**Group member names**

**Syeda Anosha Mohsin. 2k20/ite/22**

**Mekhar Mumtaz 2k20/ite/68**

**Eisha Liaquat. 2k20/ite/34**

**Datasets**

Import numpy as np

Import pandas as pd

From PIL import Image

From tqdm import tqdm

Import os

# convert string to integer

Def atoi(s):

N = 0

For i in s:

N = n\*10 + ord(i) – ord(“0”)

Return n

# making folders

Outer\_names = [‘test’,’train’]

Inner\_names = [‘angry’, ‘disgusted’, ‘fearful’, ‘happy’, ‘sad’, ‘surprised’, ‘neutral’]

Os.makedirs(‘data’, exist\_ok=True)

For outer\_name in outer\_names:

Os.makedirs(os.path.join(‘data’,outer\_name), exist\_ok=True)

For inner\_name in inner\_names:

Os.makedirs(os.path.join(‘data’,outer\_name,inner\_name), exist\_ok=True)

# to keep count of each category

Angry = 0

Disgusted = 0

Fearful = 0

Happy = 0

Sad = 0

Surprised = 0

Neutral = 0

Angry\_test = 0

Disgusted\_test = 0

Fearful\_test = 0

Happy\_test = 0

Sad\_test = 0

Surprised\_test = 0

Neutral\_test = 0

Df = pd.read\_csv(‘./fer2013.csv’)

Mat = np.zeros((48,48),dtype=np.uint8)

Print(“Saving images...”)

# read the csv file line by line

For i in tqdm(range(len(df))):

Txt = df[‘pixels’][i]

Words = txt.split()

# the image size is 48x48

For j in range(2304):

Xind = j // 48

Yind = j % 48

Mat[xind][yind] = atoi(words[j])

Img = Image.fromarray(mat)

# train

If i < 28709:

If df[‘emotion’][i] == 0:

Img.save(‘train/angry/im’+str(angry)+’.png’)

Angry += 1

Elif df[‘emotion’][i] == 1:

Img.save(‘train/disgusted/im’+str(disgusted)+’.png’)

Disgusted += 1

Elif df[‘emotion’][i] == 2:

Img.save(‘train/fearful/im’+str(fearful)+’.png’)

Fearful += 1

Elif df[‘emotion’][i] == 3:

Img.save(‘train/happy/im’+str(happy)+’.png’)

Happy += 1

Elif df[‘emotion’][i] == 4:

Img.save(‘train/sad/im’+str(sad)+’.png’)

Sad += 1

Elif df[‘emotion’][i] == 5:

Img.save(‘train/surprised/im’+str(surprised)+’.png’)

Surprised += 1

Elif df[‘emotion’][i] == 6:

Img.save(‘train/neutral/im’+str(neutral)+’.png’)

Neutral += 1

# test

Else:

If df[‘emotion’][i] == 0:

Img.save(‘test/angry/im’+str(angry\_test)+’.png’)

Angry\_test += 1

Elif df[‘emotion’][i] == 1:

Img.save(‘test/disgusted/im’+str(disgusted\_test)+’.png’)

Disgusted\_test += 1

Elif df[‘emotion’][i] == 2:

Img.save(‘test/fearful/im’+str(fearful\_test)+’.png’)

Fearful\_test += 1

Elif df[‘emotion’][i] == 3:

Img.save(‘test/happy/im’+str(happy\_test)+’.png’)

Happy\_test += 1

Elif df[‘emotion’][i] == 4:

Img.save(‘test/sad/im’+str(sad\_test)+’.png’)

Sad\_test += 1

Elif df[‘emotion’][i] == 5:

Img.save(‘test/surprised/im’+str(surprised\_test)+’.png’)

Surprised\_test += 1

Elif df[‘emotion’][i] == 6:

Img.save(‘test/neutral/im’+str(neutral\_test)+’.png’)

Neutral\_test += 1

Print(“Done!”)

**Emotionas**.

Import numpy as np

Import argparse

Import matplotlib.pyplot as plt

Import cv2

From tensorflow.keras.models import Sequential

From tensorflow.keras.layers import Dense, Dropout, Flatten

From tensorflow.keras.layers import Conv2D

From tensorflow.keras.optimizers import Adam

From tensorflow.keras.layers import MaxPooling2D

From tensorflow.keras.preprocessing.image import ImageDataGenerator

Import os

Os.environ[‘TF\_CPP\_MIN\_LOG\_LEVEL’] = ‘2’

# command line argument

Ap = argparse.ArgumentParser()

Ap.add\_argument(“—mode”,help=”train/display”)

Mode = ap.parse\_args().mode

# plots accuracy and loss curves

Def plot\_model\_history(model\_history):

“””

Plot Accuracy and Loss curves given the model\_history

“””

Fig, axs = plt.subplots(1,2,figsize=(15,5))

# summarize history for accuracy

Axs[0].plot(range(1,len(model\_history.history[‘accuracy’])+1),model\_history.history[‘accuracy’])

Axs[0].plot(range(1,len(model\_history.history[‘val\_accuracy’])+1),model\_history.history[‘val\_accuracy’])

Axs[0].set\_title(‘Model Accuracy’)

Axs[0].set\_ylabel(‘Accuracy’)

Axs[0].set\_xlabel(‘Epoch’)

Axs[0].set\_xticks(np.arange(1,len(model\_history.history[‘accuracy’])+1),len(model\_history.history[‘accuracy’])/10)

Axs[0].legend([‘train’, ‘val’], loc=’best’)

# summarize history for loss

Axs[1].plot(range(1,len(model\_history.history[‘loss’])+1),model\_history.history[‘loss’])

Axs[1].plot(range(1,len(model\_history.history[‘val\_loss’])+1),model\_history.history[‘val\_loss’])

Axs[1].set\_title(‘Model Loss’)

Axs[1].set\_ylabel(‘Loss’)

Axs[1].set\_xlabel(‘Epoch’)

Axs[1].set\_xticks(np.arange(1,len(model\_history.history[‘loss’])+1),len(model\_history.history[‘loss’])/10)

Axs[1].legend([‘train’, ‘val’], loc=’best’)

Fig.savefig(‘plot.png’)

Plt.show()

# Define data generators

Train\_dir = ‘data/train’

Val\_dir = ‘data/test’

Num\_train = 28709

Num\_val = 7178

Batch\_size = 64

Num\_epoch = 50

Train\_datagen = ImageDataGenerator(rescale=1./255)

Val\_datagen = ImageDataGenerator(rescale=1./255)

Train\_generator = train\_datagen.flow\_from\_directory(

Train\_dir,

Target\_size=(48,48),

Batch\_size=batch\_size,

Color\_mode=”grayscale”,

Class\_mode=’categorical’)

Validation\_generator = val\_datagen.flow\_from\_directory(

Val\_dir,

Target\_size=(48,48),

Batch\_size=batch\_size,

Color\_mode=”grayscale”,

Class\_mode=’categorical’)

# Create the model

Model = Sequential()

Model.add(Conv2D(32, kernel\_size=(3, 3), activation=’relu’, input\_shape=(48,48,1)))

Model.add(Conv2D(64, kernel\_size=(3, 3), activation=’relu’))

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

Model.add(Dropout(0.25))

Model.add(Conv2D(128, kernel\_size=(3, 3), activation=’relu’))

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

Model.add(Conv2D(128, kernel\_size=(3, 3), activation=’relu’))

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

Model.add(Dropout(0.25))

Model.add(Flatten())

Model.add(Dense(1024, activation=’relu’))

Model.add(Dropout(0.5))

Model.add(Dense(7, activation=’softmax’))

# If you want to train the same model or try other models, go for this

If mode == “train”:

Model.compile(loss=’categorical\_crossentropy’,optimizer=Adam(lr=0.0001, decay=1e-6),metrics=[‘accuracy’])

Model\_info = model.fit\_generator(

Train\_generator,

Steps\_per\_epoch=num\_train // batch\_size,

Epochs=num\_epoch,

Validation\_data=validation\_generator,

Validation\_steps=num\_val // batch\_size)

Plot\_model\_history(model\_info)

Model.save\_weights(‘model.h5’)

# emotions will be displayed on your face from the webcam feed

Elif mode == “display”:

Model.load\_weights(‘model.h5’)

# prevents openCL usage and unnecessary logging messages

Cv2.ocl.setUseOpenCL(False)

# dictionary which assigns each label an emotion (alphabetical order)

Emotion\_dict = {0: “Angry”, 1: “Disgusted”, 2: “Fearful”, 3: “Happy”, 4: “Neutral”, 5: “Sad”, 6: “Surprised”}

# start the webcam feed

Cap = cv2.VideoCapture(0)

While True:

# Find haar cascade to draw bounding box around face

Ret, frame = cap.read()

If not ret:

Break

Facecasc = cv2.CascadeClassifier(‘haarcascade\_frontalface\_default.xml’)

Gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

Faces = facecasc.detectMultiScale(gray,scaleFactor=1.3, minNeighbors=5)

For (x, y, w, h) in faces:

Cv2.rectangle(frame, (x, y-50), (x+w, y+h+10), (255, 0, 0), 2)

Roi\_gray = gray[y:y + h, x:x + w]

Cropped\_img = np.expand\_dims(np.expand\_dims(cv2.resize(roi\_gray, (48, 48)), -1), 0)

Prediction = model.predict(cropped\_img)

Maxindex = int(np.argmax(prediction))

Cv2.putText(frame, emotion\_dict[maxindex], (x+20, y-60), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE\_AA)

Cv2.imshow(‘Video’, cv2.resize(frame,(1600,960),interpolation = cv2.INTER\_CUBIC))

If cv2.waitKey(1) & 0xFF == ord(‘q’):

Break

Cap.release()

Cv2.destroyAllWindows()