, known as “training data”, in order to make predictions or decisions without being explicitly programmed to perform the task. Machine Learning (ML) is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning.

3.7 COMPUTER VISION:

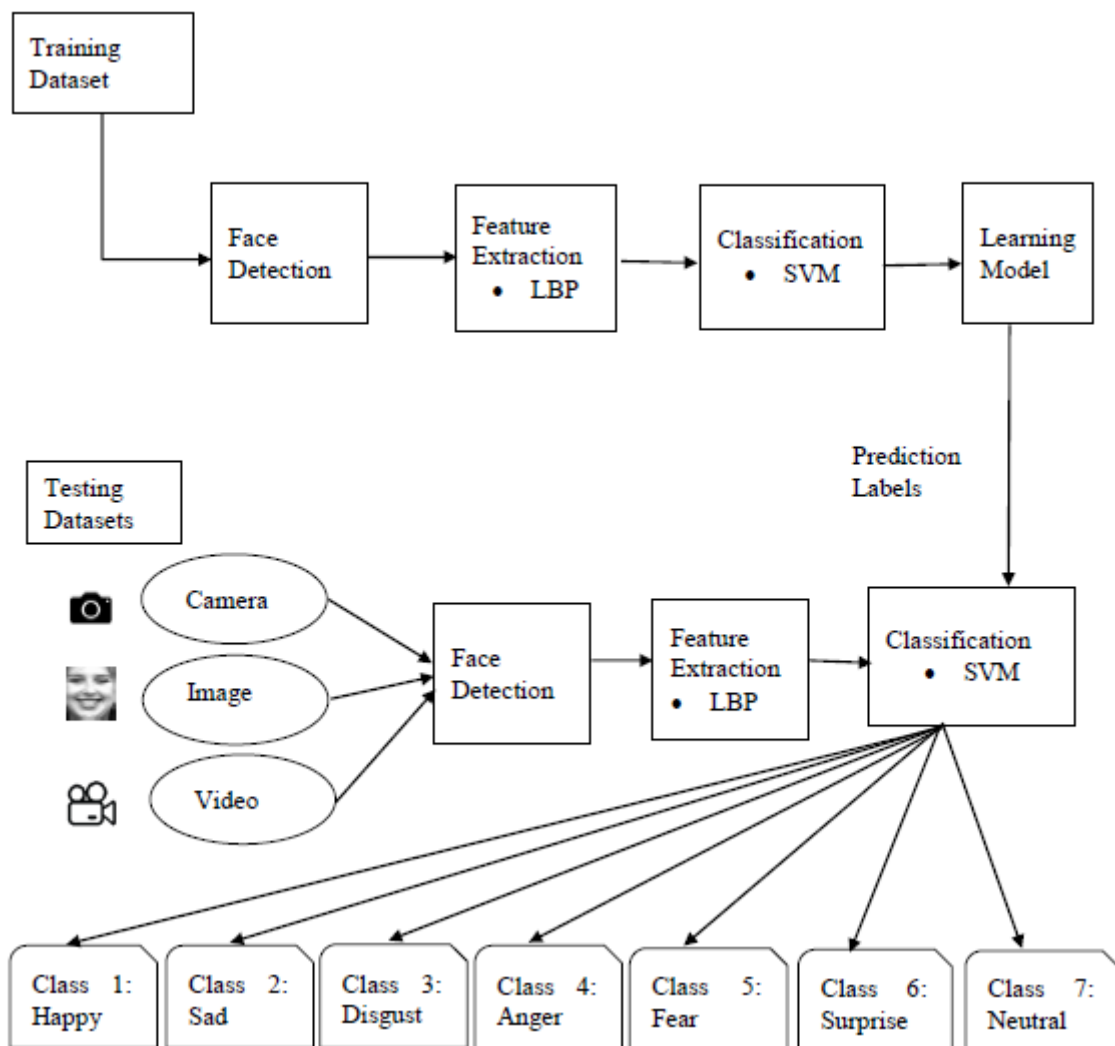
Computer Vision, often abbreviated as CV, is defined as a field of study that seeks to develop techniques to help computers “see” and understand the content of digital images such as photographs and videos. Computer vision is a field of study focused on the problem of helping computers to see. At an abstract level, the goal of computer vision problems is to use the observed image data to infer something about the world. It is a multidisciplinary field that could broadly be called a subfield of artificial intelligence and machine learning, which may involve the use of specialized methods and make use of general learning algorithms. convolutional neural network processes an image, each of its layers extracts specific features from the pixels. The first layer detects very basic things, such as vertical and horizontal edges. As you move deeper into the neural network, the layers detect more-complex features, including corners and shapes. The final layers of the CNN detect specific things such as faces, doors, and cars. The output layer of the CNN provides a table of numerical values representing the probability that a specific object was discovered in the image.

CHAPTER 4

SYSTEM ARCHITECTURE

4.1 OVERVIEW OF ARCHITECTURE

The diagram shows the overall architecture of the project which consists of the web pages and server block. Each block shows their structural and functional components of the project.



CHAPTER 5

IMPLEMENTATION MODULES

5.1.1 DATA PREPROCESSING

A dataset can be viewed as a collection of data objects, which are often also called as a records, points, vectors, patterns, events, cases, samples, observations, or entities. Data objects are described by several features, that capture the basic characteristics of an object, such as the mass of a physical object or the time at which an event occurred, etc. Features are often called as variables, characteristics, fields, attributes, or dimensions. Once the data is collected, it's time to assess the condition of it, including looking for trends, outliers, exceptions, incorrect, inconsistent, missing, or skewed information.

This is important because your source data will inform all your model's findings, so it is critical to be sure it does not contain unseen biases. For example, if you are looking at customer behaviour nationally, but only pulling in data from a limited sample, you might miss important geographic regions. This is the time to catch any issues that could incorrectly skew your model's findings, on the entire data set, and not just on partial or sample data sets.

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task.

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

5.2 CONVOLUTION NEURAL NETWORK (CNN)

A Convolutional Neural Network is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand engineered, with enough training, ConvNets can learn these filters/characteristics.

We evaluated both a variety of pre-processing techniques as well as several model architectures, ultimately developing a custom CNN model capable of attaining near-state-of-the-art accuracy of 66.67% on the FER-2013 test set.

We implemented several CNN architectures from papers applying emotion recognition to these and other datasets. Ultimately, what yielded the best performance was our custom developed CNN architecture (left). Analysing error in neural networks is infamously difficult. We analysed our error across different classes, as well as by visual inspection of images we classified correctly and incorrectly. One early observation was that we fail much more at certain emotions (see confusion matrix), and that we were failing to classify images where it was necessary to rely on fine details in the images (e.g., small facial features or curves). Due to this, we increased the number of layers and decreased filter sizes to increase the number of parameters in our network, which had a clear effect in allowing us to fit the dataset better.

This led to some overfitting, which we addressed by using dropout, early stopping around 100 epochs and augmenting our training set. Given this, we only start learning training set noise after achieving approx. 60% dev set accuracy; this is clear from plotting accuracy during training. This leaves us with some suggestions for future work, which largely focus on enabling increased parameterization of the network.

5.3 TRAIN

Looking at our training accuracy vs. test accuracy, our relatively low variance shows that we avoid overfitting if we stop around 100 epochs. Training beyond that consistently results in overfitting. Despite that, our model suffers from high bias; increasing the parameters in our network did not lead to more overfitting, so long as it was counteracted by adding dropout.

5.4 REAL-TIME CLASSIFICATION

We used OpenCV's Haar cascades to detect and extract a face region from a webcam video feed, then classified it using our CNN model. We found it best to neither subtract the training mean nor normalize the pixels in the detected face region before classifying it. Real-time

classification better exposed our model's strengths: neutral, happy, surprised, and angry were generally well-detected. Illumination was a very important factor in the model's performance. This suggests that our training set may not truthfully represent the distribution of lighting conditions.

CHAPTER 6

CONCLUSION

For continued work on this project, we believe there are two major areas of focus that would improve our real-time emotion recognition system. First, we suggest fine tuning the architecture of the CNN used for the model to fit perfectly with the problem at hand. Some examples of this fine tuning include finding and removing redundant parameters, adding new parameters in more useful places in the CNN's structure, adjusting the learning rate decay schedule, adapting the location and probability of dropout and experimenting to find ideal stride sizes.

A second area of focus lies in adapting the datasets to more closely reflect real-time recognition conditions. For example, simulating low light conditions and “noisy” image backgrounds, could help the model become more accurate in real-time recognition. Additionally making sure that the distribution of models in the training dataset accurately reflects the distribution of subjects that the system will see when running in real-time.

We noticed that our real-time recognition demo tended to perform better on Caucasian males than it did on other races or genders. We believe this is in part due to a skew in the training data towards white males and a more balanced dataset might be able to correct this skew.

More work is necessary to make the real-time system robust outside laboratory conditions, and it is possible that a deeper, more finely tuned CNN could improve results.

APPENDICE

APPENDICE 1

SAMPLE CODE

EMOTION DETECTION

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from keras.layers import Flatten, Dense
from keras.models import Model
from keras.preprocessing.image import ImageDataGenerator, img_to_array,
load_img
from keras.applications.mobilenet import MobileNet, preprocess_input
from keras.losses import categorical_crossentropy

# Working with pre trained model

base_model = MobileNet( input_shape=(224,224,3), include_top= False )

for layer in base_model.layers:
    layer.trainable = False

x = Flatten()(base_model.output)
x = Dense(units=7 , activation='softmax' )(x)

# creating our model.
model = Model(base_model.input, x)
model.compile(optimizer='adam', loss= categorical_crossentropy ,
metrics=['accuracy'] )

train_datagen = ImageDataGenerator(
    zoom_range = 0.2,
    shear_range = 0.2,
    horizontal_flip=True,
    rescale = 1./255
)

train_data = train_datagen.flow_from_directory(directory= "/content/train",
                                              target_size=(224,224),
                                              batch_size=32,
                                              )

train_data.class_indices
```

```

val_datagen = ImageDataGenerator(rescale = 1./255 )

val_data = val_datagen.flow_from_directory(directory= "/content/test",
                                          target_size=(224,224),
                                          batch_size=32,
                                          )

# to visualize the images in the traing data denerator

t_img , label = train_data.next()

#-----
# function when called will prot the images
def plotImages(img_arr, label):
    """
    input :- images array
    output :- plots the images
    """
    count = 0
    for im, l in zip(img_arr,label) :
        plt.imshow(im)
        plt.title(im.shape)
        plt.axis = False
        plt.show()

        count += 1
        if count == 10:
            break

#-----
# function call to plot the images
plotImages(t_img, label)
## having early stopping and model check point

from keras.callbacks import ModelCheckpoint, EarlyStopping

# early stopping
es = EarlyStopping(monitor='val_accuracy', min_delta= 0.01 , patience= 5,
verbose= 1, mode='auto')

# model check point
mc = ModelCheckpoint(filepath="best_model.h5", monitor= 'val_accuracy',
verbose= 1, save_best_only= True, mode = 'auto')

# puting call back in a list
call_back = [es, mc]
hist = model.fit_generator(train_data,
                          steps_per_epoch= 10,
                          epochs= 30,

```

```

        validation_data= val_data,
        validation_steps= 8,
        callbacks=[es,mc])

# Loading the best fit model
from keras.models import load_model
model = load_model("/content/best_model.h5")
plt.plot(h['accuracy'])
plt.plot(h['val_accuracy'] , c = "red")
plt.title("acc vs v-acc")
plt.show()
plt.plot(h['loss'])
plt.plot(h['val_loss'] , c = "red")
plt.title("loss vs v-loss")
plt.show()
# just to map o/p values
op = dict(zip( train_data.class_indices.values(),
train_data.class_indices.keys()))
# path for the image to see if it predicts correct class

path = "/content/test/angry/PrivateTest_1054527.jpg"
img = load_img(path, target_size=(224,224) )

i = img_to_array(img)/255
input_arr = np.array([i])
input_arr.shape

pred = np.argmax(model.predict(input_arr))

print(f" the image is of {op[pred]}")

# to display the image
plt.imshow(input_arr[0])
plt.title("input image")
plt.show()

```


REAL-TIME CLASSIFICATION

```
import os
import cv2
import numpy as np
from keras.preprocessing import image
import warnings
warnings.filterwarnings("ignore")
from keras.preprocessing.image import load_img, img_to_array
from keras.models import load_model
import matplotlib.pyplot as plt
import numpy as np

# load model
model = load_model("best_model.h5")

face_haar_cascade = cv2.CascadeClassifier(cv2.data.harcascades +
'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_BGR2RGB)

    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=7)
        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, (224, 224))
        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

```

APPENDICE 2

Datasets Collection

Table 6: Dataset images of facial recognition

Happy			
Sad			
Angry			
Surprise			
Disgust			
Fear			
Neutral			

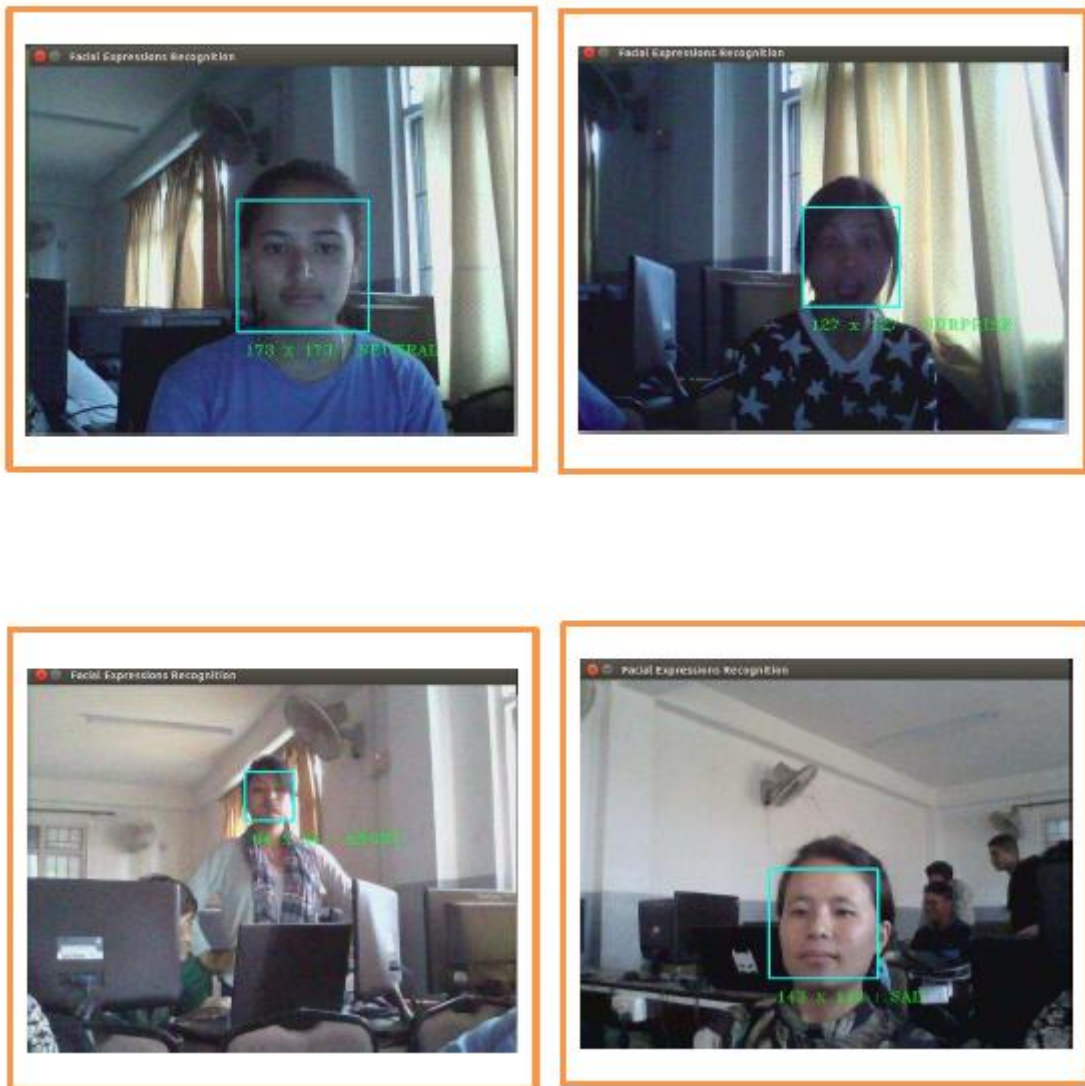
Experimental Demonstration

Experimental Demonstration from Image File



Figure 12: Experimental Demonstration from Image File

Experimental Demonstration from Camera



COs, PO Mapping, PSO Mapping

Course Code &Name :

REGULATION: R2017

YEAR/SEM: III/VI

COURSE OUTCOMES

CS8611.1	Identify the problem by applying acquired knowledge.
CS8611.2	Analyze and categorize executable project modules after considering risks.
CS8611.3	Choose efficient tools for designing project modules.
CS8611.4	Combine all the modules through effective team work after efficient testing.
CS8611.5	Elaborate the completed task and compile the project report.

CORRELATION LEVELS

Substantial/ High	3
Moderate/ Medium	2
Slight/ Low	1
No correlation	

CO – PSO CORRELATION LEVEL MATRIX

COs	PSO1	PSO2
CS8611.1	3	1
CS8611.2		2
CS8611.3		3
CS8611.4		3
CS8611.5		

CO-PO CORRELATION LEVEL MATRIX

COs	POs											
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CS8611.1	3	3		1			1	2	3			
CS8611.2		3	2	3		3	1		3	3		2
CS8611.3			3	3	3			3	3	3	1	2
CS8611.4							3	3	3	3	2	3
CS8611.5									3	3		3