Mohammad Omar Sadat, CS660

Building a facial Expression Recognition System to assess Customer's Experiences

Understanding Problem Statement:

To develop Facial Expression Recognition system where our code receives a stream of pictures either from the web camera in our laptop or the CCTV camera attached in any transportation like (NJTransit) or any Retail stores to identify passengers or customer's satisfactions. This system will replace the traditional conducting a Survey's to find out what customer's wants & needs, while interacting with product brand or traveling in a public transport like NJ Transit.

Tools used to implement this project:

- Google Colab
- GPU
- Datasets of Images
- Python (3.0 ++ versions) [installed in Local machine]
- Jupyter Notebook [installed in Local machine]
- Tensorflow 2.0 [installed in Local machine]
- OpenCV [installed in Local machine]
- Dlib [installed in Local machine]

Following Concepts are required:

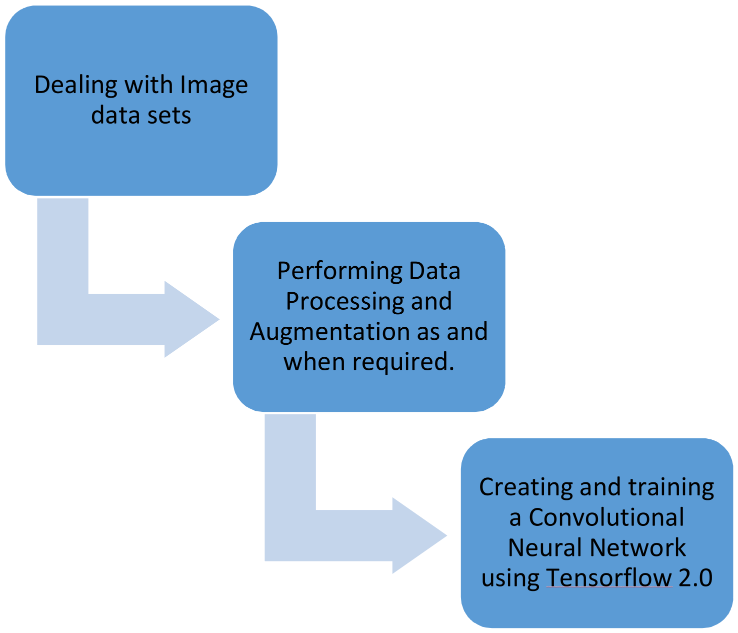
- Python programming language
- Artificial Neural Networks
- Image processing
- Convolutional neural networks

DataSet Description

The data consists of 48x48 pixel grayscale images of faces. This will be double in size (96x96) in the phase of training as DLib tools require a bit large zoomed picture to execute its machine learning algorithms. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

We have used 28,709 pictures for training data set and 7,000 pictures for testing dataset.

Process at a glance


Dealing with Image data sets
Performing Data Processing and Augmentation as and when required.
Creating and training a Convolutional Neural Network using Tensorflow 2.0

Process in details

- Created a Model that can detect different expressions in a face given their images.
- Performed data augmentation on our images to make our model more robust.
- Also performed two phase training on our models, where we have frozen a few layers before our model starts overfitting.
- Also evaluated our model on Test set and applied real time prediction.

Training Approaches

- **The seven basic universal expressions: -**
  - **Neutral, Happiness, Sadness, Anger, Surprise, Fear, and Disgust. Based on that following approaches have been adopted**

1. Performing face detection and cropping face images in our training dataset.
   - Although we have closely cropped faces in our dataset, they are very noisy.

- By filtering out all images which were not detected by face detection algorithm we can make our model more robust.

2. . Increase Number of phases and Decrease number of epochs per phase while training.

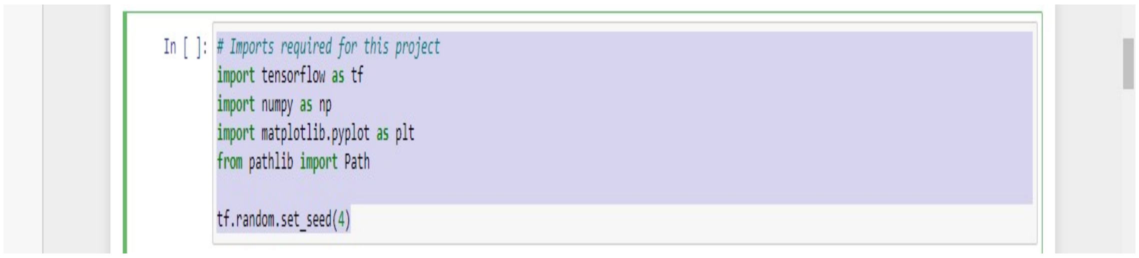
   For example:

   You can train a model using a phase 1 with 6 epochs repeated by phase 2 with 4 epochs for 2 times.

3. Try out a better version of EfficientNet.
   - There are even better versions of EfficientNet such as B3 to B7.
   - We have used EfficientNetB2 just to keep it less computationally intensive. So that it can be executed even on Low hardware specification.

CODE Explanations:

<u>\# Imports required for this project</u>



```
In [ ]: # Imports required for this project
        import tensorflow as tf
        import numpy as np
        import matplotlib.pyplot as plt
        from pathlib import Path

        tf.random.set_seed(4)
```

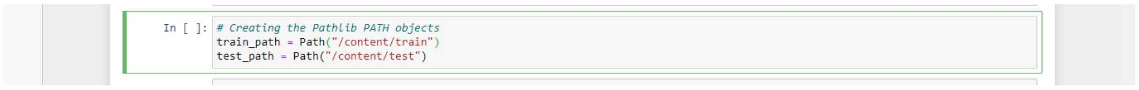
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
tf.random.set_seed(4)
```

Descriptions

- Here we are using tensorflow library for creating Deep Learning models directly from the images we considered during the training phase.
- Numpy library is used for mathematical operations on arrays.
- Matplotlib is a library which we have used with its function pyplot for plotting the lines or area in the grayscal images based on the pixels variations.
- Pathlib module in Python provides **various classes representing file system paths with semantics appropriate** for different operating systems.

Mohammad Omar Sadat, CS660

Creating the Pathlib PATH objects



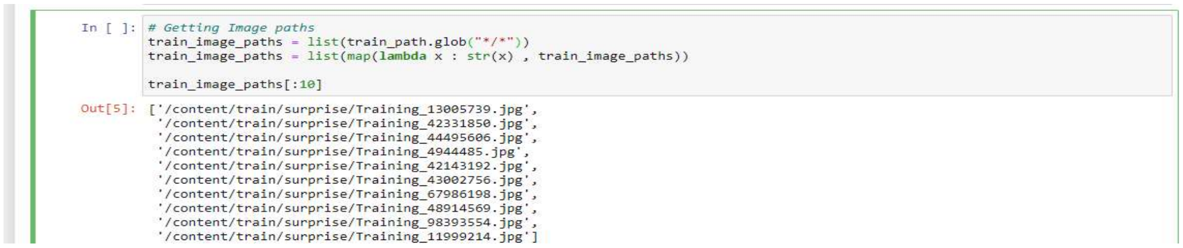
```
In [ ]: # Creating the Pathlib PATH objects
        train_path = Path("/content/train")
        test_path = Path("/content/test")
```

```
train_path = Path("/content/train")
test_path = Path("/content/test")
```

Descriptions

Here we have used for PathLib library to create object for the directory locations of the pictures used for training & test purposes.

Getting Image paths



```
In [ ]: # Getting Image paths
        train_image_paths = list(train_path.glob("*/*"))
        train_image_paths = list(map(lambda x : str(x) , train_image_paths))

        train_image_paths[:10]

Out[5]: ['/content/train/surprise/Training_13005739.jpg',
         '/content/train/surprise/Training_42331850.jpg',
         '/content/train/surprise/Training_44495606.jpg',
         '/content/train/surprise/Training_4944485.jpg',
         '/content/train/surprise/Training_42143192.jpg',
         '/content/train/surprise/Training_43002756.jpg',
         '/content/train/surprise/Training_67986198.jpg',
         '/content/train/surprise/Training_48914569.jpg',
         '/content/train/surprise/Training_98393554.jpg',
         '/content/train/surprise/Training_11999214.jpg']
```

```
train_image_paths = list(train_path.glob("*/*"))
train_image_paths = list(map(lambda x : str(x) , train_image_paths))
train_image_paths [:10]
```

Descriptions

Here glob is used to match file paths of the directory and the value is stored in train_iamge_paths.
Lambda function is used to create a function without declaring its name and return the value of the images file locations used for training purpose.
Map function is also used to process the locations of the image files without declaring any explicit loop.
For training purpose, we used 10 random images in array

# **Getting their respective labels**

```
def get_label(image_path):
   return image_path.split("/")[-2]
train_image_labels = list(map(lambda x : get_label(x) , train_image_paths))
train_image_labels[:10]
```

Descriptions

Here we have used get_label function to get the respective labels of images that we have segregated in Seven types based on the emotions varieties of Seven (7) types.

# Converting the labels in to array & select the model

```
from sklearn.preprocessing import LabelEncoder
Le = LabelEncoder()
train_image_labels = Le.fit_transform(train_image_labels)
train_image_labels[:10]


train_image_labels = tf.keras.utils.to_categorical(train_image_labels)
train_image_labels[:10]

from sklearn.model_selection import train_test_split
Train_paths , Val_paths , Train_labels , Val_labels = train_test_split(train_image_paths ,
train_image_labels , test_size = 0.25)
```

Descriptions

Scikit-learn (Sklearn) is the **most useful and robust library for machine learning in Python**. Here It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

# Compute class weights

```
classTotals = Train_labels.sum(axis=0)
classWeight = classTotals.max() / classTotals
class_weight = {e : weight for e , weight in enumerate(classWeight)}
print(class_weight)
```

Descriptions

As we are working in a multi class environment, we need to give weight balance among all classes.

# Function used for Transformation

```
def load(image , label):
  image = tf.io.read_file(image)
  image = tf.io.decode_jpeg(image , channels = 3)
  return image , label
```

Descriptions

To process and apply augmentations on our data we have called a function named load and pass image and label by reading file path.

Mohammad Omar Sadat, CS660

# Define IMAGE SIZE and BATCH SIZE

```
IMG_SIZE = 96
BATCH_SIZE = 32

# Basic Transformation
resize = tf.keras.Sequential([
    tf.keras.layers.experimental.preprocessing.Resizing(IMG_SIZE, IMG_SIZE)
])

# Data Augmentation
data_augmentation = tf.keras.Sequential([
    tf.keras.layers.experimental.preprocessing.RandomFlip("horizontal"),
    tf.keras.layers.experimental.preprocessing.RandomRotation(0.1),
    tf.keras.layers.experimental.preprocessing.RandomZoom(height_factor = (-0.1, -0.05))
])
```

Descriptions

For data transformation purpose in to object we defined the image sized and batch size. We also used resize function for all images too. We doubled the size of our images from 48 pixels to 96 too, as small images are not enough for the deep learning processing. For Data augmentation purpose, we used RandomFlip, RandomRotation and Random Zoom funtions. We use Data Augmentation as Data augmentation in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data.

# Function used to Create a Tensorflow Data Object

```
AUTOTUNE = tf.data.experimental.AUTOTUNE
def get_dataset(paths , labels , train = True):
    image_paths = tf.convert_to_tensor(paths)
    labels = tf.convert_to_tensor(labels)

    image_dataset = tf.data.Dataset.from_tensor_slices(image_paths)
    label_dataset = tf.data.Dataset.from_tensor_slices(labels)

    dataset = tf.data.Dataset.zip((image_dataset , label_dataset))

    dataset = dataset.map(lambda image , label : load(image , label))
    dataset = dataset.map(lambda image, label: (resize(image), label) ,
num_parallel_calls=AUTOTUNE)
    dataset = dataset.shuffle(1000)
    dataset = dataset.batch(BATCH_SIZE)
    if train:
        dataset = dataset.map(lambda image, label: (data_augmentation(image), label) ,
num_parallel_calls=AUTOTUNE)

    dataset = dataset.repeat()
    return dataset
```

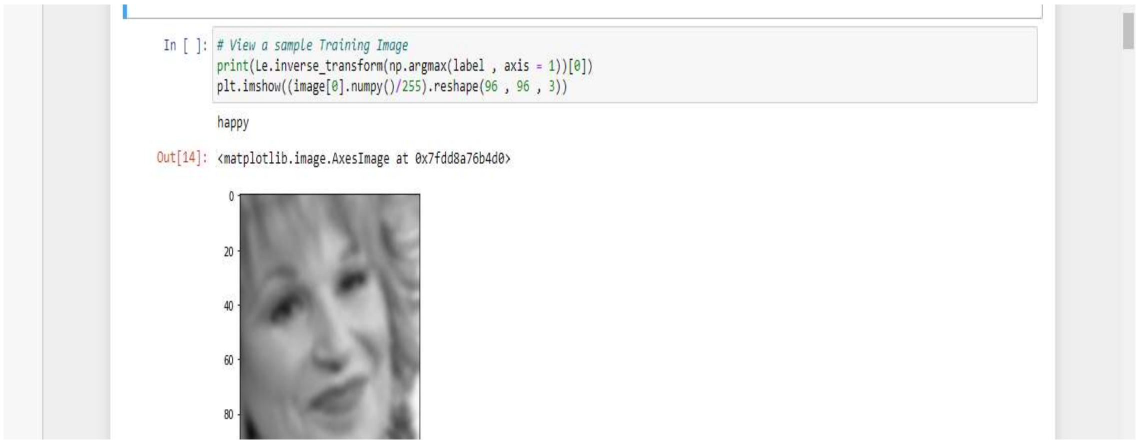
Descriptions
Here we have implemented function to create Tensorflow data object.

# Creating Train Dataset object and Verifying it

```
%time train_dataset = get_dataset(Train_paths , Train_labels)

image , label = next(iter(train_dataset))
print(image.shape)
print(label.shape)
```

# View a sample Training Image

```
print(Le.inverse_transform(np.argmax(label , axis = 1))[0])
plt.imshow((image[0].numpy()/255).reshape(96 , 96 , 3))
```



```
In [ ]: # View a sample Training Image
        print(Le.inverse_transform(np.argmax(label , axis = 1))[0])
        plt.imshow((image[0].numpy()/255).reshape(96 , 96 , 3))

        happy

Out[14]: <matplotlib.image.AxesImage at 0x7fdd8a76b4d0>
```

# Building EfficientNet model

```
from tensorflow.keras.applications import EfficientNetB2

backbone = EfficientNetB2(
    input_shape=(96, 96, 3),
    include_top=False
)

model = tf.keras.Sequential([
    backbone,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(7, activation='softmax')
])

model.summary()
```

# Building EfficientNet model

```
from tensorflow.keras.applications import EfficientNetB2

backbone = EfficientNetB2(
    input_shape=(96, 96, 3),
    include_top=False
)

model = tf.keras.Sequential([
    backbone,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(7, activation='softmax')
])

model.summary()
```

# Compiling your model by providing the Optimizer , Loss and Metrics

```
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-
07),
    loss = 'categorical_crossentropy',
    metrics=['accuracy' ,
tf.keras.metrics.Precision(name='precision'),tf.keras.metrics.Recall(name='recall')]
)
```

# Train the model

```
history = model.fit(
    train_dataset,
    steps_per_epoch=len(Train_paths)//BATCH_SIZE,
    epochs=12,
    validation_data=val_dataset,
    validation_steps = len(Val_paths)//BATCH_SIZE,
    class_weight=class_weight
)

model.layers[0].trainable = False
```

# Defining our callbacks

```
checkpoint =
tf.keras.callbacks.ModelCheckpoint("best_weights.h5",verbose=1,save_best_only=True,save_weight
s_only = True)
early_stop = tf.keras.callbacks.EarlyStopping(patience=4)

model.summary()
```

# Train the model

```
history = model.fit(
    train_dataset,
    steps_per_epoch=len(Train_paths)//BATCH_SIZE,
    epochs=8,
    callbacks=[checkpoint , early_stop],
    validation_data=val_dataset,
    validation_steps = len(Val_paths)//BATCH_SIZE,
    class_weight=class_weight
)
```

Testing Phase

```
from tensorflow.keras.applications import EfficientNetB2

backbone = EfficientNetB2(
    input_shape=(96, 96, 3),
    include_top=False
)

model = tf.keras.Sequential([
    backbone,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(7, activation='softmax')
])

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-
07),
    loss = 'categorical_crossentropy',
    metrics=['accuracy' ,
tf.keras.metrics.Precision(name='precision'),tf.keras.metrics.Recall(name='recall')]
)
model.load_weights("best_weights.h5")
```

# Create a Dataset Object for 'Testing' Set just the way we did for Training and Validation

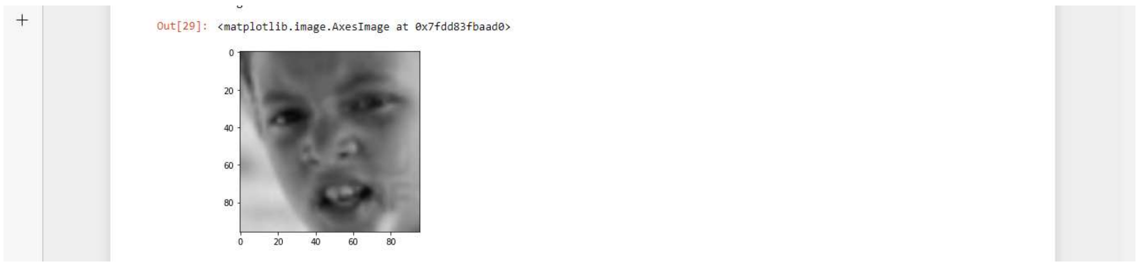
```
test_image_paths = list(test_path.glob("*/*"))
test_image_paths = list(map(lambda x : str(x) , test_image_paths))
test_labels = list(map(lambda x : get_label(x) , test_image_paths))

test_labels = Le.transform(test_labels)
test_labels = tf.keras.utils.to_categorical(test_labels)

test_image_paths = tf.convert_to_tensor(test_image_paths)
test_labels = tf.convert_to_tensor(test_labels)

def decode_image(image , label):
    image = tf.io.read_file(image)
    image = tf.io.decode_jpeg(image , channels = 3)
    image = tf.image.resize(image , [96 , 96] , method="bilinear")
    return image , label

test_dataset = (
     tf.data.Dataset
    .from_tensor_slices((test_image_paths, test_labels))
    .map(decode_image)
    .batch(BATCH_SIZE)
)
```

# Verify Test Dataset Object

```
image , label = next(iter(test_dataset))
print(image.shape)
print(label.shape)
```

# View a sample Validation Image

```
print(Le.inverse_transform(np.argmax(label , axis = 1))[0])
plt.imshow((image[0].numpy()/255).reshape(96 , 96 , 3))
```



```
Out[29]: <matplotlib.image.AxesImage at 0x7fdd83fbaad0>
```

# Evaluating the loaded model

```
loss, acc, prec, rec = model.evaluate(test_dataset)

print(" Testing Acc : " , acc)
print(" Testing Precision " , prec)
print(" Testing Recall " , rec)
```

Save Objects
# Save Model

```
model.save("FacialExpressionModel.h5")
```

# Save Label Encoder

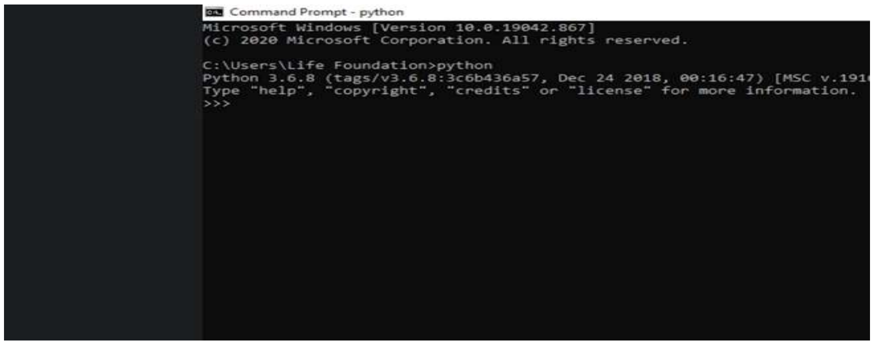
```
import pickle

def save_object(obj , name):
   pickle_obj = open(f"{name}.pck","wb")
   pickle.dump(obj, pickle_obj)
   pickle_obj.close()

save_object(Le, "LabelEncoder")
```

============================xxxxxx====================================

We should know that Google CoLab should be only used for large computational activities like Training Neural Networks and Google CoLab does not have any interface to connect to our local machine. That's why , when we are going to connect either our laptop web cam or any other CCTV camera, we have to install the codes in our local machine.

<u>Local Machine Setup</u>

Mohammad Omar Sadat, CS660

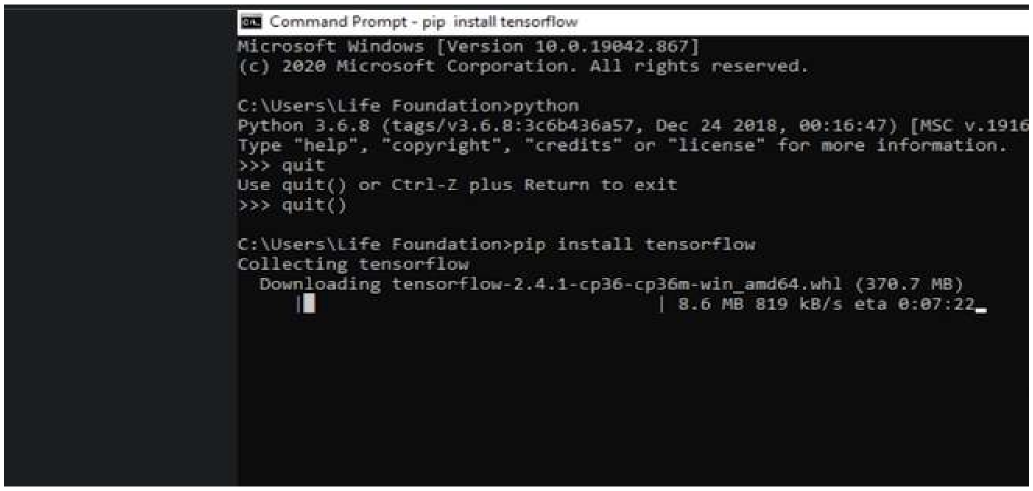


Command Prompt - python

```
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Life Foundation>python
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.191
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Install python (3) version or upper first.

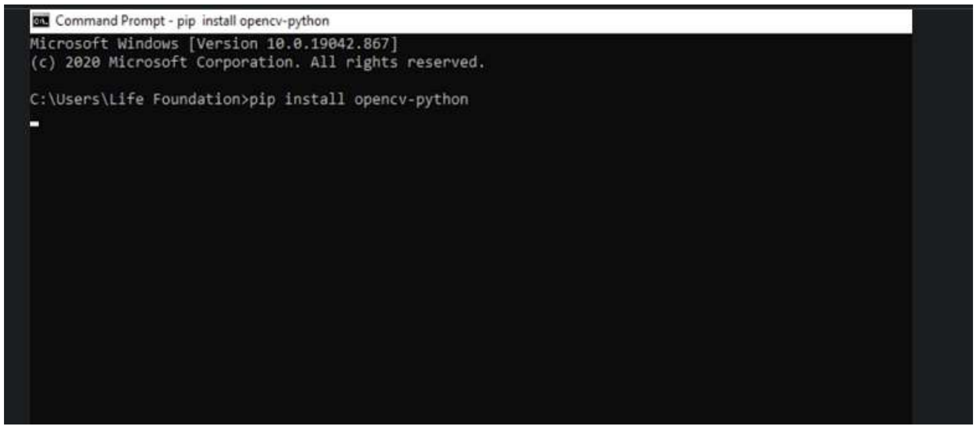
Install Tensorflow 2.0 (at least)



Command Prompt - pip install tensorflow

```
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Life Foundation>python
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916
Type "help", "copyright", "credits" or "license" for more information.
>>> quit
Use quit() or Ctrl-Z plus Return to exit
>>> quit()

C:\Users\Life Foundation>pip install tensorflow
Collecting tensorflow
  Downloading tensorflow-2.4.1-cp36-cp36m-win_amd64.whl (370.7 MB)
     |█                               | 8.6 MB 819 kB/s eta 0:07:22
```

Install OpenCV

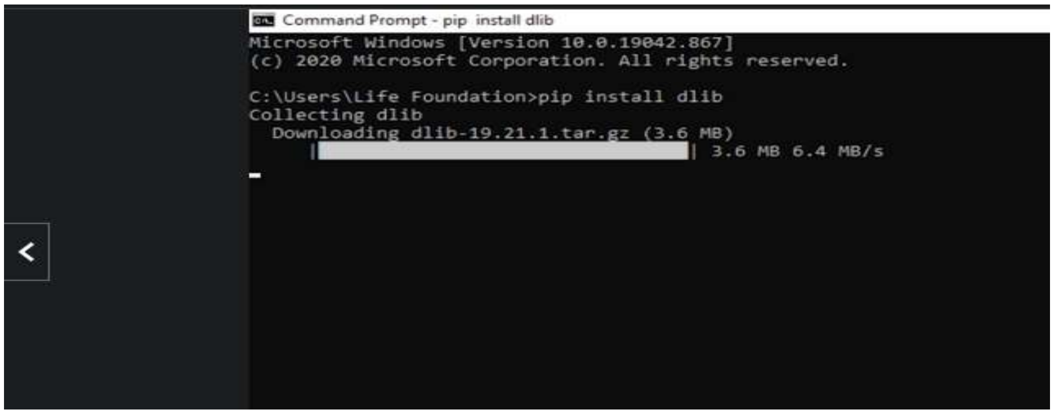


Command Prompt - pip install opencv-python

```
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Life Foundation>pip install opencv-python
```

Install DLib (toolkit for machine learning algorithm) [C++ compiler is required, which can be solved by installing Visual Studio]

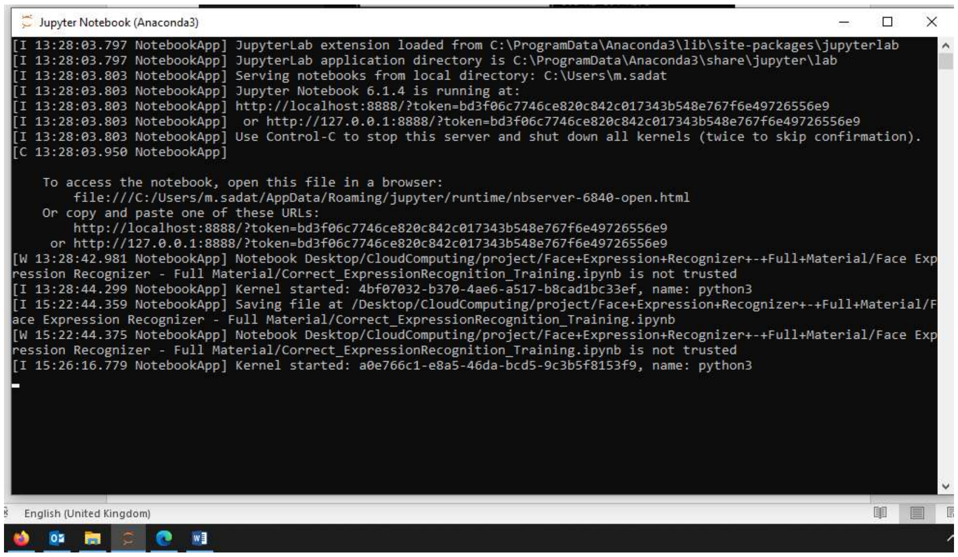
Mohammad Omar Sadat, CS660



Command Prompt - pip install dlib

```
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Life Foundation>pip install dlib
Collecting dlib
  Downloading dlib-19.21.1.tar.gz (3.6 MB)
     |████████████████████████████████| 3.6 MB 6.4 MB/s
```

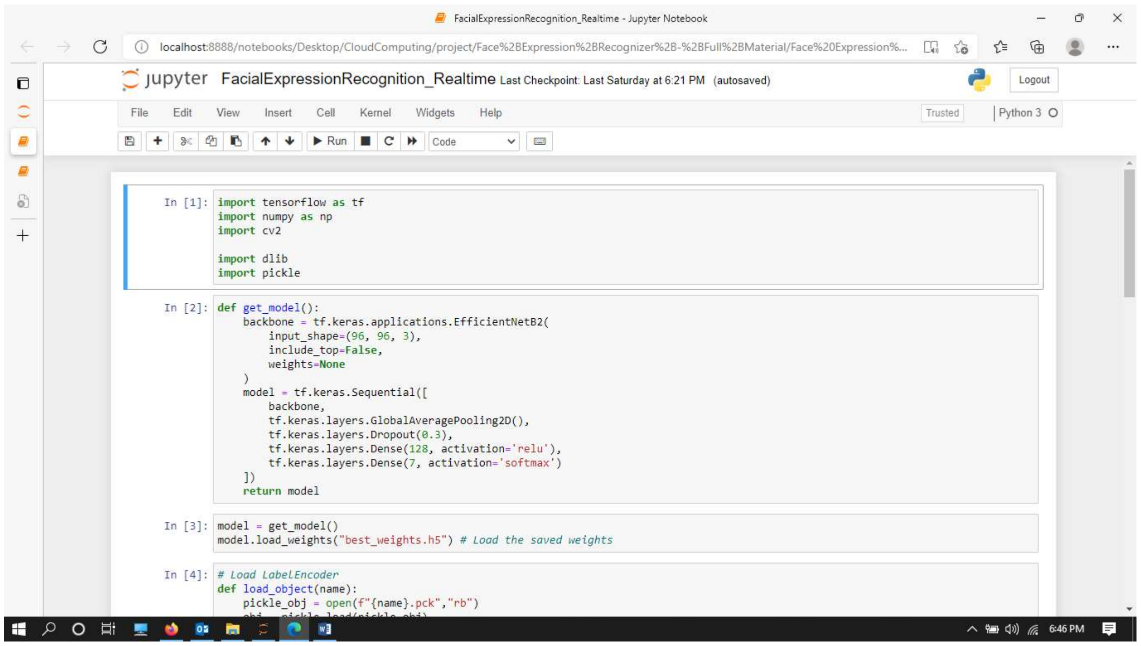
For Coding platform , we have used Jupyter Notebook



Jupyter Notebook (Anaconda3)

```
[I 13:28:03.797 NotebookApp] JupyterLab extension loaded from C:\ProgramData\Anaconda3\lib\site-packages\jupyterlab
[I 13:28:03.797 NotebookApp] JupyterLab application directory is C:\ProgramData\Anaconda3\share\jupyter\lab
[I 13:28:03.803 NotebookApp] Serving notebooks from local directory: C:\Users\m.sadat
[I 13:28:03.803 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[I 13:28:03.803 NotebookApp] http://localhost:8888/?token=bd3f06c7746ce820c842c017343b548e767f6e49726556e9
[I 13:28:03.803 NotebookApp]  or http://127.0.0.1:8888/?token=bd3f06c7746ce820c842c017343b548e767f6e49726556e9
[I 13:28:03.803 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 13:28:03.950 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/m.sadat/AppData/Roaming/jupyter/runtime/nbserver-6840-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=bd3f06c7746ce820c842c017343b548e767f6e49726556e9
     or http://127.0.0.1:8888/?token=bd3f06c7746ce820c842c017343b548e767f6e49726556e9
[W 13:28:42.981 NotebookApp] Notebook Desktop/CloudComputing/project/Face+Expression+Recognizer+-+Full+Material/Face Exp
ression Recognizer - Full Material/Correct_ExpressionRecognition_Training.ipynb is not trusted
[I 13:28:44.299 NotebookApp] Kernel started: 4bf07032-b370-4ae6-a517-b8cad1bc33ef, name: python3
[I 15:22:44.359 NotebookApp] Saving file at /Desktop/CloudComputing/project/Face+Expression+Recognizer+-+Full+Material/F
ace Expression Recognizer - Full Material/Correct_ExpressionRecognition_Training.ipynb
[W 15:22:44.375 NotebookApp] Notebook Desktop/CloudComputing/project/Face+Expression+Recognizer+-+Full+Material/Face Exp
ression Recognizer - Full Material/Correct_ExpressionRecognition_Training.ipynb is not trusted
[I 15:26:16.779 NotebookApp] Kernel started: a0e766c1-e8a5-46da-bcd5-9c3b5f8153f9, name: python3
```

Mohammad Omar Sadat, CS660


FacialExpressionRecognition_Realtime - Jupyter Notebook
localhost:8888/notebooks/Desktop/CloudComputing/project/Face%2BExpression%2BRecognizer%2B-%2BFull%2BMaterial/Face%20Expression%...
jupyter FacialExpressionRecognition_Realtime Last Checkpoint: Last Saturday at 6:21 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
6:46 PM

```
In [1]: import tensorflow as tf
        import numpy as np
        import cv2

        import dlib
        import pickle

In [2]: def get_model():
            backbone = tf.keras.applications.EfficientNetB2(
                input_shape=(96, 96, 3),
                include_top=False,
                weights=None
            )
            model = tf.keras.Sequential([
                backbone,
                tf.keras.layers.GlobalAveragePooling2D(),
                tf.keras.layers.Dropout(0.3),
                tf.keras.layers.Dense(128, activation='relu'),
                tf.keras.layers.Dense(7, activation='softmax')
            ])
            return model

In [3]: model = get_model()
        model.load_weights("best_weights.h5") # Load the saved weights

In [4]: # Load LabelEncoder
        def load_object(name):
            pickle_obj = open(f"{name}.pck","rb")
```

Codes for Output

```
import tensorflow as tf
import numpy as np
import cv2
import dlib
import pickle
```

All required libraries are import, like CV2 for real time video capturing, dlib for the tools of machine learning algorithms, pickle is used for converting python object to byte streams.

```
def get_model():
    backbone = tf.keras.applications.EfficientNetB2(
        input_shape=(96, 96, 3),
        include_top=False,
        weights=None
    )
    model = tf.keras.Sequential([
        backbone,
        tf.keras.layers.GlobalAveragePooling2D(),
        tf.keras.layers.Dropout(0.3),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(7, activation='softmax')
    ])
    return model
```

# Load the saved weights

```
model = get_model()
model.load_weights("best_weights.h5")
```

# Load LabelEncoder

```
def load_object(name):
    pickle_obj = open(f"{name}.pck","rb")
    obj = pickle.load(pickle_obj)
    return obj

Le = load_object("LabelEncoder")
```

Declare the function , named ProcessImage with image parameter for the processing of image tensors.

```
def ProcessImage(image):
    image = tf.convert_to_tensor(image)
    image = tf.image.resize(image , [96 , 96] , method="bilinear")
    image = tf.expand_dims(image , 0)
    return image

def RealtimePrediction(image , model, encoder_):
    prediction = model.predict(image)
    prediction = np.argmax(prediction , axis = 1)
    return encoder_.inverse_transform(prediction)[0]

def rect_to_bb(rect):
    x = rect.left()
    y = rect.top()
    w = rect.right() - x
    h = rect.bottom() - y
    return (x, y, w, h)
```

As we are using our Laptop bulletin camera, so we are using cv2.VideoCapture(0) . If you capture video from any other sources like CCTV camera, then we should use cv2.VideoCapture(1).

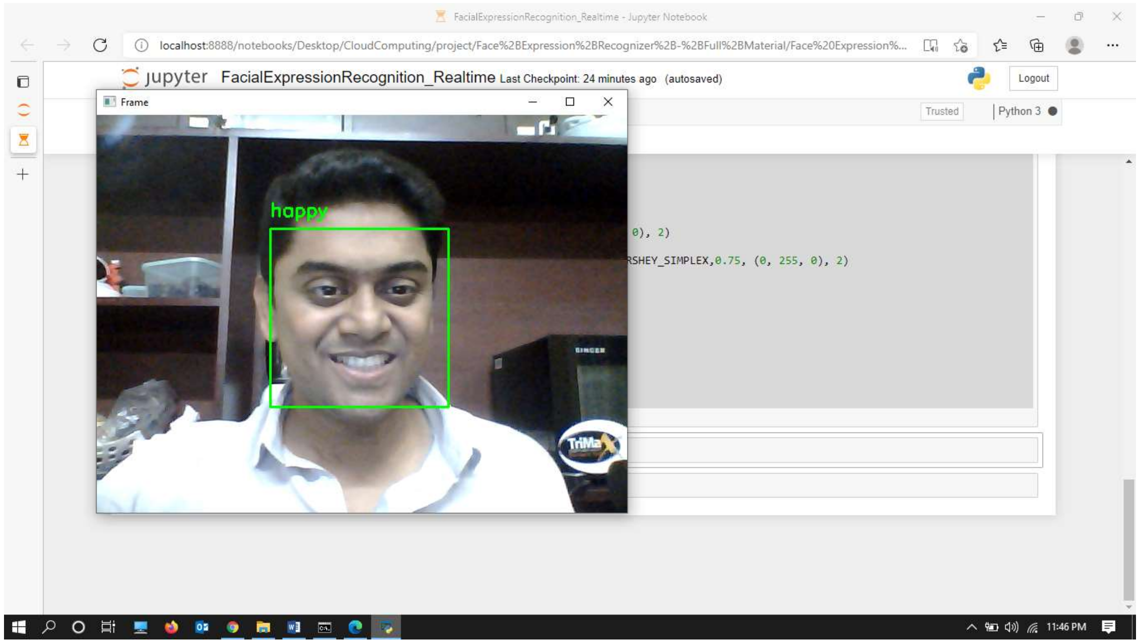
```
VideoCapture = cv2.VideoCapture(0)

detector = dlib.get_frontal_face_detector()

while True :

    ret , frame = VideoCapture.read()

    if not ret :
        break

    gray = cv2.cvtColor( frame , cv2.COLOR_BGR2GRAY)

    rects = detector(gray , 0)

    if len(rects) >= 1 :
        for rect in rects :
            (x , y , w , h) = rect_to_bb(rect)
            img = gray[y-10 : y+h+10 , x-10 : x+w+10]

            if img.shape[0] == 0 or img.shape[1] == 0 :
                cv2.imshow("Frame", frame)

            else :
                img = cv2.cvtColor(img , cv2.COLOR_GRAY2RGB)
                img = ProcessImage(img)
                out = RealtimePrediction(img , model , Le)
                cv2.rectangle(frame, (x, y), (x+w, y+h),(0, 255, 0), 2)
                z = y - 15 if y - 15 > 15 else y + 15
                cv2.putText(frame, str(out), (x, z), cv2.FONT_HERSHEY_SIMPLEX,0.75, (0, 255, 0), 2)

        cv2.imshow("Frame", frame)

    else :
        cv2.imshow("Frame", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

VideoCapture.release()
cv2.destroyAllWindows()
```

Real Time Output

Mohammad Omar Sadat, CS660

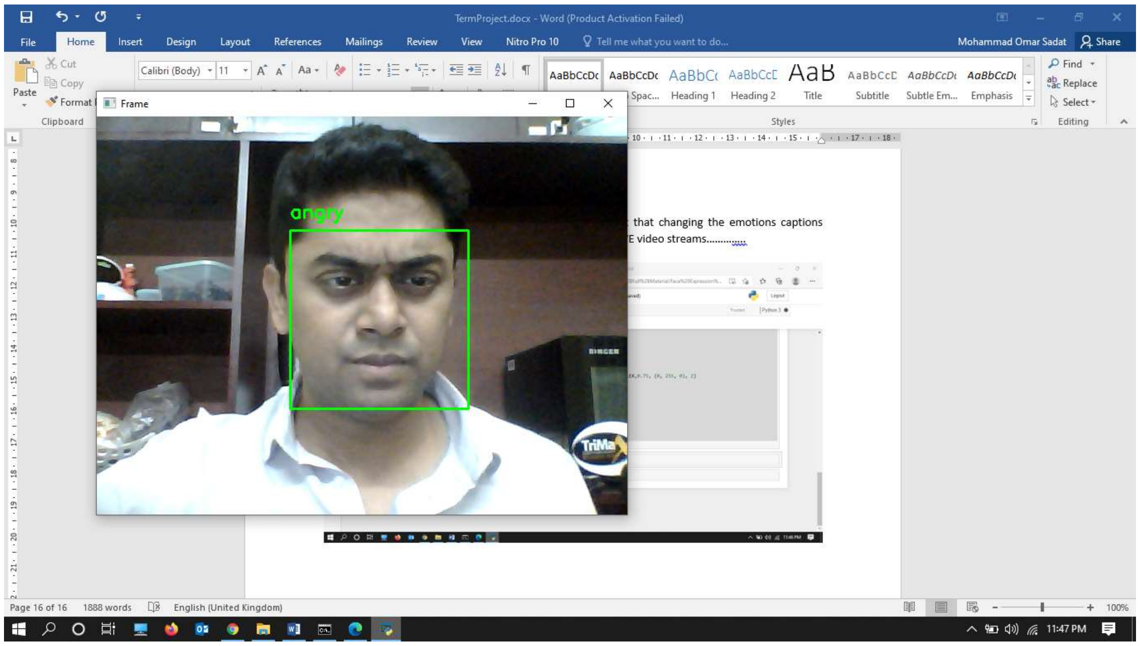
As I am using my own laptop camera to check the output that changing the emotions captions according to the change of my facial expressions based on LIVE video streams………….

Happy Customer/ passenger

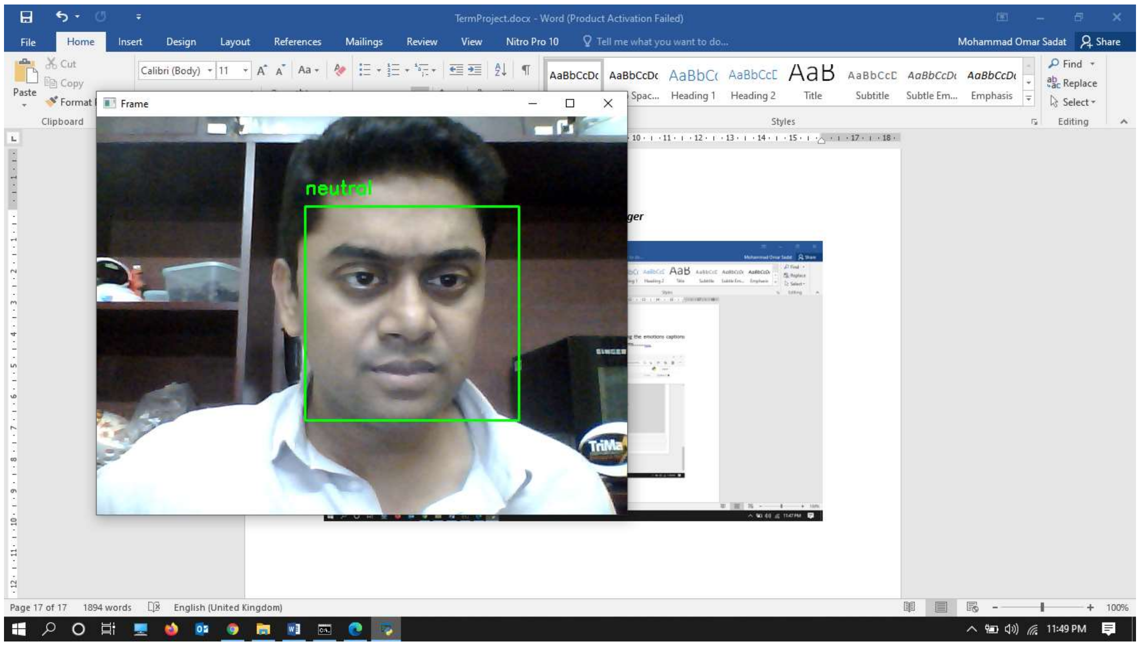

FacialExpressionRecognition_Realtime - Jupyter Notebook
localhost:8888/notebooks/Desktop/CloudComputing/project/Face%2BExpression%2BRecognizer%2B-%2BFull%2BMaterial/Face%20Expression%...
jupyter FacialExpressionRecognition_Realtime Last Checkpoint: 24 minutes ago (autosaved)
Logout
Trusted
Python 3
Frame
happy
11:46 PM

```
0), 2)
RSHEY_SIMPLEX,0.75, (0, 255, 0), 2)
```

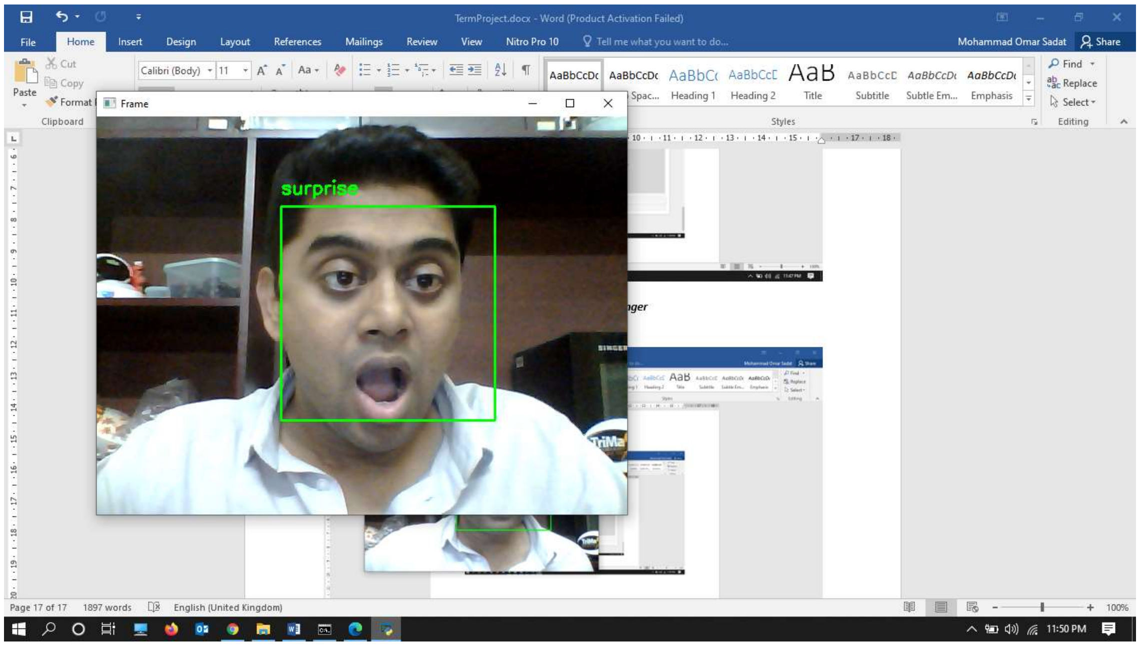
Angry Customer/ passenger



Neutral Customer/ passenger



Surprise Customer/ passenger



Reference: Dataset's pictures are taken from https://www.kaggle.com/..

The End