EMOTION DETECTION

DIVIJA L	PES1UG20CS134
GAURAV MAHAJAN	PES1UG20CS150

SUMMARY:

Emotions are fundamental to human behavior and play a crucial role in decision-making, relationships, and overall well-being. By accurately detecting and analyzing emotions, we can gain insights into human behavior, motivations, and attitudes. This information can be valuable for various fields such as psychology, marketing, customer service, and healthcare.This project provides a complete pipeline for emotion detection. It starts with preparing the dataset and organizing the images, followed by training a CNN model on the dataset. Finally, the trained model is deployed to perform real-time emotion detection from a webcam feed, enabling the system to recognize and display emotions in human faces.The combination of the dataset preparation code and the emotion detection code allows for the training and deployment of an emotion detection model capable of recognizing facial expressions in real-time applications.

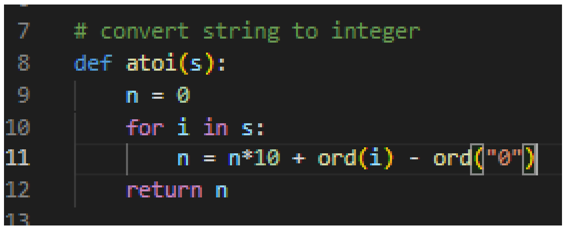
DATASET:

The FER2013 dataset is a widely used dataset in the field of facial expression recognition. It consists of 48x48-pixel grayscale images representing facial expressions categorized into seven emotion classes.

- Number of Images: The dataset contains a total of 35,887 images.
- Emotion Classes: The images are categorized into the following seven emotion classes:
  0: Angry
  1: Disgusted
  2: Fearful
  3: Happy
  4: Sad
  5: Surprised
  6: Neutral
- Data Format: Each image in the dataset is represented by a string of pixel values. The pixel values are space-separated integers ranging from 0 to 255, representing the grayscale intensity of each pixel in the image. The dataset also includes the emotion labels for each image.

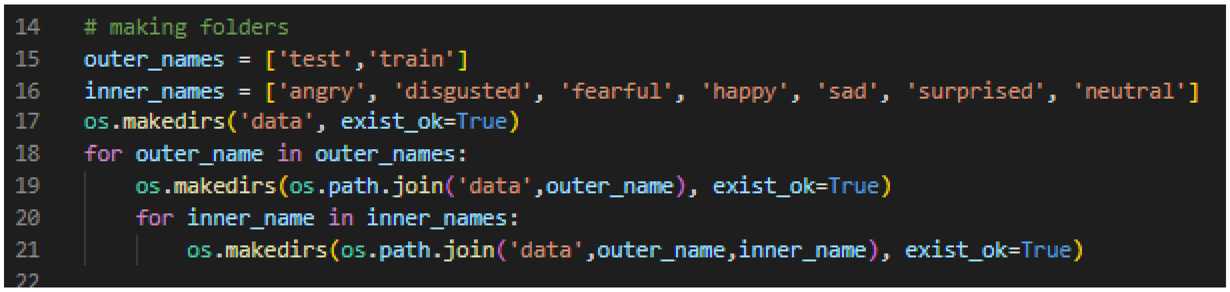
TASK:
1.DATASET PREPARATION:

- The code begins by defining a function, atoi, which converts a string of digits to an integer. It is used later to convert pixel values from the dataset into integer format.



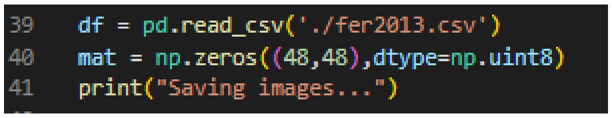
```python
# convert string to integer
def atoi(s):
    n = 0
    for i in s:
        n = n*10 + ord(i) - ord("0")
    return n
```

- Next it creates the necessary folder structure for storing the image data. It creates an outer folder for both the train and test sets and inner folders for each emotion category within them.



```python
# 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)
```

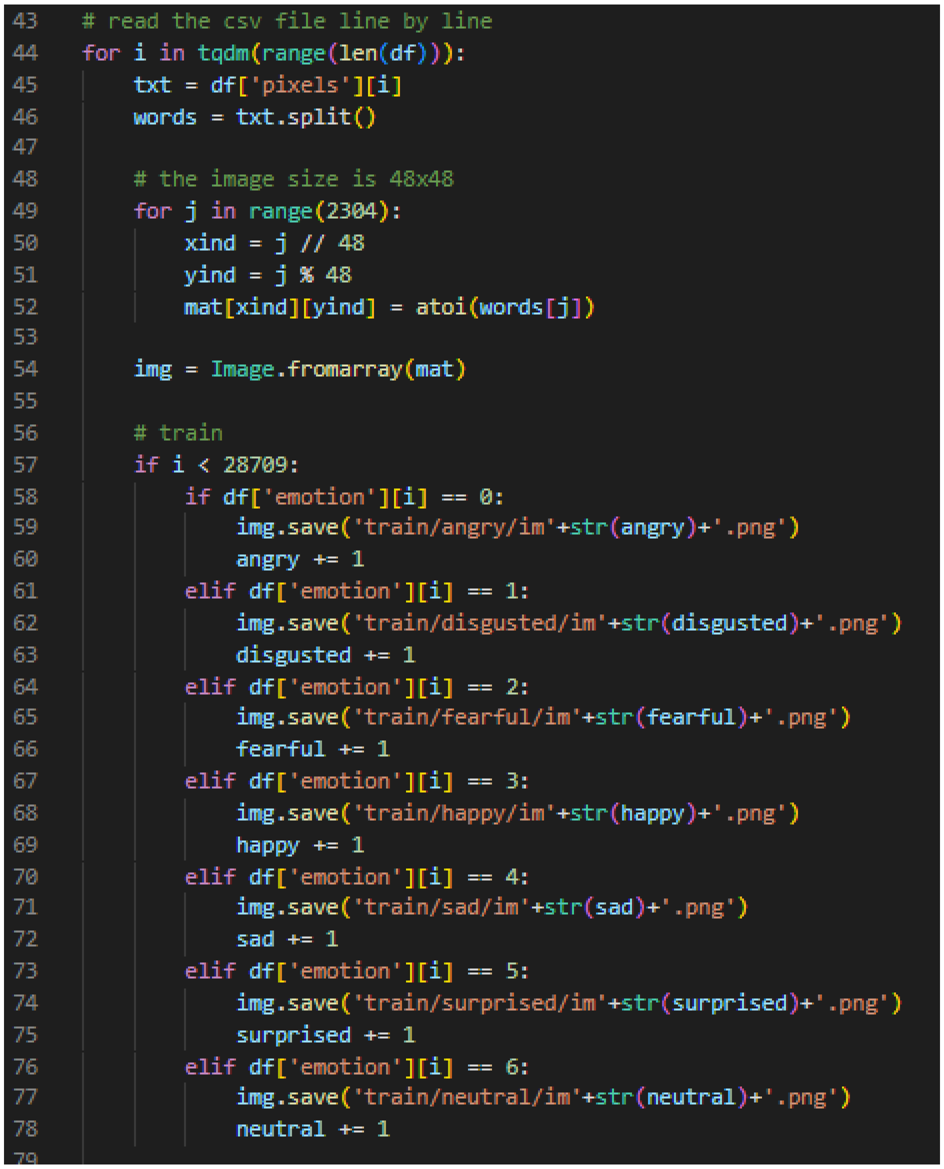
- reads the CSV file using pd.read_csv() and initializes an empty matrix mat of size 48x48, representing the image dimensions.



```python
df = pd.read_csv('./fer2013.csv')
mat = np.zeros((48,48),dtype=np.uint8)
print("Saving images...")
```

- A loop iterates over each row in the dataset. For each row, the code extracts the pixel values for the image and stores them in the mat matrix. The matrix is then converted into an image using Image.fromarray().

- Based on the index of the row, the code determines whether to save the image in the train or test folder. It checks the corresponding emotion label and increments the respective count for that emotion. The image is saved with a unique name based on the emotion and count.



```python
# 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
```

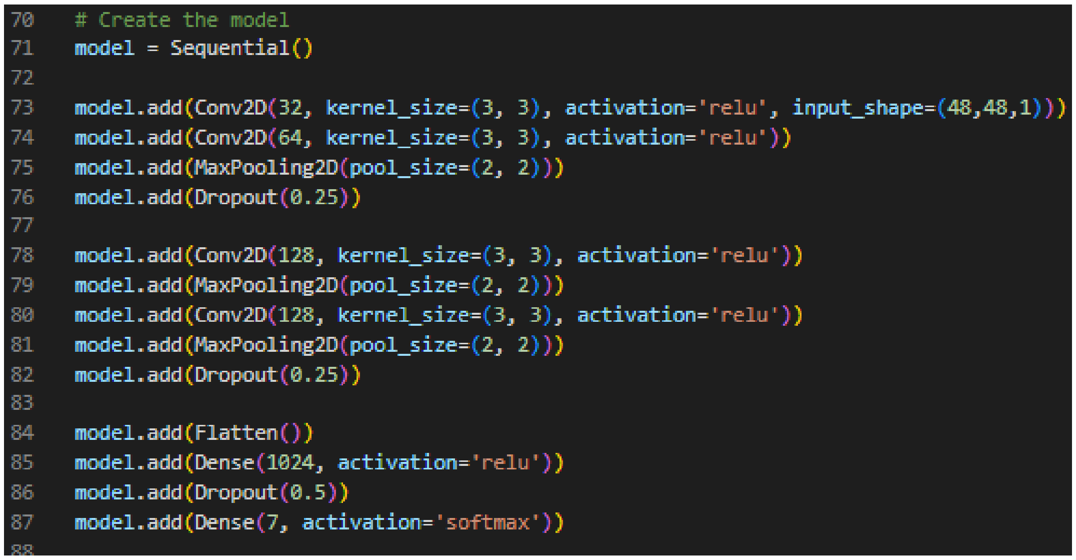
- This code reads the FER2013 dataset, extracts the images and their corresponding emotions, and organizes them into train and test sets based on their emotion labels. It saves the images in separate folders for each emotion category, facilitating further analysis or training of emotion detection models.

2.MODEL TRAINING:

We have used a Convolutional Neural Network (CNN) model to classify facial expressions into seven emotions: angry, disgusted, fearful, happy, neutral, sad, and surprised. The code allows two phases: training and displaying.

(i)TRAINING PHASE:

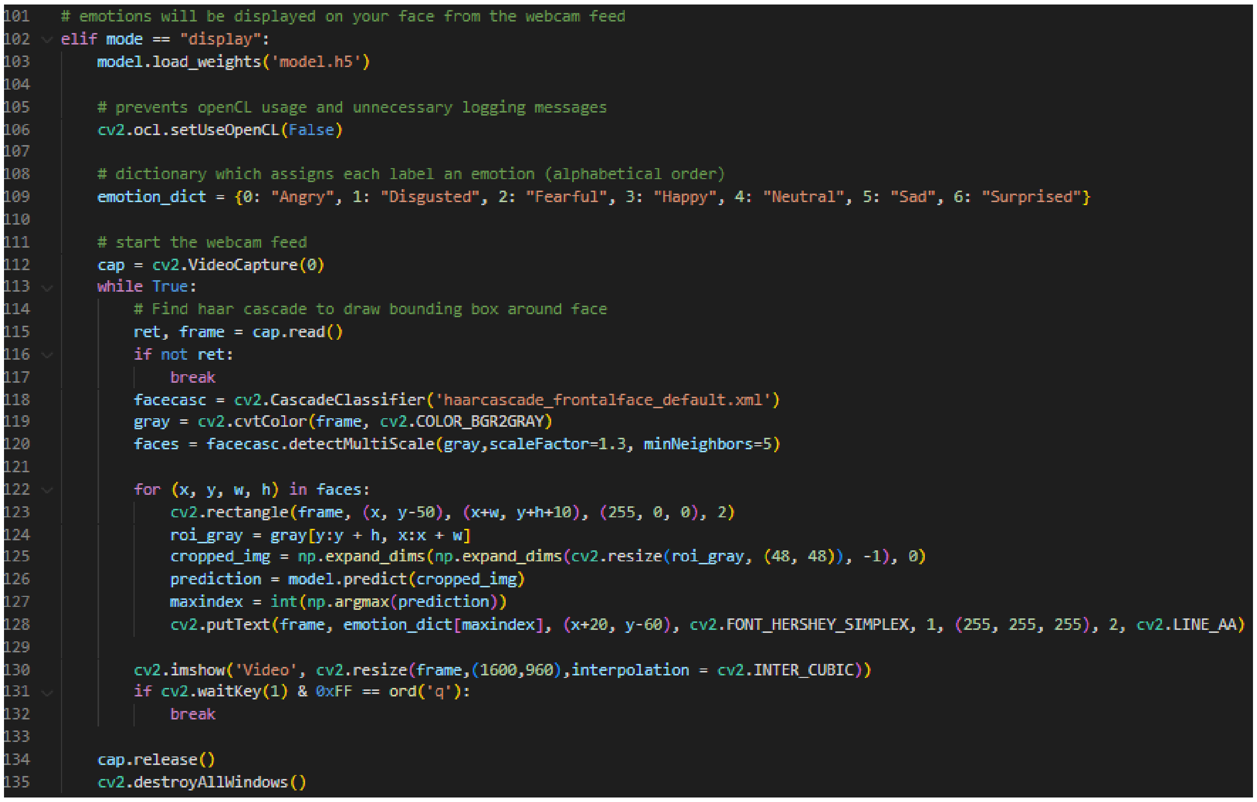
The code creates data generators to preprocess the images and feed them into the model. It then builds a CNN model with several convolutional, pooling, dropout, and dense layers. The model is compiled with appropriate loss function, optimizer, and metrics.It compiles the model with appropriate loss function, optimizer, and metrics. Data generators are created using the ImageDataGenerator class to preprocess the training and validation images. The model is then trained using the fit_generator() function, and the training history is plotted.The model's training history is plotted, and the trained weights are saved for future use.



```
# 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'))
```

(ii)DISPLAYING PHASE:

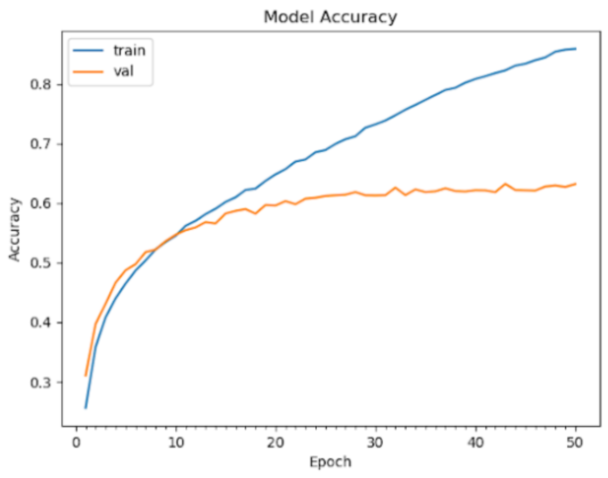
The pre-trained model's weights are loaded. The code accesses the webcam feed and utilizes the Haar cascade classifier to detect faces within the video frames. For each detected face, the corresponding region of interest is extracted, preprocessed, and fed into the model for emotion prediction. The predicted emotion label is overlaid on the frame. This process continues until the user decides to exit.

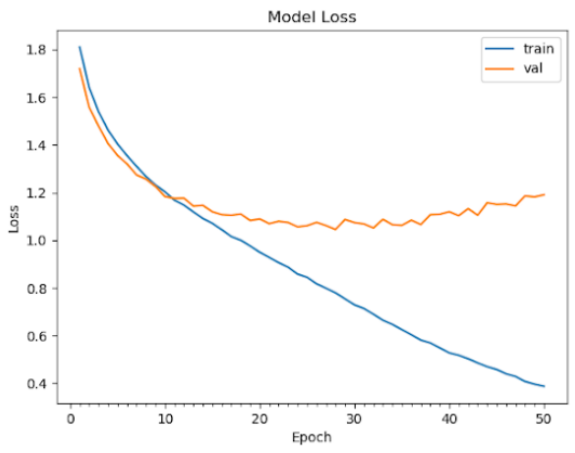


```python
# emotions will be displayed on your face from the webcam feed
elif mode == "display":
    model.load_weights('model.h5')

    # 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()
```

Overall,it provides a complete pipeline for training an emotion detection model on the FER2013 dataset and using the trained model for real-time emotion detection from a webcam feed.

ACCURACY:


Model Accuracy
train
val
Accuracy
Epoch


Model Loss
train
val
Loss
Epoch