Installation

```
# Installing DeepFace
!pip install deepface

Collecting deepface
  Downloading deepface-0.0.68-py3-none-any.whl (61 kB)
ent already satisfied: Pillow>=5.2.0 in /usr/local/lib/python3.7/dist-
packages (from deepface) (7.1.2)
Requirement already satisfied: tensorflow>=1.9.0 in
/usr/local/lib/python3.7/dist-packages (from deepface) (2.6.0)
Collecting mtcnn>=0.1.0
  Downloading mtcnn-0.1.1-py3-none-any.whl (2.3 MB)
ent already satisfied: tqdm>=4.30.0 in /usr/local/lib/python3.7/dist-
packages (from deepface) (4.62.0)
Requirement already satisfied: keras>=2.2.0 in
/usr/local/lib/python3.7/dist-packages (from deepface) (2.6.0)
Collecting retina-face>=0.0.1
  Downloading retina_face-0.0.5-py3-none-any.whl (14 kB)
Requirement already satisfied: Flask>=1.1.2 in
/usr/local/lib/python3.7/dist-packages (from deepface) (1.1.4)
Requirement already satisfied: pandas>=0.23.4 in
/usr/local/lib/python3.7/dist-packages (from deepface) (1.1.5)
Collecting gdown>=3.10.1
  Downloading gdown-3.13.1.tar.gz (10 kB)
  Installing build dependencies ... ents to build wheel ...
etadata ... ent already satisfied: opencv-python>=3.4.4 in
/usr/local/lib/python3.7/dist-packages (from deepface) (4.1.2.30)
Requirement already satisfied: numpy>=1.14.0 in
/usr/local/lib/python3.7/dist-packages (from deepface) (1.19.5)
Requirement already satisfied: click<8.0,>=5.1 in
/usr/local/lib/python3.7/dist-packages (from Flask>=1.1.2->deepface)
(7.1.2)
Requirement already satisfied: itsdangerous<2.0,>=0.24 in
/usr/local/lib/python3.7/dist-packages (from Flask>=1.1.2->deepface)
(1.1.0)
Requirement already satisfied: Werkzeug<2.0,>=0.15 in
/usr/local/lib/python3.7/dist-packages (from Flask>=1.1.2->deepface)
(1.0.1)
Requirement already satisfied: Jinja2<3.0,>=2.10.1 in
/usr/local/lib/python3.7/dist-packages (from Flask>=1.1.2->deepface)
(2.11.3)
Requirement already satisfied: filelock in
/usr/local/lib/python3.7/dist-packages (from gdown>=3.10.1->deepface)
(3.0.12)
Requirement already satisfied: requests[socks]>=2.12.0 in
/usr/local/lib/python3.7/dist-packages (from gdown>=3.10.1->deepface)
```

```
(2.23.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-
packages (from gdown>=3.10.1->deepface) (1.15.0)
Requirement already satisfied: MarkupSafe>=0.23 in
/usr/local/lib/python3.7/dist-packages (from Jinja2<3.0,>=2.10.1-
>Flask>=1.1.2->deepface) (2.0.1)
Requirement already satisfied: python-dateutil>=2.7.3 in
/usr/local/lib/python3.7/dist-packages (from pandas>=0.23.4->deepface)
(2.8.2)
Requirement already satisfied: pytz>=2017.2 in
/usr/local/lib/python3.7/dist-packages (from pandas>=0.23.4->deepface)
(2018.9)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests[socks]>=2.12.0-
>gdown>=3.10.1->deepface) (2021.5.30)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
in /usr/local/lib/python3.7/dist-packages (from
requests[socks]>=2.12.0->gdown>=3.10.1->deepface) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in
/usr/local/lib/python3.7/dist-packages (from requests[socks]>=2.12.0-
>gdown>=3.10.1->deepface) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests[socks]>=2.12.0-
>gdown>=3.10.1->deepface) (3.0.4)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in
/usr/local/lib/python3.7/dist-packages (from requests[socks]>=2.12.0-
>gdown>=3.10.1->deepface) (1.7.1)
Requirement already satisfied: termcolor~=1.1.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=1.9.0-
>deepface) (1.1.0)
Requirement already satisfied: typing-extensions~=3.7.4 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=1.9.0-
>deepface) (3.7.4.3)
Requirement already satisfied: wheel~=0.35 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=1.9.0-
>deepface) (0.37.0)
Requirement already satisfied: opt-einsum~=3.3.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=1.9.0-
>deepface) (3.3.0)
Requirement already satisfied: gast==0.4.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=1.9.0-
>deepface) (0.4.0)
Requirement already satisfied: tensorboard~=2.6 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=1.9.0-
>deepface) (2.6.0)
Requirement already satisfied: astunparse~=1.6.3 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=1.9.0-
>deepface) (1.6.3)
Requirement already satisfied: absl-py~=0.10 in
```

```
/usr/local/lib/python3.7/dist-packages (from tensorflow>=1.9.0-
>deepface) (0.12.0)
Requirement already satisfied: clang~=5.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=1.9.0-
>deepface) (5.0)
Requirement already satisfied: wrapt~=1.12.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=1.9.0-
>deepface) (1.12.1)
Requirement already satisfied: keras-preprocessing~=1.1.2 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=1.9.0-
>deepface) (1.1.2)
Requirement already satisfied: grpcio<2.0,>=1.37.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=1.9.0-
>deepface) (1.39.0)
Requirement already satisfied: tensorflow-estimator~=2.6 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=1.9.0-
>deepface) (2.6.0)
Requirement already satisfied: google-pasta~=0.2 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=1.9.0-
>deepface) (0.2.0)
Requirement already satisfied: h5py~=3.1.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=1.9.0-
>deepface) (3.1.0)
Requirement already satisfied: flatbuffers~=1.12.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=1.9.0-
>deepface) (1.12)
Requirement already satisfied: protobuf>=3.9.2 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=1.9.0-
>deepface) (3.17.3)
Requirement already satisfied: cached-property in
/usr/local/lib/python3.7/dist-packages (from h5py~=3.1.0-
>tensorflow>=1.9.0->deepface) (1.5.2)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/usr/local/lib/python3.7/dist-packages (from tensorboard~=2.6-
>tensorflow>=1.9.0->deepface) (0.4.5)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.7/dist-packages (from tensorboard~=2.6-
>tensorflow>=1.9.0->deepface) (3.3.4)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0
in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.6-
>tensorflow>=1.9.0->deepface) (0.6.1)
Requirement already satisfied: google-auth<2,>=1.6.3 in
/usr/local/lib/python3.7/dist-packages (from tensorboard~=2.6-
>tensorflow>=1.9.0->deepface) (1.34.0)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from tensorboard~=2.6-
>tensorflow>=1.9.0->deepface) (1.8.0)
Requirement already satisfied: setuptools>=41.0.0 in
/usr/local/lib/python3.7/dist-packages (from tensorboard~=2.6-
>tensorflow>=1.9.0->deepface) (57.4.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3-
>tensorboard~=2.6->tensorflow>=1.9.0->deepface) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in
/usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3-
>tensorboard~=2.6->tensorflow>=1.9.0->deepface) (4.7.2)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3-
>tensorboard~=2.6->tensorflow>=1.9.0->deepface) (4.2.2)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard~=2.6->tensorflow>=1.9.0->deepface)
(1.3.0)
Requirement already satisfied: importlib-metadata in
/usr/local/lib/python3.7/dist-packages (from markdown>=2.6.8-
>tensorboard~=2.6->tensorflow>=1.9.0->deepface) (4.6.4)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=0.2.1-
>google-auth<2,>=1.6.3->tensorboard~=2.6->tensorflow>=1.9.0->deepface)
(0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.7/dist-packages (from requests-oauthlib>=0.7.0-
>google-auth-oauthlib<0.5,>=0.4.1->tensorboard~=2.6-
>tensorflow>=1.9.0->deepface) (3.1.1)
Requirement already satisfied: zipp>=0.5 in
/usr/local/lib/python3.7/dist-packages (from importlib-metadata-
>markdown>=2.6.8->tensorboard~=2.6->tensorflow>=1.9.0->deepface)
(3.5.0)
Building wheels for collected packages: gdown
  Building wheel for gdown (PEP 517) ... e=gdown-3.13.1-py3-none-
any.whl size=9920
sha256=9328d22f889f29fa1ddfd4963ced0695650a242fc72e77bfa3bb90295e81ac4
e
  Stored in directory:
/root/.cache/pip/wheels/f2/8d/0b/2e7e6c725f898bd7ef654b660528e459a4d79
f3a68976ca9fc
Successfully built gdown
Installing collected packages: gdown, retina-face, mtcnn, deepface
  Attempting uninstall: gdown
    Found existing installation: gdown 3.6.4
    Uninstalling gdown-3.6.4:
      Successfully uninstalled gdown-3.6.4
Successfully installed deepface-0.0.68 gdown-3.13.1 mtcnn-0.1.1
retina-face-0.0.5
```

2.2 To Load test data

Since its a pre-trained model we will only test it with some random image.

```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

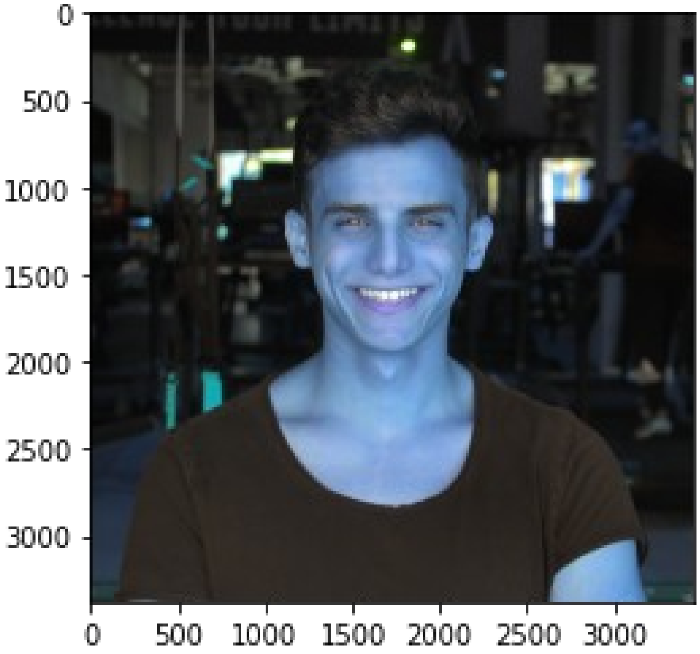
```python
image = "/content/drive/MyDrive/Capstone Project 5/PreTRained Model
Test Images/faces-of-smiling-happy-children-in-makueni-county-kenya-
2C8A5PJ.jpg"
```

```python
# Image Show
import cv2
import matplotlib.pyplot as plt
img_array=cv2.imread(image)
plt.imshow(img_array)
```

```
<matplotlib.image.AxesImage at 0x7efcc8d5fd90>
```

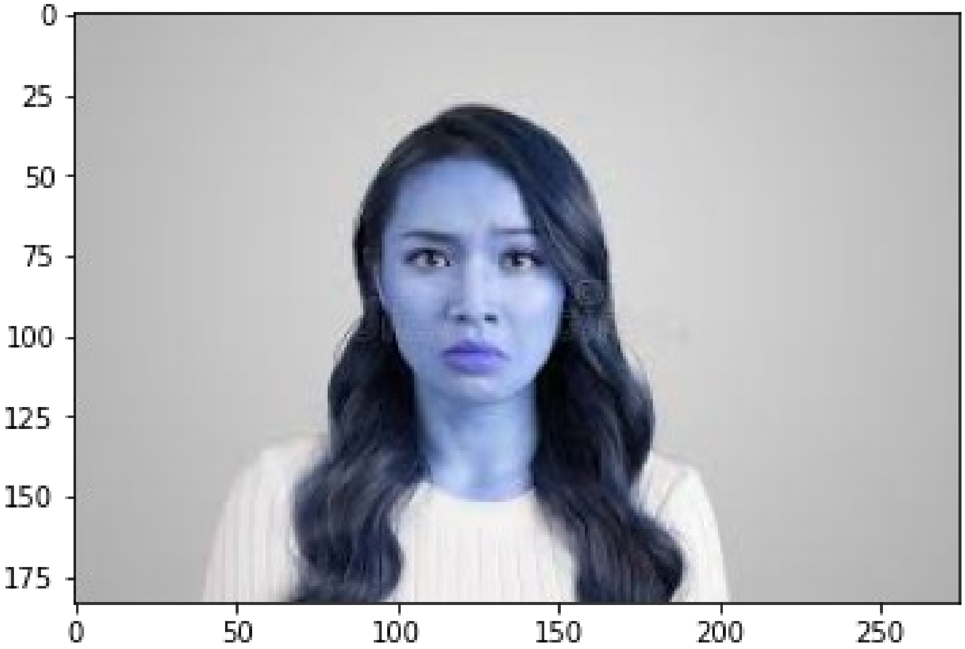
```python
image1 = '/content/drive/MyDrive/Capstone Project 5/PreTRained Model
Test Images/abdullah-ali-1w9I6H4aftw-unsplash.jpg'
img_array1=cv2.imread(image1)
plt.imshow(img_array1)
```

```
<matplotlib.image.AxesImage at 0x7efcc8add050>
```



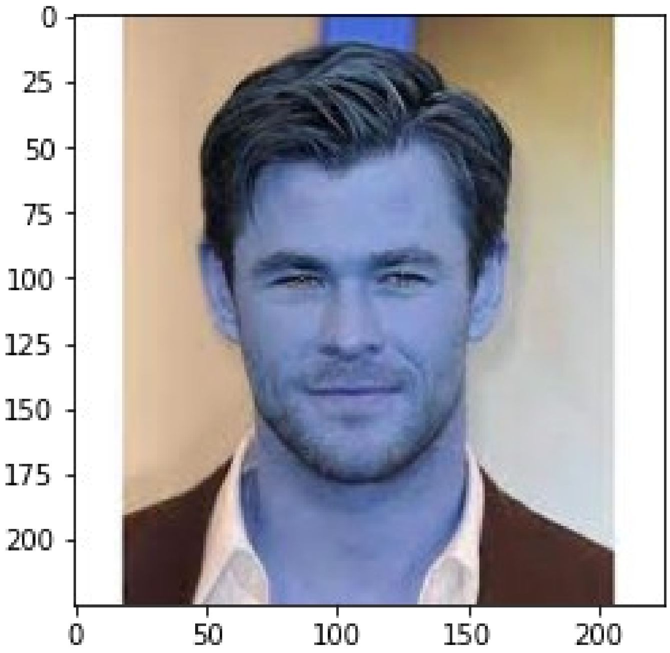
```
image2 = '/content/drive/MyDrive/Capstone Project 5/PreTRained Model
Test Images/sad face.jpeg'
img_array2=cv2.imread(image2)
plt.imshow(img_array2)
```

```
<matplotlib.image.AxesImage at 0x7efcc99f0fd0>
```



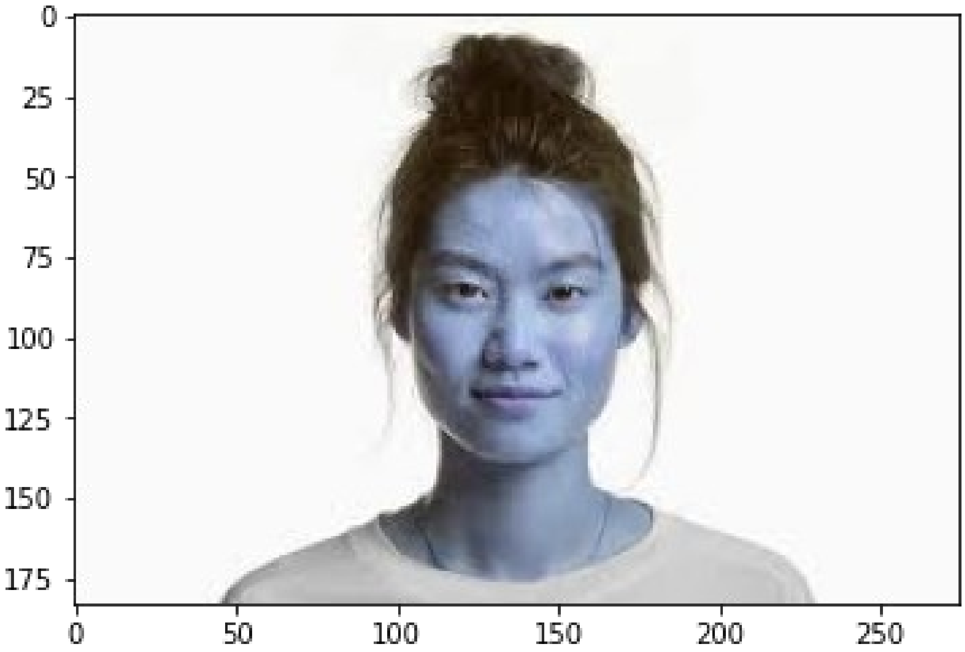
```
image3 = '/content/drive/MyDrive/Capstone Project 5/PreTRained Model
Test Images/image 3.jpeg'
img_array3=cv2.imread(image3)
plt.imshow(img_array3)
```

```
<matplotlib.image.AxesImage at 0x7efcc99d0790>
```



```
image4 = '/content/drive/MyDrive/Capstone Project 5/PreTRained Model
Test Images/image 4.jpeg'
img_array4=cv2.imread(image4)
plt.imshow(img_array4)
```

```
<matplotlib.image.AxesImage at 0x7efcc8d08510>
```



2.3 Analyze Data

```
obj1 = DeepFace.analyze(img_path = image1, actions = ['age', 'gender',
'race', 'emotion'])
print("Result for image1", obj1["age"]," years old
",obj1["dominant_race"]," ",obj1["dominant_emotion"]," ",
obj1["gender"])

Action: emotion: 100%|██████████| 4/4 [00:06<00:00,  1.74s/it]

Result for image1 28  years old  white   happy   Man



obj2 = DeepFace.analyze(img_path = image2, actions = ['age', 'gender',
'race', 'emotion'])
print("Result for image2", obj2["age"]," years old
",obj2["dominant_race"]," ",obj2["dominant_emotion"]," ",
obj2["gender"])

Action: emotion: 100%|██████████| 4/4 [00:01<00:00,  2.29it/s]

Result for image2 31  years old  latino hispanic   sad   Woman



obj3 = DeepFace.analyze(img_path = image3, actions = ['age', 'gender',
'race', 'emotion'])
print("Result for image3", obj3["age"]," years old
",obj3["dominant_race"]," ",obj3["dominant_emotion"]," ",
obj3["gender"])
```

```
Action: emotion: 100%|██████████| 4/4 [00:01<00:00,  2.24it/s]

Result for image3 23  years old  white    neutral    Man



obj4 = DeepFace.analyze(img_path = image4, actions = ['age', 'gender',
'race', 'emotion'])
print("Result for image4", obj4["age"]," years old
",obj4["dominant_race"]," ",obj4["dominant_emotion"]," ",
obj4["gender"])

Action: emotion: 100%|██████████| 4/4 [00:01<00:00,  2.27it/s]

Result for image4 24  years old  asian    neutral    Man
```

3.Building Face Emotion Recognition Model

Link Google Drive

```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

Import Necessary Libraries

```
%matplotlib inline
import matplotlib.pyplot as plt

import numpy as np
import pandas as pd
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

from keras.models import Sequential #Initialise our neural network
model as a sequential network
from keras.layers import Conv2D #Convolution operation
from keras.layers import BatchNormalization
from keras.regularizers import l2
from keras.layers import Activation#Applies activation function
from keras.layers import Dropout#Prevents overfitting by randomly
converting few outputs to zero
from keras.layers import MaxPooling2D # Maxpooling function
from keras.layers import Flatten # Converting 2D arrays into a 1D
linear vector
```

```
from keras.layers import Dense # Regular fully connected neural
network
from tensorflow.keras import optimizers
from keras.callbacks import ReduceLROnPlateau, EarlyStopping,
TensorBoard, ModelCheckpoint
from sklearn.metrics import accuracy_score
```

##3.3 Load Dataset

```
def load_data(dataset_path): #Run once

  #classes = ['Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprsie',
'Neutral']  #We will be dealing with seven different types of
emotions.

  data = []
  test_data = []
  test_labels = []
  labels =[]

  with open(dataset_path, 'r') as file:
      for line_no, line in enumerate(file.readlines()):
          if 0 < line_no <= 35887:
            curr_class, line, set_type = line.split(',')
            image_data = np.asarray([int(x) for x in
line.split()]).reshape(48, 48)#Creating a list out of the string then
converting it into a 2-Dimensional numpy array.
            image_data =image_data.astype(np.uint8)/255.0

            if (set_type.strip() == 'PrivateTest'):

              test_data.append(image_data)
              test_labels.append(curr_class)
            else:
              data.append(image_data)
              labels.append(curr_class)

      test_data = np.expand_dims(test_data, -1)
      test_labels = to_categorical(test_labels, num_classes = 7)
      data = np.expand_dims(data, -1)
      labels = to_categorical(labels, num_classes = 7)

      return np.array(data), np.array(labels), np.array(test_data),
np.array(test_labels)
```

Splitting of Data

```
dataset_path = "/content/drive/MyDrive/Capstone Project 5/fer2013.csv"
```

```
train_data, train_labels, test_data, test_labels =
load_data(dataset_path)

print("Number of images in Training set:", len(train_data))
print("Number of images in Test set:", len(test_data))
```

```
Number of images in Training set: 32298
Number of images in Test set: 3589
```

Model Training

```
######HYPERPARAMATERS###########
epochs = 75
batch_size = 64
learning_rate = 0.001
#################################

model = Sequential()

model.add(Conv2D(64, (3, 3), activation='relu', input_shape=(48, 48,
1), kernel_regularizer=l2(0.01)))
model.add(Conv2D(64, (3, 3), padding='same',activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2), strides=(2, 2)))
model.add(Dropout(0.5))

model.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.5))

model.add(Conv2D(256, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(256, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(256, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.5))

model.add(Conv2D(512, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(512, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(512, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.5))

model.add(Flatten())

model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(7, activation='softmax'))

adam = optimizers.Adam(lr = learning_rate)
model.compile(optimizer = adam, loss = 'categorical_crossentropy',
metrics = ['accuracy'])

print(model.summary())

lr_reducer = ReduceLROnPlateau(monitor='val_loss', factor=0.9,
patience=3)
early_stopper = EarlyStopping(monitor='val_acc', min_delta=0,
patience=6, mode='auto')
checkpointer = ModelCheckpoint('/content/gdrive/My Drive/Colab
Notebooks/Emotion Recognition/Model/weights.hd5', monitor='val_loss',
verbose=1, save_best_only=True)

history = model.fit(
          train_data,
          train_labels,
          epochs = epochs,
          batch_size = batch_size,
          validation_split = 0.2,
          shuffle = True
          )
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/
optimizer_v2.py:356: UserWarning: The `lr` argument is deprecated, use
`learning_rate` instead.
  "The `lr` argument is deprecated, use `learning_rate` instead.")

Model: "sequential_1"
_________________________________________________________________
```

```
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_11 (Conv2D)           (None, 46, 46, 64)        640       
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 46, 46, 64)        36928     
_________________________________________________________________
batch_normalization_10 (Batc (None, 46, 46, 64)        256       
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 23, 23, 64)        0         
_________________________________________________________________
dropout_8 (Dropout)          (None, 23, 23, 64)        0         
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 23, 23, 128)       73856     
_________________________________________________________________
batch_normalization_11 (Batc (None, 23, 23, 128)       512       
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 23, 23, 128)       147584    
_________________________________________________________________
batch_normalization_12 (Batc (None, 23, 23, 128)       512       
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 23, 23, 128)       147584    
_________________________________________________________________
batch_normalization_13 (Batc (None, 23, 23, 128)       512       
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 11, 11, 128)       0         
_________________________________________________________________
dropout_9 (Dropout)          (None, 11, 11, 128)       0         
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 11, 11, 256)       295168    
_________________________________________________________________
batch_normalization_14 (Batc (None, 11, 11, 256)       1024      
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 11, 11, 256)       590080    
_________________________________________________________________
batch_normalization_15 (Batc (None, 11, 11, 256)       1024      
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 11, 11, 256)       590080    
_________________________________________________________________
batch_normalization_16 (Batc (None, 11, 11, 256)       1024      
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 5, 5, 256)         0         
_________________________________________________________________
dropout_10 (Dropout)         (None, 5, 5, 256)         0         
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 5, 5, 512)         1180160   
_________________________________________________________________
batch_normalization_17 (Batc (None, 5, 5, 512)         2048      
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 5, 5, 512)         2359808   
```

```
_________________________________________________________________
batch_normalization_18 (Batc (None, 5, 5, 512)         2048
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 5, 5, 512)         2359808
_________________________________________________________________
batch_normalization_19 (Batc (None, 5, 5, 512)         2048
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 2, 2, 512)         0
_________________________________________________________________
dropout_11 (Dropout)         (None, 2, 2, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_5 (Dense)              (None, 512)               1049088
_________________________________________________________________
dropout_12 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_6 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_13 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_7 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_14 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_8 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_15 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_9 (Dense)              (None, 7)                 455
=================================================================
Total params: 9,014,727
Trainable params: 9,009,223
Non-trainable params: 5,504
_________________________________________________________________
None
Epoch 1/75
404/404 [==============================] - 59s 141ms/step - loss:
2.0908 - accuracy: 0.2009 - val_loss: 1.8732 - val_accuracy: 0.2489
Epoch 2/75
404/404 [==============================] - 57s 140ms/step - loss:
1.8643 - accuracy: 0.2362 - val_loss: 1.8446 - val_accuracy: 0.2489
Epoch 3/75
404/404 [==============================] - 56s 140ms/step - loss:
1.8404 - accuracy: 0.2485 - val_loss: 1.8268 - val_accuracy: 0.2489
Epoch 4/75
404/404 [==============================] - 57s 140ms/step - loss:
1.8323 - accuracy: 0.2509 - val_loss: 1.8246 - val_accuracy: 0.2489
Epoch 5/75
```

```
404/404 [==============================] - 57s 140ms/step - loss:
1.8244 - accuracy: 0.2506 - val_loss: 1.8210 - val_accuracy: 0.2489
Epoch 6/75
298/404 [=====================>........] - ETA: 13s - loss: 1.8191 -
accuracy: 0.2553
```

3.6 Evaluation Metrics

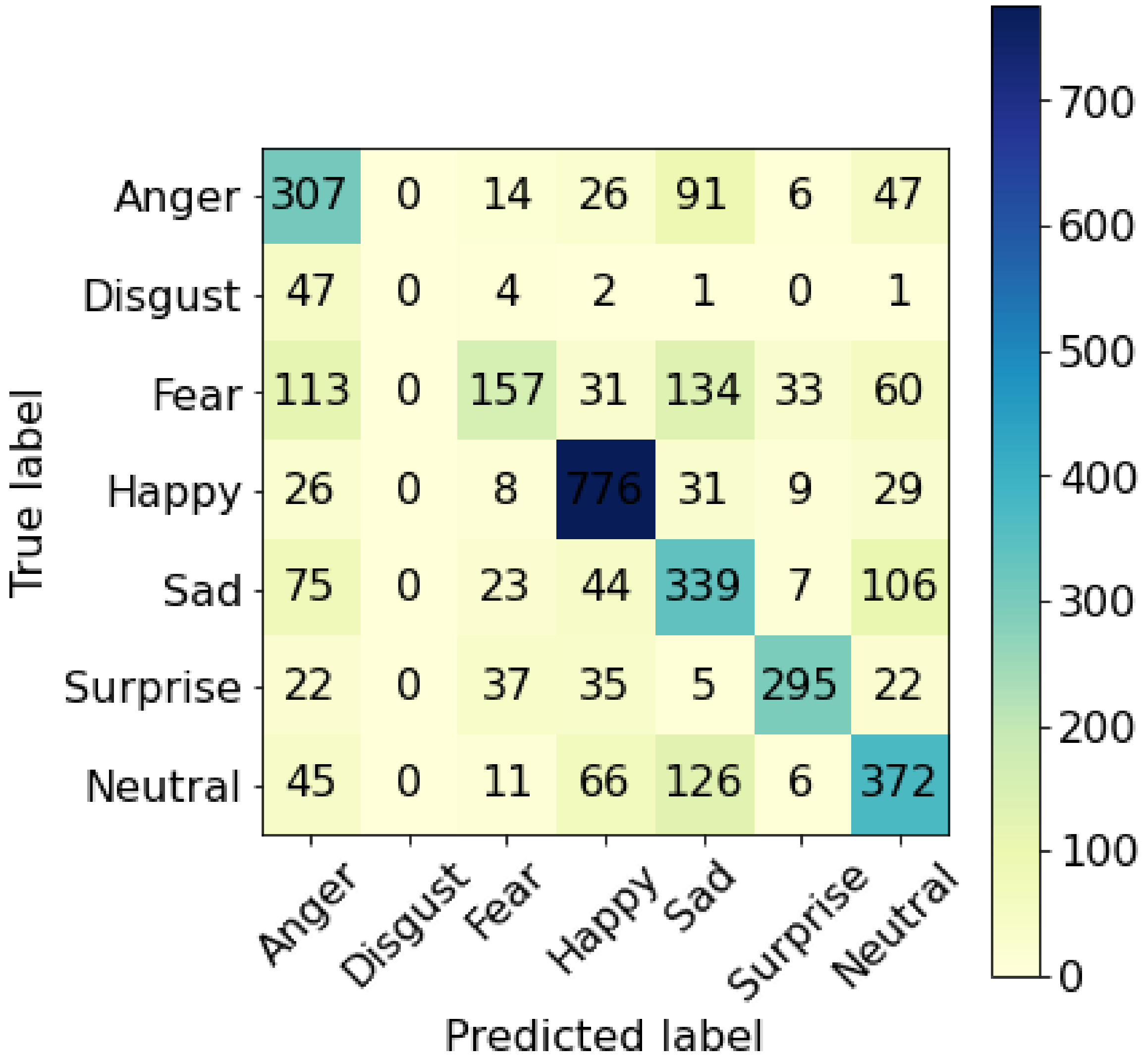
To Check Accuracy

```
predicted_test_labels = np.argmax(model.predict(test_data), axis=1)
test_labels = np.argmax(test_labels, axis=1)
print ("Accuracy score = ", accuracy_score(test_labels,
predicted_test_labels))
print ("Accuracy percentage = ", accuracy_score(test_labels,
predicted_test_labels)*100, "%")
```

```
Accuracy score =  0.6258010587907495
Accuracy percentage =  62.58010587907496 %
```

Confusion Matrix

```
labels = ['Anger', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise',
'Neutral']

def plot_confusion_matrix(y_true, y_pred, cmap=plt.cm.Blues):
    cm = confusion_matrix(y_true, y_pred)
    fig = plt.figure(figsize=(6,6))
    plt.rcParams.update({'font.size': 16})
    ax  = fig.add_subplot(111)
    matrix = ax.imshow(cm, interpolation='nearest', cmap=cmap)
    fig.colorbar(matrix)
    for i in range(0,7):
        for j in range(0,7):
            ax.text(j,i,cm[i,j],va='center', ha='center')
    # ax.set_title('Confusion Matrix')
    ticks = np.arange(len(labels))
    ax.set_xticks(ticks)
    ax.set_xticklabels(labels, rotation=45)
    ax.set_yticks(ticks)
    ax.set_yticklabels(labels)
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

plot_confusion_matrix(test_labels, predicted_test_labels,
cmap=plt.cm.YlGnBu)
plt.show()
```



Classification report

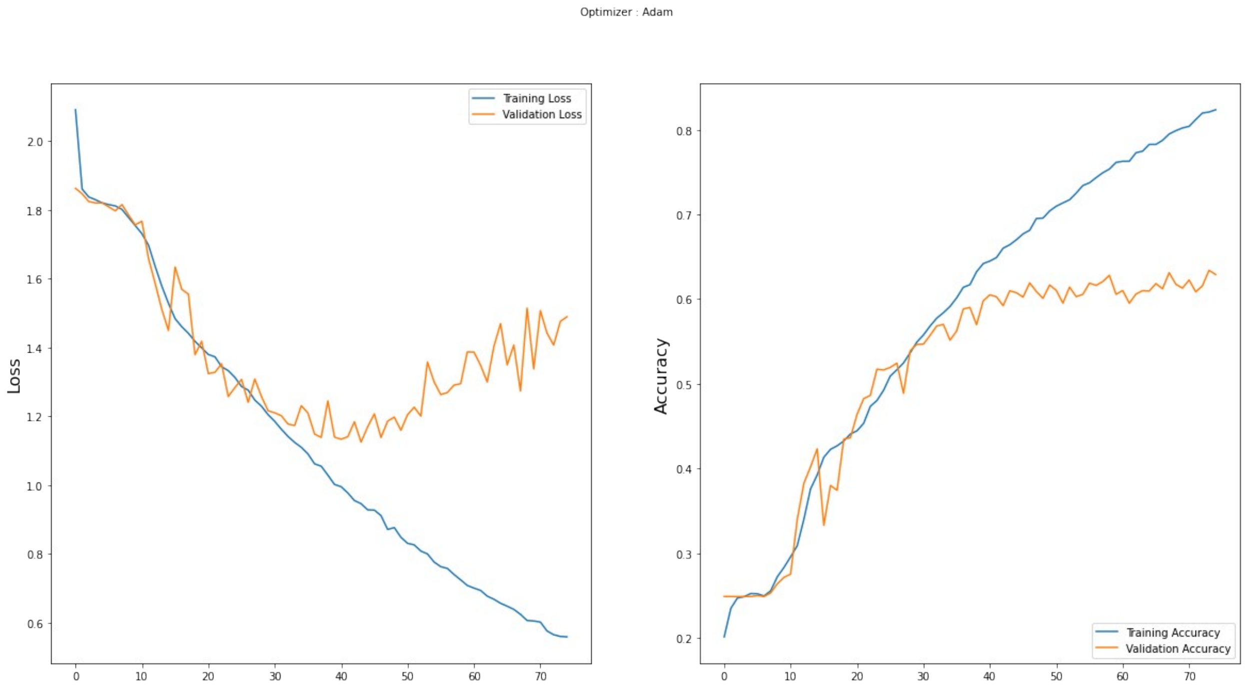
```
from sklearn.metrics import classification_report
print(classification_report(test_labels, predicted_test_labels,
target_names=labels))
```

```
              precision    recall  f1-score   support

       Anger       0.48      0.63      0.55       491
     Disgust       0.00      0.00      0.00        55
        Fear       0.62      0.30      0.40       528
       Happy       0.79      0.88      0.83       879
         Sad       0.47      0.57      0.51       594
    Surprise       0.83      0.71      0.76       416
     Neutral       0.58      0.59      0.59       626

    accuracy                           0.63      3589
```

```
   macro avg       0.54      0.53      0.52      3589
weighted avg       0.63      0.63      0.61      3589


/usr/local/lib/python3.7/dist-packages/sklearn/metrics/
_classification.py:1272: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Loss & Accuracy Graph

```
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : Adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```



Saving of Model

```
from keras.models import model_from_json
model_json = model.to_json()
with open("/content/drive/MyDrive/Capstone Project 5/model.json", "w")
as json_file:
    json_file.write(model_json)

model.save_weights("/content/drive/MyDrive/Capstone Project
5/model.hdf5")
print("Saved model to disk")
```

```
Saved model to disk
```

```
model.save('/content/drive/MyDrive/Capstone Project 5/FER_model.h5')
print('Model Saved')
```

```
Model Saved
```

```
!pip install keras
```

```
Requirement already satisfied: keras in /usr/local/lib/python3.7/dist-
packages (2.6.0)
```

```
import keras

saveBestModel = keras.callbacks.ModelCheckpoint('/best_model.hdf5',
monitor='val_acc', verbose=0, save_best_only=True,
save_weights_only=False, mode='auto', period=1,save_freq='epoch')
```

```
WARNING:tensorflow:`period` argument is deprecated. Please use
`save_freq` to specify the frequency in number of batches seen.
```

4.To Check our Model on Images

Preparation for emotion recognition using photo

```
def emotion_analysis(emotions):
    objects = ('angry', 'disgust', 'fear', 'happy', 'sad', 'surprise',
'neutral')
    y_pos = np.arange(len(objects))

    plt.bar(y_pos, emotions, align='center', alpha=0.5)
    plt.xticks(y_pos, objects)
    plt.ylabel('percentage')
    plt.title('emotion')

    plt.show()
```

Model Loading

```
from keras.models import model_from_json
import numpy as np
import cv2

def load_model(path):

      json_file = open(path + 'model.json', 'r')
      loaded_model_json = json_file.read()
      json_file.close()

      model = model_from_json(loaded_model_json)
      model.load_weights(path + "model.h5")
      print("Loaded model from disk")
      return model

#model loading
path = "/content/drive/MyDrive/Capstone Project 5/"
model = load_model(path)
```

```
Loaded model from disk
```

Defining function which will click images

```
##CODE for Capturing an image on Colab from here:
https://colab.research.google.com/notebook#fileId=1OnUy6eFE7XhdfGfAHDC
qQxpwueTOj_NO
```

```
from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode

def take_photo(filename='photo.jpg', quality=0.8):
  js = Javascript('''
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({video: true});

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();

      // Resize the output to fit the video element.
      google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

      // Wait for Capture to be clicked.
      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
      canvas.getContext('2d').drawImage(video, 0, 0);
      stream.getVideoTracks()[0].stop();
      div.remove();
      return canvas.toDataURL('image/jpeg', quality);
    }
    ''')
  display(js)
  data = eval_js('takePhoto({})'.format(quality))
  binary = b64decode(data.split(',')[1])
  with open(filename, 'wb') as f:
    f.write(binary)
  return filename
```

Use this to click *photo*

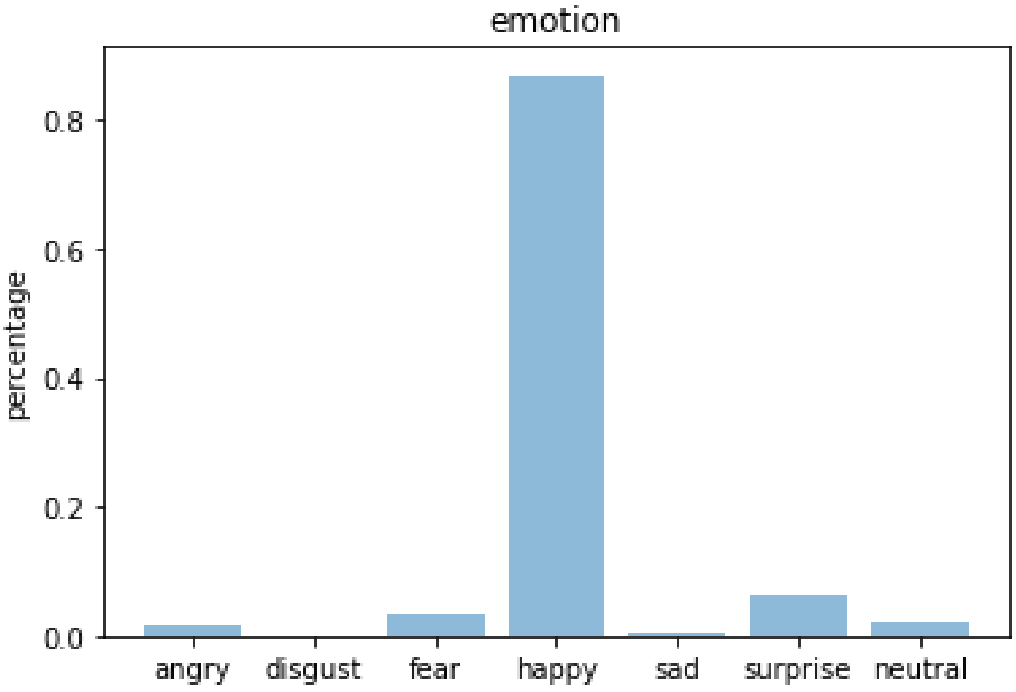
```
take_photo()
```

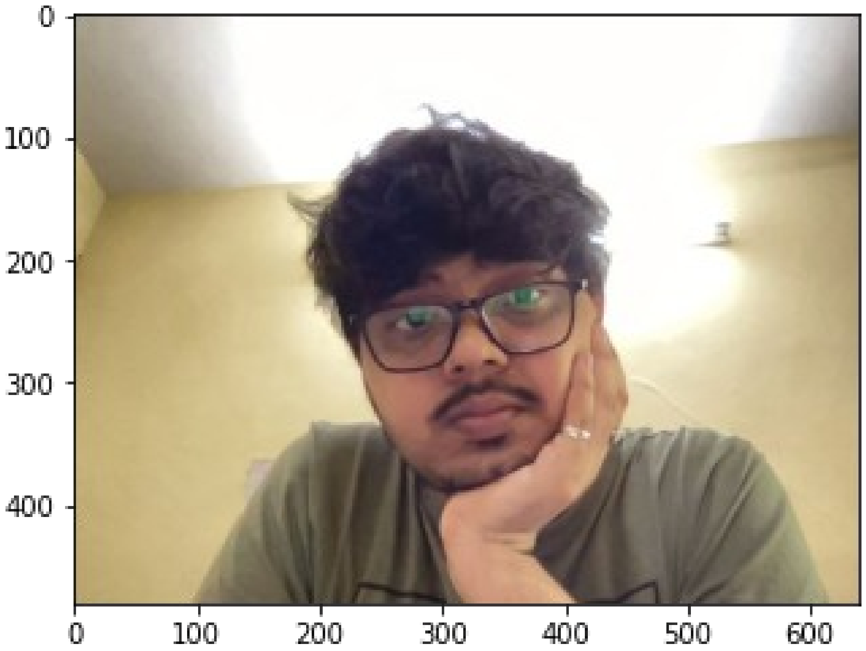
```
<IPython.core.display.Javascript object>

{"type":"string"}
```

Defining Function to crop face and analyszing photo taken

```python
import cv2

def facecrop(image):
    facedata = '/content/haarcascade_frontalface_alt.xml'
    cascade = cv2.CascadeClassifier(facedata)

    img = cv2.imread(image)

    try:

        minisize = (img.shape[1],img.shape[0])
        miniframe = cv2.resize(img, minisize)

        faces = cascade.detectMultiScale(miniframe)

        for f in faces:
            x, y, w, h = [ v for v in f ]
            cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)

            sub_face = img[y:y+h, x:x+w]


            cv2.imwrite('capture.jpg', sub_face)


    except Exception as e:
        print (e)




if __name__ == '__main__':
    facecrop('/content/photo.jpg')


from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator

import numpy as np
import matplotlib.pyplot as plt


file = '/content/photo.jpg'
true_image = image.load_img(file)
img = image.load_img(file, color_mode="grayscale", target_size=(48,
48))

x = image.img_to_array(img)
x = np.expand_dims(x, axis = 0)

x /= 255

custom = model.predict(x)
emotion_analysis(custom[0])

x = np.array(x, 'float32')
x = x.reshape([48, 48]);


plt.imshow(true_image)
plt.show()
```

```
OpenCV(4.1.2) /io/opencv/modules/objdetect/src/cascadedetect.cpp:1689:
error: (-215:Assertion failed) !empty() in function 'detectMultiScale'
```





#5.To run Webcam

```
!wget --no-check-certificate \
    https://raw.githubusercontent.com/computationalcore/introduction-to-opencv/master/assets/haarcascade_frontalface_default.xml \
    -O haarcascade_frontalface_default.xml



from IPython.display import display, Javascript, Image
from google.colab.output import eval_js
from base64 import b64decode, b64encode
import cv2
import numpy as np
import PIL
import io
import html
import time



def js_to_image(js_reply):
  """
  Params:
          js_reply: JavaScript object containing image from webcam
  Returns:
          img: OpenCV BGR image
  """
  image_bytes = b64decode(js_reply.split(',')[1])

  jpg_as_np = np.frombuffer(image_bytes, dtype=np.uint8)

  img = cv2.imdecode(jpg_as_np, flags=1)

  return img


def bbox_to_bytes(bbox_array):
  """
  Params:
          bbox_array: Numpy array (pixels) containing rectangle to
overlay on video stream.
  Returns:
        bytes: Base64 image byte string
  """

  bbox_PIL = PIL.Image.fromarray(bbox_array, 'RGBA')
  iobuf = io.BytesIO()

  bbox_PIL.save(iobuf, format='png')

  bbox_bytes = 'data:image/png;base64,
{}'.format((str(b64encode(iobuf.getvalue()), 'utf-8')))

  return bbox_bytes




face_cascade =
cv2.CascadeClassifier(cv2.samples.findFile(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml'))



def video_stream():
  js = Javascript('''
    var video;
    var div = null;
    var stream;
    var captureCanvas;
    var imgElement;
    var labelElement;

    var pendingResolve = null;
    var shutdown = false;
    function removeDom() {
       stream.getVideoTracks()[0].stop();
       video.remove();
       div.remove();
       video = null;
       div = null;
       stream = null;
       imgElement = null;
       captureCanvas = null;
       labelElement = null;
    }

    function onAnimationFrame() {
      if (!shutdown) {
        window.requestAnimationFrame(onAnimationFrame);
      }
      if (pendingResolve) {
        var result = "";
        if (!shutdown) {
          captureCanvas.getContext('2d').drawImage(video, 0, 0, 640,
480);
          result = captureCanvas.toDataURL('image/jpeg', 0.8)
        }
        var lp = pendingResolve;
        pendingResolve = null;
        lp(result);
      }
    }

    async function createDom() {
      if (div !== null) {
        return stream;
      }

      div = document.createElement('div');
      div.style.border = '2px solid black';
      div.style.padding = '3px';
      div.style.width = '100%';
      div.style.maxWidth = '600px';
      document.body.appendChild(div);

      const modelOut = document.createElement('div');
      modelOut.innerHTML = "<span>Status:</span>";
      labelElement = document.createElement('span');
      labelElement.innerText = 'No data';
      labelElement.style.fontWeight = 'bold';
      modelOut.appendChild(labelElement);
      div.appendChild(modelOut);
      video = document.createElement('video');
      video.style.display = 'block';
      video.width = div.clientWidth - 6;
      video.setAttribute('playsinline', '');
      video.onclick = () => { shutdown = true; };
      stream = await navigator.mediaDevices.getUserMedia(
          {video: { facingMode: "environment"}});
      div.appendChild(video);

      imgElement = document.createElement('img');
      imgElement.style.position = 'absolute';
      imgElement.style.zIndex = 1;
      imgElement.onclick = () => { shutdown = true; };
      div.appendChild(imgElement);

      const instruction = document.createElement('div');
      instruction.innerHTML =
          '<span style="color: red; font-weight: bold;">' +
          'When finished, click here or on the video to stop this
demo</span>';
      div.appendChild(instruction);
      instruction.onclick = () => { shutdown = true; };

      video.srcObject = stream;
      await video.play();

      captureCanvas = document.createElement('canvas');
      captureCanvas.width = 640; //video.videoWidth;
      captureCanvas.height = 480; //video.videoHeight;
      window.requestAnimationFrame(onAnimationFrame);

      return stream;
    }
    async function stream_frame(label, imgData) {
      if (shutdown) {
        removeDom();
        shutdown = false;
        return '';
      }

      var preCreate = Date.now();
      stream = await createDom();

      var preShow = Date.now();
      if (label != "") {
        labelElement.innerHTML = label;
      }

      if (imgData != "") {
        var videoRect = video.getClientRects()[0];
        imgElement.style.top = videoRect.top + "px";
        imgElement.style.left = videoRect.left + "px";
        imgElement.style.width = videoRect.width + "px";
        imgElement.style.height = videoRect.height + "px";
        imgElement.src = imgData;
      }

      var preCapture = Date.now();
      var result = await new Promise(function(resolve, reject) {
        pendingResolve = resolve;
      });
      shutdown = false;

      return {'create': preShow - preCreate,
              'show': preCapture - preShow,
              'capture': Date.now() - preCapture,
              'img': result};
    }
    ''')

  display(js)

def video_frame(label, bbox):
  data = eval_js('stream_frame("{}", "{}")'.format(label, bbox))
  return data

colour_cycle = ((255, 0, 0), (0, 255, 0), (0, 0, 255), (230, 230,
250))
```

5.1 To load our model

```
from keras.models import model_from_json
import numpy as np
import cv2

def load_model(path):

      json_file = open(path + 'model.json', 'r')
      loaded_model_json = json_file.read()
      json_file.close()

      model = model_from_json(loaded_model_json)
      model.load_weights(path + "model.h5")
      print("Loaded model from disk")
      return model

def predict_emotion(gray, x, y, w, h):
      face = np.expand_dims(np.expand_dims(np.resize(gray[y:y+w,
x:x+h]/255.0, (48, 48)),-1), 0)
```

```
    prediction = model.predict([face])

    return(int(np.argmax(prediction)), round(max(prediction[0])*100,
2))

path = "/content/drive/MyDrive/Capstone Project 5/"
model = load_model(path)

fcc_path = "Tools/haarcascade_frontalface_alt.xml"
faceCascade = cv2.CascadeClassifier(fcc_path)
emotion_dict = {0: "Angry", 1: "Disgust", 2: "Fear", 3: "Happy", 4:
"Sad", 5: "Surprise", 6: "Neutral"}
colour_cycle = ((255, 0, 0), (0, 255, 0), (0, 0, 255), (230, 230,
250))

Loaded model from disk
```

Start VideoStream and live Recognition

```
from google.colab.patches import cv2_imshow
from IPython.display import clear_output


video_stream()

label_html = 'Capturing...'

bbox = ''
count = 0
counter = 1
while True:
    js_reply = video_frame(label_html, bbox)
    if not js_reply:
        break


    img = js_to_image(js_reply["img"])


    bbox_array = np.zeros([480,640,4], dtype=np.uint8)


    gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)


    faces =
face_cascade.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=5,minS
ize=(30, 30))

    for (count,(x, y, w, h)) in enumerate(faces):
      colour = colour_cycle[int(count%len(colour_cycle))]
      bbox_array = cv2.rectangle(bbox_array, (x, y), (x+w, y+h),
colour, 2)
      bbox_array = cv2.line(bbox_array, (x+5, y+h+5),(x+100, y+h+5),
colour, 20)
      bbox_array = cv2.putText(bbox_array, "Face #"+str(count+1),
(x+5, y+h+11), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255),
lineType=cv2.LINE_AA)
      bbox_array = cv2.line(bbox_array, (x+8, y),(x+150, y), colour,
20)
      emotion_id, confidence = predict_emotion(gray, x, y, w, h)
      emotion = emotion_dict[emotion_id]
      bbox_array = cv2.putText(bbox_array, emotion + ": " +
str(confidence) + "%" , (x+20, y+5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
(255, 255, 255), lineType=cv2.LINE_AA)
      bbox_array[:,:,3] = (bbox_array.max(axis = 2) > 0 ).astype(int)
* 255

      bbox_bytes = bbox_to_bytes(bbox_array)

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

cv2.destroyAllWindows()
```