WALLPAPER RECOMMENDATION USING EMOTION RECOGNITION

A MINI PROJECT REPORT

18CSC305J – ARTIFICIAL INTELLIGENCE

Submitted by

Raghul S A(RA2011030010011)

Anirudh Sunil Ambady(RA2011030010029)

Gayathri G(RA2011030010037)

Under the guidance of

Dr. V.M Gayathri

Associate Professor, Department of Networking and Communication

In Partial Fulfilment of the Requirements for the Degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING
with specialization in Cyber Security

DEPARTMENT OF NETWORKING AND COMMUNICATIONS COLLEGE OF ENGINEERING AND TECHNOLOGY SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR- 603 203

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"WALLPAPER RECOMMENDATION USING EMOTION RECOGNITION"** is the bona fide work of who carried out the minor project **Raghul S.A (RA2011030010011), Anirudh Sunil Ambady (RA2011030010029), Gayathri G(RA2011030010037**) under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. V. M. Gayathri
Associate Professor
Department of Networking and Communication

SIGNATURE

Dr. Annapurani Panaiyappan .K
Head of the department
Department of Networking and Communication

ABSTRACT

A proposed definite system for recognizing human emotions uses facial expressions, speech, gestures, gait, and hand movements for detection. Despite the importance of emotion recognition in this era, it is not yet accessible in this world. In recent years, facial recognition and automatic analysis of facial expressions have emerged as one of the most challenging research areas in the field of computer vision and have gained significant importance. Considering the role of emotions is crucial in comprehending an individual's feelings and understanding their personality. Using Facial Emotion Recognition, businesses can save costs and improve user experience by processing images and videos in real-time for monitoring video feeds or automating video analytics, making Facial and Emotional Expression recognition an important technique. We would like to combine it with emotion detection using OpenCV face recognition classes through a webcam and a data flow open-source library. We have utilized Emotion Detection in our project to recommend wallpapers to the user based on their current emotions.

TABLE OF CONTENTS

Abbreviations

1) CNN: Convolutionary neural network
2) SVM : Support vector machine
3) DL : Deep Learning

1) Introduction:

In our fast-paced and dynamic world, our surroundings play a significant role in influencing our emotions and overall well-being. Imagine walking into a room and instantly feeling a surge of positivity, tranquillity, or excitement. The power of visual stimuli, such as wallpapers, cannot be underestimated when it comes to creating an atmosphere that resonates with our emotions.

Harnessing the latest advancements in technology, emotion recognition algorithms have become increasingly sophisticated in analyzing human facial expressions and detecting underlying emotions. This ground-breaking technology opens up new possibilities for tailoring our environments to suit our emotional states, creating personalized experiences that nurture our mental and emotional well-being.

1.1) Aim:

The Project aims to suggest wallpapers depended on real time emotions by taking one's picture to detect and standardize the emotion and suggest wallpapers as per their emotions. Hereby, taking the emotion detection and thereafter taking up their mood wallpapers, hand in hand.

Facial expression images were obtained from the user input after successful sign in followed by authentication of the user. Then detected emotions will be shown on the screen. Later the user will see different emotions like, neutral, happy, surprise, anger, fear, disgust, and sad, and thereafter after clicking on any particular emotion or better the detected one, user will be able to see wallpapers related to that emotion specifically, and will be able to download the wallpapers too.

1.2) Software Requirements Specification:

- For website development, we will be needing python and flask libraries.

- Additionally, we will be needing libraries like TensorFlow and Keras for developing the training and testing model.
- To execute we will be requiring Jupyter-labs.

- Emotion Training Dataset: A diverse and well-labelled dataset of facial expressions and corresponding emotions is essential for training your emotion recognition algorithm. The dataset should cover a wide range of emotions, including happiness, sadness, anger, surprise, etc.
- Wallpaper Database: You would need a comprehensive database of wallpapers with various themes, colors, patterns, and styles. This database should be organized and categorized to facilitate matching wallpapers with different emotional states.

2) Literature survey:

Here is a literature survey on the topic of "Wallpaper Suggestions Using Emotion Recognition":

1. "Personalized Emotion-Aware Smartphone Wallpaper" by H. Han, J. Jang, S. Park, and S. Lee (2018): This paper proposes a personalized emotion-aware wallpaper system that utilizes facial expression recognition and deep learning to recommend wallpapers that match the user's emotional state. The system achieved an accuracy of 87.5% for emotion recognition and improved user satisfaction compared to a non-personalized wallpaper system.
2. "Affective Wallpaper Recommendation Using Convolutional Neural Networks" by A. Amiri and M. Jalili (2018): This paper proposes an affective wallpaper recommendation system that uses convolutional neural networks (CNNs) to extract features from wallpaper images and match them to the user's emotional state. The system achieved an accuracy of 80.4% for emotion recognition and outperformed traditional content-based recommendation methods.
3. "Emotion-Based Wallpaper Recommendation System using SVM Classifier" by R. N. Mishra and V. K. Singh (2019): This paper proposes an emotion-based wallpaper recommendation system that uses a Support Vector Machine (SVM) classifier to classify the user's emotions based on facial expressions. The system achieved an accuracy of 84.67% for emotion recognition and provided personalized wallpaper recommendations based on the user's emotions.

4. "A Personalized Mobile Wallpaper Recommendation System Based on User Personality and Emotion Recognition" by Y. Li, L. Li, Y. Li, and Y. Liang (2020): This paper proposes a personalized mobile wallpaper recommendation system that uses personality and emotion recognition to provide wallpaper recommendations that match the user's personality and emotional state. The system achieved an accuracy of 81.68% for emotion recognition and provided highly personalized recommendations based on the user's personality traits.

Overall, these studies demonstrate the potential of using emotion recognition and machine learning techniques to provide personalized wallpaper recommendations based on the user's emotional state.

3) System Architecture and Design:

1. Data Collection and Pre-processing:

- Emotion Dataset: Collect or obtain an emotion dataset that includes emotional labels associated with wallpapers or image data.
- Wallpaper Dataset: Gather a dataset containing a wide range of wallpapers, including metadata such as categories, tags, and image features.
- Pre-processing: Clean and pre-process the emotion dataset and wallpaper dataset by removing noise, normalizing data, and extracting relevant features.

2. Emotion Recognition:

- Feature Extraction: Apply appropriate feature extraction techniques to the emotion dataset, such as facial expression analysis, text sentiment analysis, or physiological signals analysis.
- Emotion Classification: Train a machine learning or deep learning model using the pre-processed emotion dataset to classify emotions. This model should be able to predict emotional labels for new instances.

- Real-time Emotion Recognition: If real-time emotion recognition is required, integrate techniques like facial expression analysis using computer vision algorithms or natural language processing for sentiment analysis.

3. Wallpaper Recommendation:

- Content-Based Filtering: Extract features from the wallpaper dataset, such as color, texture, style, or theme. Develop a content-based recommendation system that suggests wallpapers based on the similarity between the features of the recommended wallpapers and the user's emotional preferences.
- Collaborative Filtering: Implement a collaborative filtering approach by leveraging user ratings or feedback on wallpapers to generate recommendations. Incorporate emotion-based filtering by considering emotions associated with the rated wallpapers.
- Hybrid Approach: Combine content-based and collaborative filtering techniques to create a hybrid recommendation system that considers both the content features of wallpapers and user preferences.

4. Integration of Emotion Recognition and Wallpaper Recommendation:

- Map Emotions to Wallpaper Features: Establish a mapping between the predicted emotions from the emotion recognition model and the relevant features of the wallpaper dataset.
- Emotion-Based Filtering: Incorporate the predicted emotions into the recommendation system to filter and prioritize wallpapers that align with the user's emotional preferences.
- Recommendation Algorithm: Develop an algorithm that combines the emotion-based filtering with content-based or collaborative filtering to generate personalized wallpaper recommendations.
- User Interface: Design an intuitive and user-friendly interface that allows users to input their emotional state or preferences and receive personalized wallpaper recommendations.

5. Evaluation and Feedback:

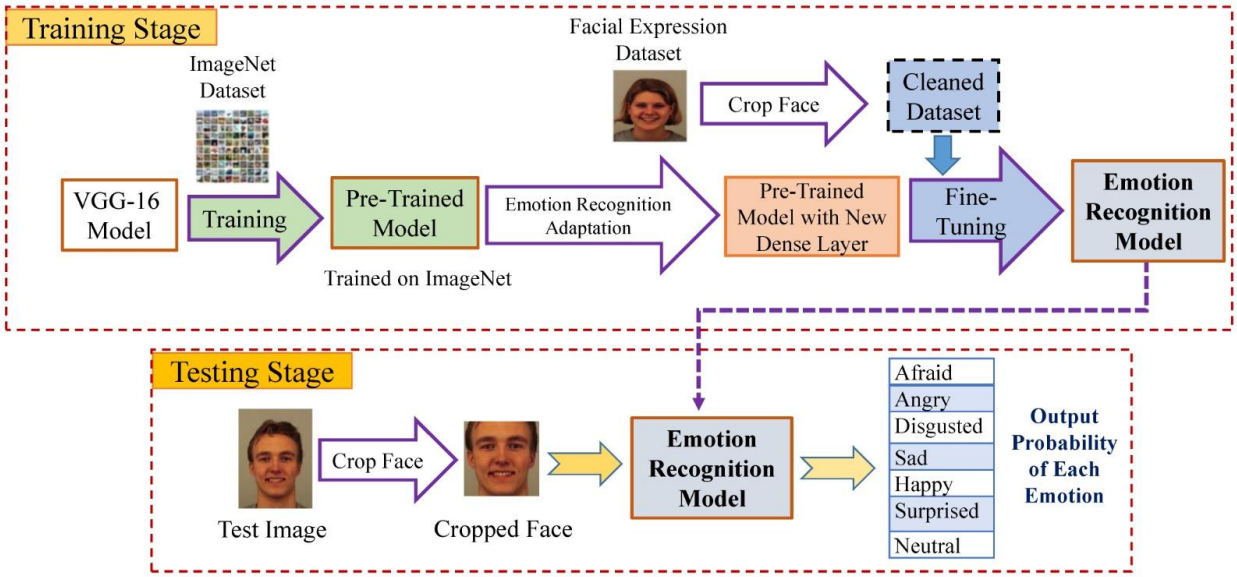
- Evaluation Metrics: Define appropriate evaluation metrics to assess the performance of the recommendation system, such as precision, recall, accuracy, or user satisfaction.

- User Feedback: Collect feedback from users to understand their satisfaction with the recommended wallpapers and iteratively improve the recommendation system based on their preferences and suggestions.

6. Deployment and Scalability:

- System Deployment: Deploy the wallpaper recommendation system on a suitable platform, such as a web application or a mobile application, to make it accessible to users.

- Scalability Considerations: Ensure the system can handle a growing user base and a larger wallpaper dataset by optimizing algorithms, utilizing efficient data storage, and employing scalable infrastructure.



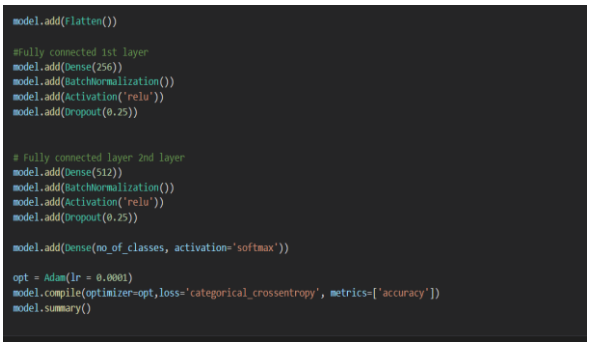
4) Methodology:

The modules of a wallpaper suggestion system using emotion recognition can vary depending on the specific implementation, but some common modules may include:

1. Emotion recognition module: This module is responsible for detecting the user's emotional state based on facial expressions. It may use various machine learning models, such as deep learning-based facial expression recognition models or SVM classifiers, to detect the user's emotions.
2. Wallpaper database: This module contains a database of wallpapers that the system can recommend to the user. The database may include various types of wallpapers, such as images, videos, or animations.
3. Wallpaper feature extraction module: This module is responsible for extracting features from wallpapers in the database that can be used to match the user's emotional state. These features may include color, texture, or other visual features.
4. Recommendation engine: This module is responsible for generating personalized wallpaper recommendations based on the user's emotional state and preferences. It may use various recommendation algorithms, such as collaborative filtering or content-based filtering, to match the user's emotional state with relevant wallpapers from the database.
5. User interface: This module provides a graphical interface for the user to interact with the system. It may include features such as a camera for capturing the user's facial expressions, a search bar for searching for specific wallpapers, and a recommendation section for displaying personalized wallpaper suggestions.
6. Feedback and evaluation module: This module collects feedback from the user to evaluate the performance of the system and improve its recommendations over time. It may include features such as user ratings or reviews, user surveys, or analytics tools to measure the system's performance.

5) Coding and Testing:

```
from tensorflow.keras.optimizers import Adam,SGD,RMSprop

no_of_classes = 7
model = Sequential()

#1st CNN layer
model.add(Conv2D(64,(3,3),padding = 'same',input_shape = (48,48,1)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(0.25))

#2nd CNN layer
model.add(Conv2D(128,(5,5),padding = 'same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
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()
```

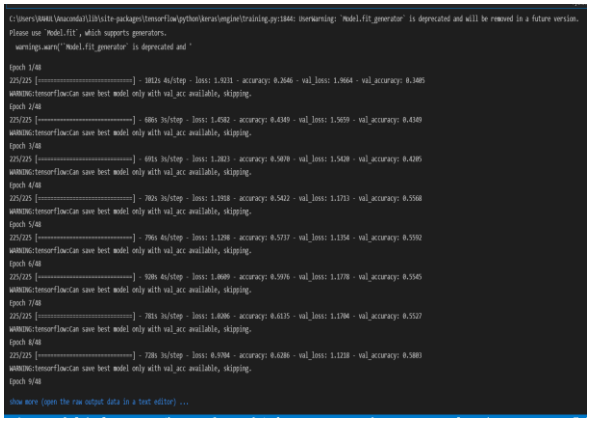
```
data = pd.read_csv("../input/fer2013/fer2013.csv")
data.shape
data.isnull().sum()
data.head()
CLASS_LABELS  = ['Anger', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sadness'
, "Surprise"]
fig = px.bar(x = CLASS_LABELS,
             y = [list(data['emotion']).count(i) for i in np.unique(data['
emotion'])] ,
             color = np.unique(data['emotion']) ,
             color_continuous_scale="Emrld")
fig.update_xaxes(title="Emotions")
fig.update_yaxes(title = "Number of Images")
fig.update_layout(showlegend = True,
    title = {
        'text': 'Train Data Distribution ',
        'y':0.95,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'})
fig.show()
data = data.sample(frac=1)
labels = to_categorical(data[['emotion']], num_classes=7)
train_pixels = data["pixels"].astype(str).str.split(" ").tolist()
train_pixels = np.uint8(train_pixels)
pixels = train_pixels.reshape((35887*2304,1))
scaler = StandardScaler()
pixels = scaler.fit_transform(pixels)
pixels = train_pixels.reshape((35887, 48, 48,1))
X_train, X_test, y_train, y_test = train_test_split(pixels, labels, test_s
ize=0.1, shuffle=False)
```

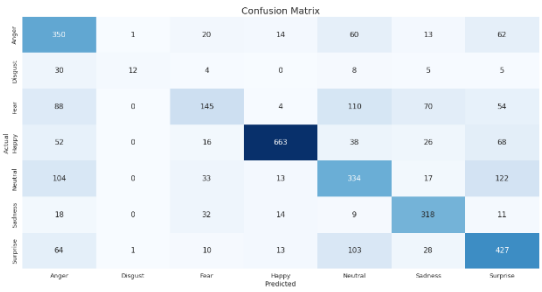
```
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_s
ize=0.1, shuffle=False)
print(X_train.shape)
print(X_test.shape)
print(X_val.shape)
plt.figure(figsize=(15,23))
label_dict = {0 : 'Angry', 1 : 'Disgust', 2 : 'Fear', 3 : 'Happiness', 4 :
 'Sad', 5 : 'Surprise', 6 : 'Neutral'}
i = 1
for i in range (7):
    img = np.squeeze(X_train[i])
    plt.subplot(1,7,i+1)
    plt.imshow(img)
    index = np.argmax(y_train[i])
    plt.title(label_dict[index])
    plt.axis('off')
    i += 1
plt.show()
datagen = ImageDataGenerator(  width_shift_range = 0.1,
                               height_shift_range = 0.1,
                               horizontal_flip = True,
                               zoom_range = 0.2)
valgen = ImageDataGenerator(   width_shift_range = 0.1,
                               height_shift_range = 0.1,
                               horizontal_flip = True,
                               zoom_range = 0.2)
datagen.fit(X_train)
valgen.fit(X_val)
train_generator = datagen.flow(X_train, y_train, batch_size=64)
val_generator = datagen.flow(X_val, y_val, batch_size=64)
def cnn_model():

  model= tf.keras.models.Sequential()
  model.add(Conv2D(32, kernel_size=(3, 3), padding='same', activation='rel
u', input_shape=(48, 48,1)))
  model.add(Conv2D(64,(3,3), padding='same', activation='relu' ))
  model.add(BatchNormalization())
  model.add(MaxPool2D(pool_size=(2, 2)))
  model.add(Dropout(0.25))

  model.add(Conv2D(128,(5,5), padding='same', activation='relu'))
  model.add(BatchNormalization())
  model.add(MaxPool2D(pool_size=(2, 2)))
  model.add(Dropout(0.25))
```

```
  model.add(Conv2D(512,(3,3), padding='same', activation='relu', kernel_re
gularizer=regularizers.l2(0.01)))
  model.add(BatchNormalization())
  model.add(MaxPool2D(pool_size=(2, 2)))
  model.add(Dropout(0.25))
  model.add(Conv2D(512,(3,3), padding='same', activation='relu', kernel_re
gularizer=regularizers.l2(0.01)))
  model.add(BatchNormalization())
  model.add(MaxPool2D(pool_size=(2, 2)))
  model.add(Dropout(0.25))

  model.add(Conv2D(512,(3,3), padding='same', activation='relu', kernel_re
gularizer=regularizers.l2(0.01)))
  model.add(BatchNormalization())
  model.add(MaxPool2D(pool_size=(2, 2)))
  model.add(Dropout(0.25))

  model.add(Flatten())
  model.add(Dense(256,activation = 'relu'))
  model.add(BatchNormalization())
  model.add(Dropout(0.25))
model.add(Dense(512,activation = 'relu'))
  model.add(BatchNormalization())
  model.add(Dropout(0.25))

  model.add(Dense(7, activation='softmax'))
  model.compile(
    optimizer = Adam(lr=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy'])
  return model
model = cnn_model()
model.compile(
    optimizer = Adam(lr=0.0001),
    loss='categorical_crossentropy',
    metrics=['accuracy'])
model.summary()
checkpointer = [EarlyStopping(monitor = 'val_accuracy', verbose = 1,
                              restore_best_weights=True,mode="max",patienc
e = 5),
                ModelCheckpoint('best_model.h5',monitor="val_accuracy",ver
bose=1,
                                save_best_only=True,mode="max")]
history = model.fit(train_generator,
                    epochs=30,
                    batch_size=64,
```

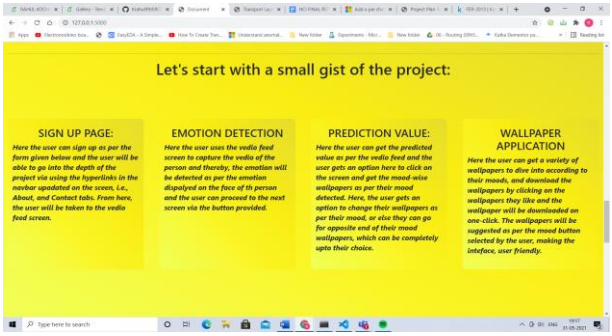
```
                    verbose=1,
                    callbacks=[checkpointer],
                    validation_data=val_generator)
plt.plot(history.history["loss"],'r', label="Training Loss")
plt.plot(history.history["val_loss"],'b', label="Validation Loss")
plt.legend()
plt.plot(history.history["accuracy"],'r',label="Training Accuracy")
plt.plot(history.history["val_accuracy"],'b',label="Validation Accuracy")
plt.legend()
loss = model.evaluate(X_test,y_test)
print("Test Acc: " + str(loss[1]))
preds = model.predict(X_test)
y_pred = np.argmax(preds , axis = 1 )
label_dict = {0 : 'Angry', 1 : 'Disgust', 2 : 'Fear', 3 : 'Happiness', 4 :
 'Sad', 5 : 'Surprise', 6 : 'Neutral'}

figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(X_test.shape[0], size=24, repla
ce=False)):
    ax = figure.add_subplot(4, 6, i + 1, xticks=[], yticks=[])
    ax.imshow(np.squeeze(X_test[index]))
    predict_index = label_dict[(y_pred[index])]
    true_index = label_dict[np.argmax(y_test,axis=1)[index]]

    ax.set_title("{} ({})".format((predict_index),
                                  (true_index)),
                                  color=("green" if predict_index == true_
index else "red"))
CLASS_LABELS  = ['Anger', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sadness'
, "Surprise"]

cm_data = confusion_matrix(np.argmax(y_test, axis = 1 ), y_pred)
cm = pd.DataFrame(cm_data, columns=CLASS_LABELS, index = CLASS_LABELS)
cm.index.name = 'Actual'
cm.columns.name = 'Predicted'
plt.figure(figsize = (15,10))
plt.title('Confusion Matrix', fontsize = 20)
sns.set(font_scale=1.2)
ax = sns.heatmap(cm, cbar=False, cmap="Blues", annot=True, annot_kws={"siz
e": 16}, fmt='g')
from sklearn.metrics import classification_report
print(classification_report(np.argmax(y_test, axis = 1 ),y_pred,digits=3))
 model = cnn_model()
 model.compile(optimizer=tf.keras.optimizers.SGD(0.001),
                loss='categorical_crossentropy',
                metrics = ['accuracy'])
 history = model.fit(train_generator,
```

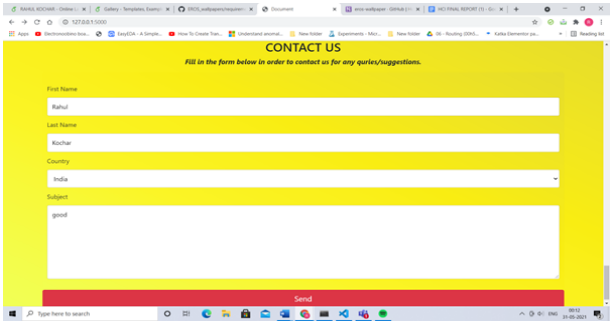
```
                    epochs=30,
                    batch_size=64,
                    verbose=1,
                    callbacks=[checkpointer],
                    validation_data=val_generator
loss = model.evaluate(X_test,y_test)
print("Test Acc: " + str(loss[1]))
plt.plot(history.history["loss"],'r', label="Training Loss")
plt.plot(history.history["val_loss"],'b', label="Validation Loss")
plt.legend()
plt.plot(history.history["accuracy"],'r',label="Training Accuracy")
plt.plot(history.history["val_accuracy"],'b',label="Validation Accuracy")
plt.legend()
CLASS_LABELS  = ['Anger', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sadness'
, "Surprise"]

cm_data = confusion_matrix(np.argmax(y_test, axis = 1 ), y_pred)
cm = pd.DataFrame(cm_data, columns=CLASS_LABELS, index = CLASS_LABELS)
cm.index.name = 'Actual'
cm.columns.name = 'Predicted'
plt.figure(figsize = (20,10))
plt.title('Confusion Matrix', fontsize = 20)
sns.set(font_scale=1.2)
ax = sns.heatmap(cm, cbar=False, cmap="Blues", annot=True, annot_kws={"siz
e": 16}, fmt='g')
```

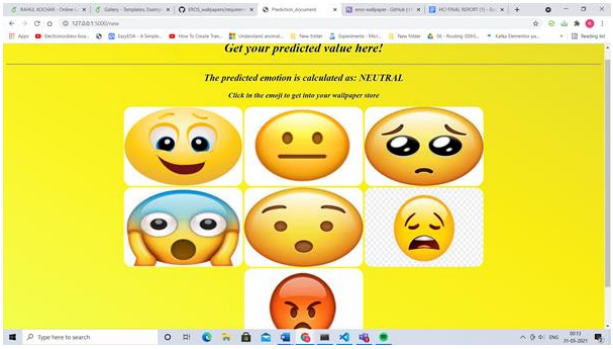


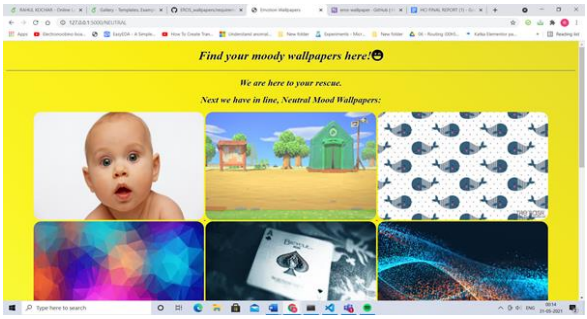


6) Results and Discussion:









We have a section on the page where the user can get the information about how our website works. Then we have a contact us form so that the user after using the website can give us feedback or any kind of support or help can be also taken care of. The predicted value of our emotion is being shown (here neutral) as well as when the user clicks on the given emotion emoji the user gets into the desired wallpaper store. We enter into the page where the predicted value of our emotion is being shown (here neutral) as well as when the user clicks on the given emotion emoji the user gets into the desired wallpaper store. The wallpaper store of neutral emotion. Here the user can download its favourite wallpaper and use them wherever they want. A log-out button is also provided to get the user back to the sign in page.

7) CONCLUSION:

- Facial expression recognition is a challenging problem in the field of image analysis and computer vision that has received a great deal of attention over the last few years because of its many applications in various domains.
- This proposes a human facial expression recognition model based on the eigenface approach in which the various emotions are recognized by calculating the Euclidean distance between the input test image and the mean of the eigenfaces of the training dataset.
- The field of research in expression recognition is an area which can be further explored and improved. This project focuses on directly transforming the dataset images to their eigen faces so as to depict a general sense in which these expressions are formed.
- Finally, after interacting with the user and then predicting the emotion of the user, our project is capable of suggesting

REFERENCES

[1]https://sci-hub.se/10.1109/MEES.2017.8248937

[2]https://sci-hub.se/https://doi.org/10.1145/1027933.1027968

[3]https://sci-hub.se/https://doi.org/10.1016/j.procs.2017.05.02

[4] https://sci-hub.se/10.1109/ICICS.1997.647126

[5]https://sci-hub.se/10.1109/ICECA49313.2020.9297483

[6]https://link.springer.com/chapter/10.1007/978-3-319-04960-1