LAB # 6

Session 2023-2027

BS in Artifical Intelligence



Department of Software Engineering

Faculty of Computer Science & Information Technology

The Superior Univeristy, Lahore

**NAME**: Saad Waleed

**ROLL NO:** 073

Contents

[LAB # 6 i](#_Toc193121653)

[Session 2023-2027 i](#_Toc193121654)

[BS in Artifical Intelligence i](#_Toc193121655)

[List of Figures iii](#_Toc193121656)

[Chapter 1 2](#_Toc193121657)

[Introduction 2](#_Toc193121658)

[Chapter 2 3](#_Toc193121659)

[Tool & Technology 3](#_Toc193121660)

[Laptop Specifications 3](#_Toc193121661)

[Development Environment 3](#_Toc193121662)

[Libraries and Frameworks 3](#_Toc193121663)

[System Workflow 4](#_Toc193121664)

[CHAPTER 3 5](#_Toc193121665)

[Implementation Code 5](#_Toc193121666)

[1. Training Model (trainmodel.ipynb) 5](#_Toc193121667)

[Step 1: Import Libraries and Define Functions 5](#_Toc193121668)

[Step 2: Create DataFrames for Training and Test Data 5](#_Toc193121669)

[Step 3: Prepare Training and Test Data 6](#_Toc193121670)

[Step 4: Extract Features from Images 6](#_Toc193121671)

[Step 5: Encode Labels and Prepare for Model Training 7](#_Toc193121672)

[Step 6: Build the CNN Model 7](#_Toc193121673)

[Step 7: Compile and Train the Model 8](#_Toc193121674)

[Step 8: Save the Model 8](#_Toc193121675)

[Step 9: Test the Model with an Image 8](#_Toc193121676)

[2. Detecting Face (detecting face.py) 9](#_Toc193121677)

[Import Libraries and Load Model 9](#_Toc193121678)

[Set Up Face Detection 9](#_Toc193121679)

[Define Helper Function 10](#_Toc193121680)

[Real-Time Emotion Detection 10](#_Toc193121681)

[Explanation: 11](#_Toc193121682)

[11](#_Toc193121683)

[Chapter 4 12](#_Toc193121684)

[Result 12](#_Toc193121685)

[Conclusion 13](#_Toc193121686)

[Why I Chose Emotion Detection 13](#_Toc193121687)

[Future Applications 13](#_Toc193121688)

# 

# List of Figures

[Figure 1 13](#_Toc184650220)

# Chapter 1

# Introduction

Human emotions are fundamental to effective communication and social interaction. Understanding and interpreting these emotions has become a focal point in the field of Artificial Intelligence (AI), with applications spanning healthcare, customer service, education, and entertainment. Emotion detection systems are at the intersection of computer vision and deep learning, offering automated analysis of facial expressions to classify emotions such as happiness, sadness, Human anger, and surprise.

This project, Emotion Detection System, is designed to perform real-time emotion recognition by analyzing facial expressions captured via a webcam. It leverages a pre-trained deep learning model to classify emotions into seven categories: angry, disgust, fear, happy, neutral, sad, and surprise. The system is implemented in two phases:

1. **Model Training Phase**: A neural network is trained using a labeled dataset of facial expressions. The model's weights and architecture are saved for deployment in the detection phase.
2. **Real-Time Detection Phase**: This component uses the trained model to process live video input, detecting faces in real time and predicting their emotional states. Detected faces are also saved for further analysis.

By combining deep learning techniques with computer vision tools, this system aims to create a versatile, efficient, and scalable solution for emotion recognition. Its practical applications include monitoring emotional well-being, enhancing virtual learning environments, and building

# Chapter 2

# Tool & Technology

## ****Laptop Specifications****

* **Model**: Dell Latitude 5340
* **Processor**: 11th Gen Intel Core
* **RAM**: 8 GB
* **Storage**: 256 GB SSD

The laptop serves as the development and deployment platform, equipped to handle computationally intensive tasks such as model training and real-time video processing.

## ****Development Environment****

1. **Python Programming Language**: The project is implemented in Python, chosen for its extensive library support, flexibility, and integration capabilities for machine learning tasks.
2. **Integrated Development Tools**: Jupyter Notebook is used for model training and testing, while standard Python scripts handle real-time detection.

## ****Libraries and Frameworks****

1. **TensorFlow and Keras**:
   * TensorFlow is used to build and train the convolutional neural network (CNN).
   * Keras, a high-level API of TensorFlow, simplifies the process of designing, training, and deploying deep learning models.
2. **Pandas**:
   * Used for dataset manipulation and preprocessing, ensuring that input data is clean and correctly formatted for training.
3. **NumPy**:
   * Provides support for array operations, essential for numerical computations and data handling during model training.
4. **OpenCV**:
   * Utilized for face detection and real-time video processing using Haar cascades.
   * Handles image resizing, grayscaling, and ROI extraction for emotion classification.
5. **scikit-learn**:
   * Assists in preprocessing tasks such as data normalization and splitting into training and testing sets.
6. **tqdm**:
   * Adds progress bars to loops, improving clarity and monitoring during training and data preprocessing.

## ****System Workflow****

1. **Model Training**:
   * The model is trained on 48x48 grayscale images of facial expressions.
   * Convolutional layers are used to extract features, followed by dense layers for classification.
   * The weights are stored in a file (emotiondetector.h5) for deployment.
2. **Real-Time Detection**:
   * Video input is captured using OpenCV, with faces detected via Haar cascades.
   * Detected faces are resized and normalized to match the model's input requirements.
   * The trained model predicts the emotion for each detected face, and the result is displayed as a label on the video feed.

# CHAPTER 3

# Implementation Code

# 1. Training Model (trainmodel.ipynb)

## ****Step 1: Import Libraries and Define Functions****

from keras.utils import to\_categorical # type: ignore

from keras\_preprocessing.image import load\_img # type: ignore

from keras.models import Sequential # type: ignore

from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D # type: ignore

import os

import pandas as pd # type: ignore

import numpy as np # type: ignore

**Explanation**:

1. **Keras** libraries are imported for creating and training the deep learning model.
2. **Pandas and NumPy** are imported for handling data and performing mathematical operations.
3. **Matplotlib** is included for visualizations.

## ****Step 2: Create DataFrames for Training and Test Data****

def createdataframe(dir):

image\_paths = []

labels = []

for label in os.listdir(dir):

for imagename in os.listdir(os.path.join(dir,label)):

image\_paths.append(os.path.join(dir,label,imagename))

labels.append(label)

print(label, "completed")

return image\_paths, labels

**Explanation**:

1. This function creates a list of image paths and their associated labels. It goes through each subfolder (representing an emotion) in the provided directory and appends the image path and the corresponding emotion label.
2. The function then returns two lists: image\_paths and labels.

## ****Step 3: Prepare Training and Test Data****

TRAIN\_DIR = r'C:\Users\Admin\Desktop\images\train'

train = pd.DataFrame()

train['image'], train['label'] = createdataframe(TRAIN\_DIR)

print(train)

TEST\_DIR = r'C:\Users\Admin\Desktop\images\test'

test = pd.DataFrame()

test['image'], test['label'] = createdataframe(TEST\_DIR)

print(test)

**Explanation**:

1. The TRAIN\_DIR and TEST\_DIR paths specify the directories where the training and testing images are stored.
2. The createdataframe() function is called to populate train and test DataFrames with image paths and their corresponding labels.

## ****Step 4: Extract Features from Images****

from tqdm.notebook import tqdm # type: ignore

def extract\_features(images):

features = []

for image in tqdm(images):

img = load\_img(image, grayscale=True)

img = np.array(img)

features.append(img)

features = np.array(features)

features = features.reshape(len(features), 48, 48, 1)

return features

train\_features = extract\_features(train['image'])

test\_features = extract\_features(test['image'])

x\_train = train\_features/255.0

x\_test = test\_features/255.0

**Explanation**:

1. The extract\_features() function loads each image, converts it to grayscale, and appends it to a list of features. Each image is resized to 48x48 pixels, as this is the required input size for the model.
2. The images are then normalized by dividing each pixel value by 255. This scales pixel values to the range [0, 1].

## ****Step 5: Encode Labels and Prepare for Model Training****

from sklearn.preprocessing import LabelEncoder # type: ignore

le = LabelEncoder()

le.fit(train['label'])

y\_train = le.transform(train['label'])

y\_test = le.transform(test['label'])

y\_train = to\_categorical(y\_train, num\_classes=7)

y\_test = to\_categorical(y\_test, num\_classes=7)

**Explanation**:

1. The LabelEncoder() is used to convert string labels (emotions) into numeric values.
2. The labels are then one-hot encoded using to\_categorical() to prepare them for multi-class classification. There are 7 emotions, hence num\_classes=7.

## ****Step 6: Build the CNN Model****

model = Sequential()

# convolutional layers

model.add(Conv2D(128, kernel\_size=(3,3), activation='relu', input\_shape=(48,48,1)))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Dropout(0.4))

model.add(Conv2D(256, kernel\_size=(3,3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Dropout(0.4))

model.add(Conv2D(512, kernel\_size=(3,3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Dropout(0.4))

model.add(Conv2D(512, kernel\_size=(3,3), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Dropout(0.4))

model.add(Flatten())

# fully connected layers

model.add(Dense(512, activation='relu'))

model.add(Dropout(0.4))

model.add(Dense(256, activation='relu'))

model.add(Dropout(0.3))

# output layer

model.add(Dense(7, activation='softmax'))

**Explanation**:

1. A **Sequential Model** is used to stack layers.
2. **Convolutional layers (Conv2D)** are added to extract features from the input images. Each Conv2D layer is followed by a **MaxPooling2D** layer to reduce spatial dimensions.
3. **Dropout** layers are included to prevent overfitting by randomly setting a fraction of input units to zero during training.
4. **Flatten** is used to convert the 2D feature maps into 1D vectors.
5. Two **Fully connected layers** are added (with Dense), followed by a **softmax output layer** to produce the probability distribution for the 7 emotions.

## ****Step 7: Compile and Train the Model****

model.compile(optimizer='adam',

loss='categorical\_crossentropy',

metrics=['accuracy'])

model.fit(x=x\_train, y=y\_train, batch\_size=128, epochs=100, validation\_data=(x\_test, y\_test))

**Explanation**:

1. The model is compiled with the **Adam optimizer** and **categorical cross-entropy** loss function, as this is a multi-class classification problem.
2. The model is trained on the training data for 100 epochs, with a batch size of 128. Validation is performed using the test set.

## ****Step 8: Save the Model****

model\_json = model.to\_json()

with open("emotiondetector.json", 'w') as json\_file:

json\_file.write(model\_json)

model.save("emotiondetector.h5")

**Explanation**:

1. After training, the model's architecture is saved as a .json file.
2. The model weights are saved as a .h5 file for future use during inference.

## ****Step 9: Test the Model with an Image****

image = r'C:\Users\Admin\Desktop\images\train\sad\42.jpg'

print("original image is of sad")

img = ef(image)

pred = model.predict(img)

pred\_label = label[pred.argmax()]

print("model prediction is ", pred\_label)

plt.imshow(img.reshape(48, 48), cmap='gray')

**Explanation**:

1. A test image is loaded and passed to the model for emotion prediction.
2. The predicted label is printed, and the image is displayed with the predicted emotion.

# ****2. Detecting Face (detecting face.py)****

## ****Import Libraries and Load Model****

import cv2

from keras.models import model\_from\_json # type: ignore

import numpy as np

import os

# Load the model architecture from JSON and weights from H5

json\_file = open("emotiondetector.json", "r")

model\_json = json\_file.read()

json\_file.close()

model = model\_from\_json(model\_json)

model.load\_weights("emotiondetector.h5")

**Explanation**:

1. cv2 is imported for handling computer vision tasks, such as face detection and video capture.
2. keras.models.model\_from\_json loads the model architecture from a .json file, while the load\_weights function loads pre-trained weights (emotiondetector.h5). This ensures the model is ready for inference.
3. Paths to the .json and .h5 files are hardcoded for direct access.

## ****Set Up Face Detection****

haar\_file = cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml'

face\_cascade = cv2.CascadeClassifier(haar\_file)

save\_directory = 'captured\_faces'

if not os.path.exists(save\_directory):

os.makedirs(save\_directory)

**Explanation**:

1. **Haar Cascade**: A pre-trained classifier (haarcascade\_frontalface\_default.xml) is loaded to detect faces.
2. **Save Directory**: Creates a directory (captured\_faces) for storing captured face images if it does not already exist.

## ****Define Helper Function****

def extract\_features(image):

feature = np.array(image)

feature = feature.reshape(1, 48, 48, 1)

return feature / 255.0

**Explanation**:

1. **Feature Extraction**: Converts an input image to a normalized numpy array for model compatibility.
2. The image is reshaped to (1, 48, 48, 1) to match the input dimensions expected by the model.

## ****Real-Time Emotion Detection****

webcam = cv2.VideoCapture(0)

labels = {0: 'angry', 1: 'disgust', 2: 'fear', 3: 'happy', 4: 'neutral', 5: 'sad', 6: 'surprise'}

image\_counter = 0

while True:

ret, im = webcam.read()

gray = cv2.cvtColor(im, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(im, 1.3, 5)

try:

for (p, q, r, s) in faces:

image = gray[q:q+s, p:p+r]

cv2.rectangle(im, (p, q), (p+r, q+s), (255, 0, 0), 2)

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

img = extract\_features(image)

pred = model.predict(img)

prediction\_label = labels[pred.argmax()]

cv2.putText(im, prediction\_label, (p-10, q-10), cv2.FONT\_HERSHEY\_COMPLEX\_SMALL, 2, (0, 0, 255))

cv2.imshow("Output", im)

if len(faces) > 0:

face\_image\_filename = os.path.join(save\_directory, f"face\_.jpg")

cv2.imwrite(face\_image\_filename, im[q:q+s, p:p+r])

image\_counter += 1

if cv2.waitKey(27) & 0xFF == ord('q'):

break

except cv2.error:

pass

webcam.release()

cv2.destroyAllWindows()

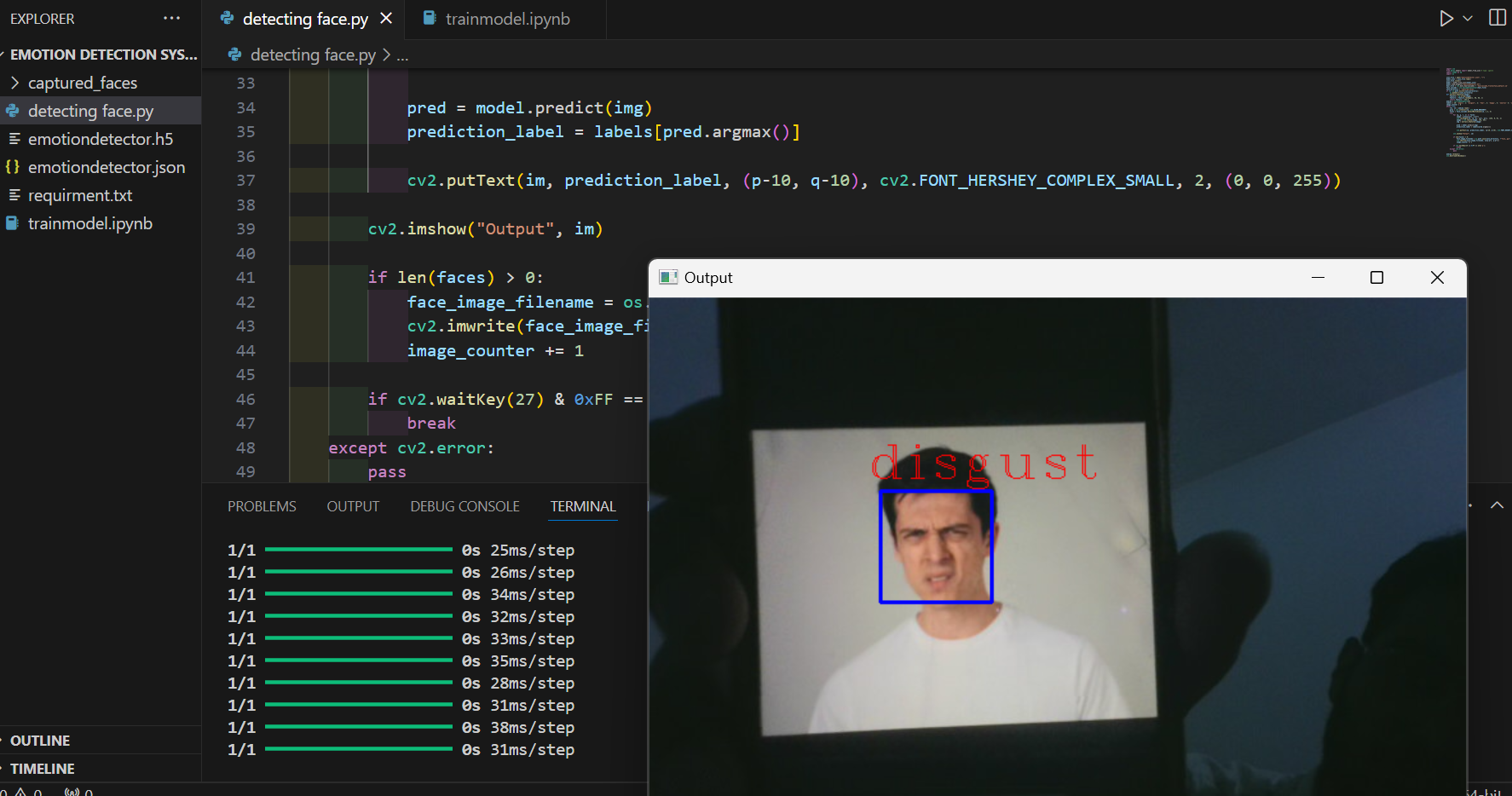
## **Explanation**:

1. **Webcam Initialization**:
   * cv2.VideoCapture(0) initializes the webcam. The program continuously captures video frames.
2. **Face Detection**:
   * face\_cascade.detectMultiScale detects faces in each frame. Detected faces are drawn with rectangles.
3. **Emotion Prediction**:
   * Detected faces are resized to 48x48 and passed through the extract\_features function.
   * The model predicts the emotion, and the result is displayed on the video feed with a label.
4. **Saving Faces**:
   * Captured face images are saved to the captured\_faces directory.
5. **Exiting the Program**:
   * The program runs indefinitely until the q key is pressed, releasing the webcam and closing windows.

# Chapter 4

# Result

OUTPUT:



Figure

# Conclusion

The Emotion Detection System represents a convergence of advanced technologies to create a solution that can identify and interpret human emotions in real time. This project aligns with the growing need for emotionally intelligent AI systems capable of understanding and responding to human behavior.

## ****Why I Chose Emotion Detection****

1. **Relevance**: Emotions play a vital role in human interaction. Automating their recognition can revolutionize areas like healthcare, where understanding patient emotions can enhance diagnosis and treatment.
2. **Personal Growth**: This project allowed me to explore and master the integration of AI with computer vision, strengthening my expertise in building real-world applications.
3. **Future Prospects**: Emotion detection is a stepping stone toward developing empathetic AI systems that adapt to user needs, making technology more intuitive and user-friendly.

## ****Future Applications****

1. **Healthcare**:
   * Assisting mental health professionals in tracking emotional changes over time.
2. **Education**:
   * Personalizing virtual learning environments based on students’ emotional states.
3. **Customer Service**:
   * Enhancing chatbot interactions by detecting user frustration or satisfaction.
4. **Entertainment**:
   * Adapting content recommendations based on viewers’ real-time emotional reactions.

This project highlights the potential of AI to bridge the gap between humans and technology, fostering interactions that are not only functional but also empathetic. By advancing this system, I aim to contribute meaningfully to the field of AI, making it more inclusive and impactful.