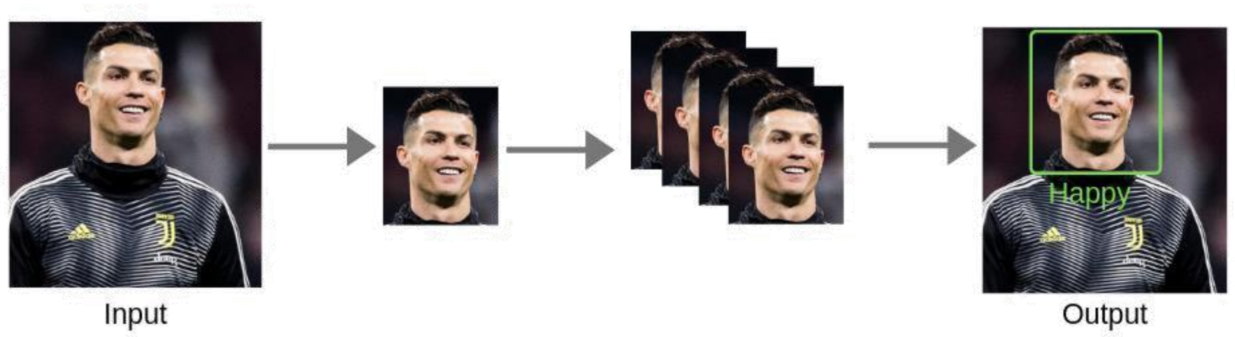


HUMAN EMOTION RECOGNITION

Contents

Project Name & Description:

Human Emotion Recognition: Emotions play an important role in mediating and facilitating human interactions. Thus, understanding emotions often provides context to social communications that seem strange or complex. There are several ways to recognize emotions, including voice intonation and body language, as well as more complex methods like electroencephalography. One of the simplest and most effective methods is to observe facial expressions. Human beings express seven basic emotions: angry, happiness, fear, disgust, sad, surprise and neutral. These emotions are universal across cultures. Based on the facts above, it is evident that human emotions have a deep impact on actions and reactions. So, in this project, we are going to identify principal human emotions in real time.

Participants & Roles:

Tejaswini Gajula: tejaswinigajula@my.unt.edu
Yaswanth Bandaru: yaswanthbandaru@my.unt.edu
Sai Tarun Gunda: SaiTarunGunda@my.unt.edu
Kushal Patel: kushalpatel2@my.unt.edu

Workflow:

Communication:

- Weekly zoom meetings with the team.
- Email correspondence.
- Microsoft team communications

Google drive:

- We will use google drive for our shared documents and files.

Github Repository:

- Our code and data will be stored here.
- Used for version control.

Goal and Objectives:

Motivation:

Studying human emotion will open many possibilities for research currently underway on human emotion, humanoid robots, medical research, etc. Identifying human emotions remains one of the most challenging research topics today. Human behavior is better understood by understanding human emotions. This has a large contribution to the well-being of society, which is explained by its importance.

Significance:

Our application helps in identifying the human emotions by extracting the facial features in real time. There are many applications that revolve around emotion recognition. By understanding human emotions, professors can analyze the students' engagement in class and improve their methods of teaching. Autistic individuals have difficulty interpreting emotions. A system that recognizes emotions would help them improve their interactions with others. By identifying human emotion in places like restaurants, customer service can be enhanced and human-robot interaction can be improved.

Milestones:

1. Requirement Analysis
2. System Design
3. Implementation
4. Testing

The application development is divided into 5 phases.

1. Collection and pre-processing of data.
2. Extracting features.
3. Train the data on different classification models.
4. Perform hyper parameter tuning and cross validation.
5. Comparing different model performances.

Objectives:

The goal of this project is to construct a model that categorizes the main emotions of humans. Initially, we perform data cleaning and augmentation techniques such as rotating and resizing images, normalizing and affine transforming them. Then we remove noise and extract Gabor, Local Binary Pattern (LBP), HOG, and Spatial features. Finally, features are fed into deep learning models such as CNN and LSTM, and performance is measured with metrics such as AUC ROC and classification report.

Features:

- Gabor
- LBP (Local Binary Pattern)
- HOG
- Noise

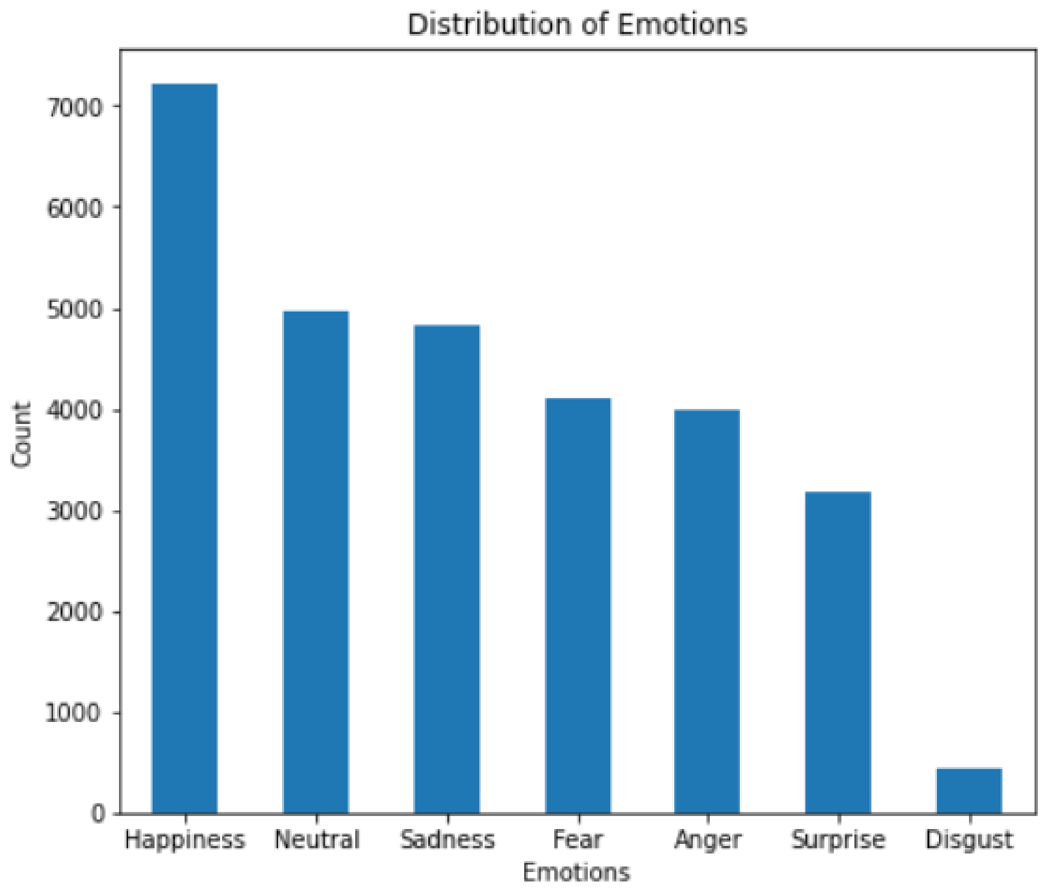
Related Work

In order to improve human-machine interaction, emotion recognition is an important area of research. It is more difficult to acquire emotions due to their complexity. It has been proposed that Quondam works capture emotion by capturing unimodal signals, such as only facial expressions or vocal inputs. With the introduction of the idea of multimodal emotion recognition, the detection of the machine has become more accurate. Additionally, deep learning with neural networks increased the success rate of machines in recognizing emotion. Researchers have studied deep learning techniques with diverse types of inputs such as audio-visual input, facial expressions, body movements, EEG signals, and brainwaves related to human behavior. Several issues in this area still need to be addressed to build an effective system that can recognize and classify emotions with greater accuracy.

Dataset

Kaggle provides the FER-2013 dataset, which we used. FER-2013 dataset has 35,887 48x48 pixel grayscale images of faces and contains 7 categories of facial expressions, with distributions of Angry (4,953), Disgust (547), Fear (5,121), Happy (8,989), Sad (6,077), Surprise (4,002), and Neutral (6,198).

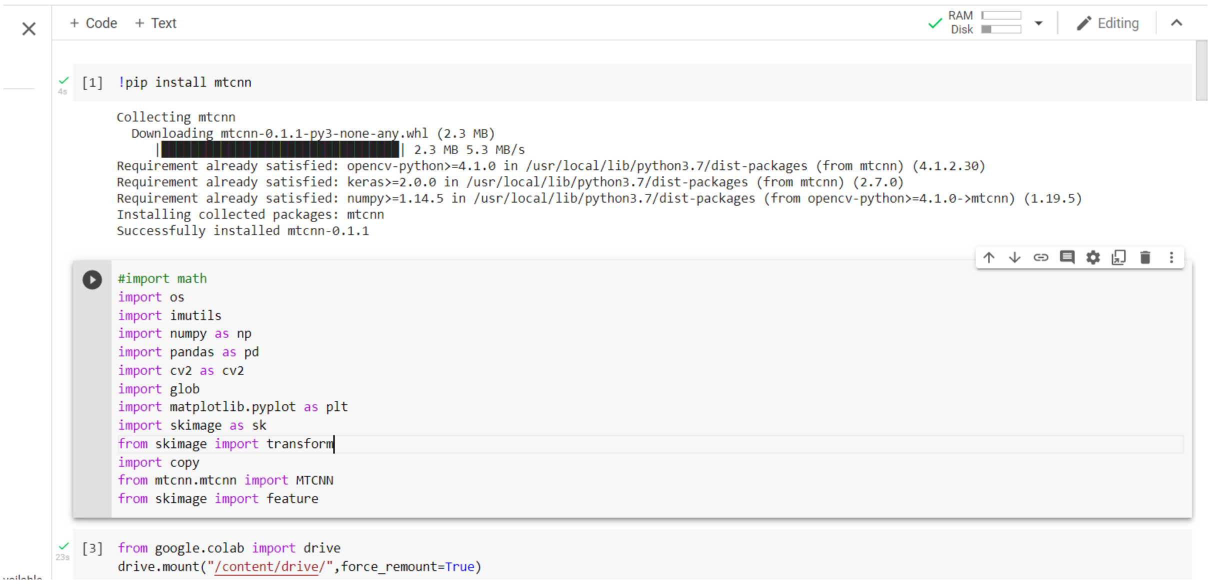
Analysis



Implementation

Our plan for implementation is as follow:

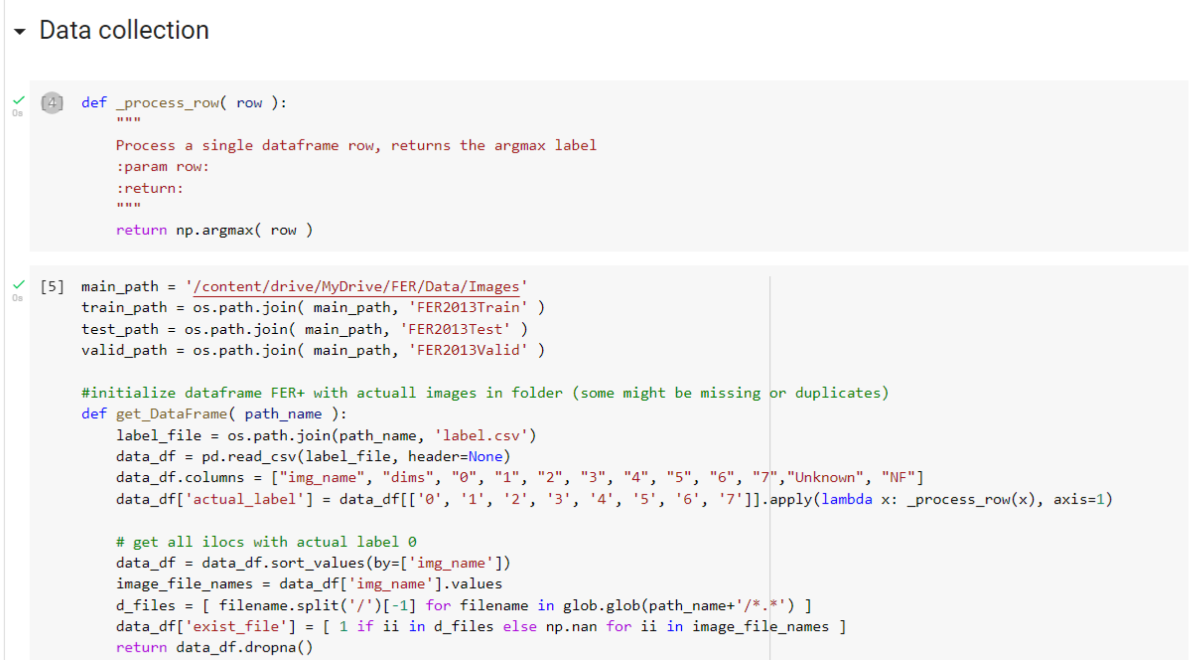
- Data Augmentation
- Extract faces using MTCNN
- Extract Features using HOG, Gabor and LBP
- Split the data into train and test
- Train CNN, LSTM and test them to get the best accurate model.



```
[1] !pip install mtcnn

    Collecting mtcnn
      Downloading mtcnn-0.1.1-py3-none-any.whl (2.3 MB)
         |████████████████████████████████| 2.3 MB 5.3 MB/s
    Requirement already satisfied: opencv-python>=4.1.0 in /usr/local/lib/python3.7/dist-packages (from mtcnn) (4.1.2.30)
    Requirement already satisfied: keras>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from mtcnn) (2.7.0)
    Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages (from opencv-python>=4.1.0->mtcnn) (1.19.5)
    Installing collected packages: mtcnn
    Successfully installed mtcnn-0.1.1
```

```python
#import math
import os
import imutils
import numpy as np
import pandas as pd
import cv2 as cv2
import glob
import matplotlib.pyplot as plt
import skimage as sk
from skimage import transform
import copy
from mtcnn.mtcnn import MTCNN
from skimage import feature
```

```python
[3] from google.colab import drive
    drive.mount("/content/drive/",force_remount=True)
```



Data collection

```python
[4] def _process_row( row ):
        """
        Process a single dataframe row, returns the argmax label
        :param row:
        :return:
        """
        return np.argmax( row )
```

```python
[5] main_path = '/content/drive/MyDrive/FER/Data/Images'
    train_path = os.path.join( main_path, 'FER2013Train' )
    test_path = os.path.join( main_path, 'FER2013Test' )
    valid_path = os.path.join( main_path, 'FER2013Valid' )

    #initialize dataframe FER+ with actuall images in folder (some might be missing or duplicates)
    def get_DataFrame( path_name ):
        label_file = os.path.join(path_name, 'label.csv')
        data_df = pd.read_csv(label_file, header=None)
        data_df.columns = ["img_name", "dims", "0", "1", "2", "3", "4", "5", "6", "7","Unknown", "NF"]
        data_df['actual_label'] = data_df[['0', '1', '2', '3', '4', '5', '6', '7']].apply(lambda x: _process_row(x), axis=1)

        # get all ilocs with actual label 0
        data_df = data_df.sort_values(by=['img_name'])
        image_file_names = data_df['img_name'].values
        d_files = [ filename.split('/')[-1] for filename in glob.glob(path_name+'/*.*') ]
        data_df['exist_file'] = [ 1 if ii in d_files else np.nan for ii in image_file_names ]
        return data_df.dropna()
    #reads image in cv2 format, transforms to grayscale since many operations needs this
    def get_Image( path ):
        return cv2.imread( path, cv2.IMREAD_GRAYSCALE )

    #converts gray image to string
    def _grayimg2string( img ):
        return str(img.flatten().tolist()).replace(']','').replace('[','')

    #converts string to gray image
    def _string2grayimage( img, size=(48,48) ):
        return np.array(img.split(', ')).astype(np.uint8).reshape( size )


    #creates dataframes per emotion loading images in gray scale
    def get_emotionsDataFrames( path_name, data_df ):
        for ii in data_df['actual_label'].unique():
            dd = data_df[data_df['actual_label']==ii][['img_name','actual_label']]
            dd['img_str'] = dd['img_name'].apply(lambda x: _grayimg2string( get_Image( path_name+'/'+x ) ) )
            dd.to_csv(path_name+'/'+f'labels_{ii}.csv', index=False)
        return
```

```
[7] get_emotionsDataFrames( train_path, get_DataFrame( train_path ) )
    get_emotionsDataFrames( test_path, get_DataFrame( test_path ) )
    get_emotionsDataFrames( valid_path, get_DataFrame( valid_path ) )
```

Data cleaning

```
[8] #For plotting image
    def show( img ):
        plt.imshow( img, cmap='gray' )
        return

    #Cleaning noise
    def Denoising( img, h=10, templateWindowSize=7, searchWindowSize=21 ):
        return cv2.fastNlMeansDenoising(img,None,h,templateWindowSize,searchWindowSize)
```

```
[9] #detect faces with mtcnn
    # we use this library after reading https://towardsdatascience.com/face-detection-models-which-to-use-and-why-d263e82c302c
    def ReturnLocalizedFace( img ):
        detector = MTCNN()
        faces = detector.detect_faces( cv2.cvtColor( img, cv2.COLOR_GRAY2RGB ) )
        return faces

    #draw faces result from mtcnn (https://towardsdatascience.com/face-detection-using-mtcnn-a-guide-for-face-extraction-with-a-focus-on-speed-c6d59f82d49)
    def DrawFacesMTCNN( img, faces ):
        plt.imshow( img, cmap='gray' )
        ax = plt.gca()
        for face in faces:
            x, y, w, h = face['box']
            rect = plt.Rectangle( (x,y), w, h , fill=False, color='green', lw=1 )
            ax.add_patch( rect )
            for key, value in face['keypoints'].items():
                dot = plt.Circle( value, radius=3, color='red' )
                ax.add_patch( dot )
        plt.show()
        return
```

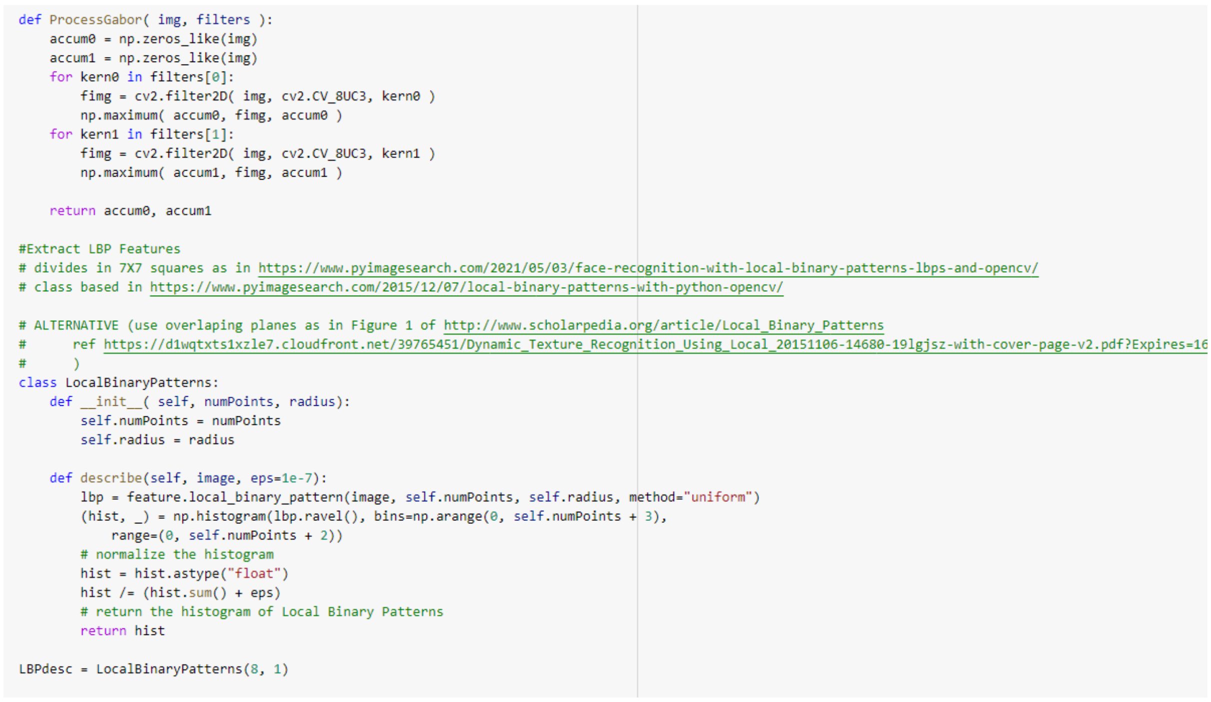
```python
#detect faces and uses eye-eye position to rotates and match a given "standard"
def RotatetoFrontalFace( img, faces ):
    #target positions from image FER2013Test/fer0032232.png
    le_t = np.array([13,18],dtype=float)
    re_t = np.array([35,18],dtype=float)
    no_t = np.array([24,30],dtype=float)
    ml_t = np.array([17,41],dtype=float)
    mr_t = np.array([31,41],dtype=float)
    eye_separation = re_t[0]-le_t[0]

    #faces = ReturnLocalizedFace( img )
    if len(faces)!=1:
        return None

    le=faces[0].get('keypoints').get('left_eye')
    if not le:
        return None
    le=np.array(le,dtype=float)
    re=faces[0].get('keypoints').get('right_eye')
    if not re:
        return None
    re=np.array(re,dtype=float)

    lr = re - le
    lr_n = np.linalg.norm(lr)
    c_alph = lr[0]/lr_n
    sgn = np.sign(lr[1])
    s_alph = sgn*np.sqrt(1-c_alph**2)
    scale = eye_separation/lr_n

    rot_mat = np.array([[  scale*c_alph,  scale*s_alph,  -scale*c_alph*le[0]+le_t[0]-scale*s_alph*le[1] ],
                        [ -scale*s_alph,  scale*c_alph,  -scale*c_alph*le[1]+le_t[1]+scale*s_alph*le[0] ]])

    new_img = cv2.warpAffine( img, rot_mat)
    #return crop image in size of "standard"
    return new_img[0:48, 0:48]
```

Feature extraction

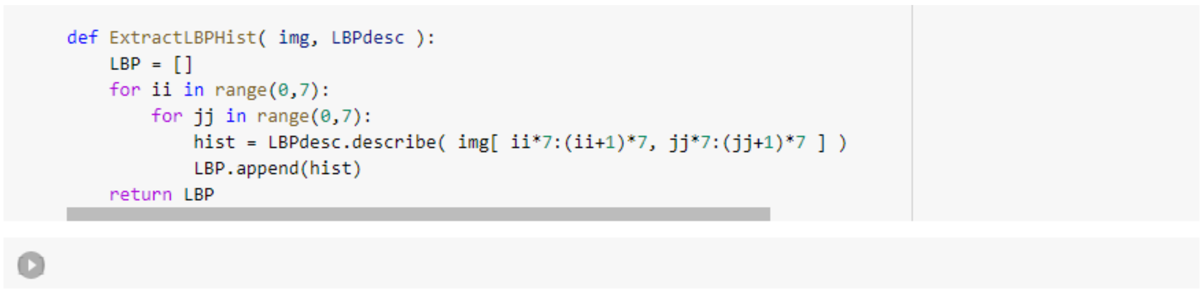
Here we assume that:

1. A an image **img** was read with **get_Image** (or loaded from new dataframes, ie 'labels_0.csv')
2. Then **img = Denoising(img)**
3. **faces = ReturnLocalizedFace(img)**
4. **img = RotatetoFrontalFace(img, faces)** before extracting features.

```python
[10] #Extracts HOG
     def ExtractHOG( img, win_size=(64,128) ):
         img = cv2.resize( img, win_size )
         #hog
         d = cv2.HOGDescriptor()
         hog = d.compute( img )
         return hog

     #Extract Gabor
     # based on https://cvtuts.wordpress.com/2014/04/27/gabor-filters-a-practical-overview/
     # TODO (change to frequency domain as in https://www.ini.rub.de/upload/file/1470692845_33efbf50567f9d637771/LadEtAl1993.pdf
     #        and file:///tmp/mozilla_gabriel0/ICCV2005-ZhangShan-LGBP.pdf)
     def BuildGaborFilters( ksize=11, sigma=3, lambd=9, gamma=0.5, psi=0. ):
         filters0=[]
         filters1=[]
         for theta in np.arange( 0, np.pi, np.pi/8 ):
             kern0 = cv2.getGaborKernel( (ksize,ksize), sigma, theta, lambd, gamma, psi, ktype=cv2.CV_32F)
             kern0 /= 1.5*kern0.sum()
             kern1 = cv2.getGaborKernel( (ksize,ksize), sigma/np.sqrt(2), theta, lambd/np.sqrt(2), gamma, psi, ktype=cv2.CV_32F)
             kern1 /= 1.5*kern0.sum()
             filters0.append(kern0)
             filters1.append(kern1)

         return filters0, filters1

     filters = BuildGaborFilters()
```



```
def ProcessGabor( img, filters ):
    accum0 = np.zeros_like(img)
    accum1 = np.zeros_like(img)
    for kern0 in filters[0]:
        fimg = cv2.filter2D( img, cv2.CV_8UC3, kern0 )
        np.maximum( accum0, fimg, accum0 )
    for kern1 in filters[1]:
        fimg = cv2.filter2D( img, cv2.CV_8UC3, kern1 )
        np.maximum( accum1, fimg, accum1 )

    return accum0, accum1

#Extract LBP Features
# divides in 7X7 squares as in https://www.pyimagesearch.com/2021/05/03/face-recognition-with-local-binary-patterns-lbps-and-opencv/
# class based in https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/

# ALTERNATIVE (use overlaping planes as in Figure 1 of http://www.scholarpedia.org/article/Local_Binary_Patterns
#      ref https://d1wqtxts1xzle7.cloudfront.net/39765451/Dynamic_Texture_Recognition_Using_Local_20151106-14680-19lgjsz-with-cover-page-v2.pdf?Expires=16
#      )
class LocalBinaryPatterns:
    def __init__( self, numPoints, radius):
        self.numPoints = numPoints
        self.radius = radius

    def describe(self, image, eps=1e-7):
        lbp = feature.local_binary_pattern(image, self.numPoints, self.radius, method="uniform")
        (hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, self.numPoints + 3),
            range=(0, self.numPoints + 2))
        # normalize the histogram
        hist = hist.astype("float")
        hist /= (hist.sum() + eps)
        # return the histogram of Local Binary Patterns
        return hist

LBPdesc = LocalBinaryPatterns(8, 1)
```



```
def ExtractLBPHist( img, LBPdesc ):
    LBP = []
    for ii in range(0,7):
        for jj in range(0,7):
            hist = LBPdesc.describe( img[ ii*7:(ii+1)*7, jj*7:(jj+1)*7 ] )
            LBP.append(hist)
    return LBP
```

GitHub Link: https://github.com/yaswanth-bandaru/Facial_Emotion_Recognition.git

Project Management

Implementation Status Report:

- **Work Completed:** Dataset collection and Preprocessing.
    - **Description:** Data has been collected from Kaggle and then applied data augmentation techniques and extracted faces using MTCNN and finally extracted features using HOG, Gabor and LBP.
    - **Contribution:** The team members contributed equally.

- **Work To Be Completed:**
    - **Description:** Below are the details about next steps
        - Train a model with training data
        - Predict for validation set and see confusion matrix and metrics

    - **Responsibility:** Using different combinations of features, each of us would conduct 1 or 2 modeling experiments.

References:

- Chendi Wang, Southeast University, China. “Human Emotional Facial Expression Recognition”.
  - https://arxiv.org/ftp/arxiv/papers/1803/1803.10864.pdf
- Jielong Tanga, Xiaotian Zhoub, Jiawei Zheng.” Design of Intelligent classroom facial recognition based on Deep Learning”.
  - https://www.researchgate.net/publication/331681396_Design_of_Intelligent_classroom_facial_recognition_based_on_Deep_Learning
- Reeshad Khan & Omar Sharif. “A Literature Review on Emotion Recognition using Various Methods”.
  - https://core.ac.uk/download/pdf/231150449.pdf
- M.Regina, Dr.M.S.Josephine, Dr.VJeyabalraja. “Analyzing the Human Emotion based on Facial Expression using Gabor Features with ensemble and CNN Based Classifiers from a video sequence”.
  - https://acadpubl.eu/hub/2018-119-16/1/28.pdf
- Zhenjie Song, “Facial Expression Emotion Recognition Model Integrating Philosophy and Machine Learning Theory”.
  - https://www.frontiersin.org/articles/10.3389/fpsyg.2021.759485/full