```
from google.colab import drive
drive.mount('/content/gdrive')
!unzip gdrive/MyDrive/FaceRecognition/archive.zip
```

```
Streaming output truncated to the last 5000 lines.
  inflating: MMAFEDB/valid/neutral/8867Exp6fighting_expression_295.jpg
  inflating: MMAFEDB/valid/neutral/8870Exp6fighting_expression_308.jpg
  inflating: MMAFEDB/valid/neutral/8871Exp6fighting_expression_308.jpg
  inflating: MMAFEDB/valid/neutral/8872Exp6fighting_expression_320.jpg
  inflating: MMAFEDB/valid/neutral/8873Exp6fighting_expression_326.jpg
  inflating: MMAFEDB/valid/neutral/8875Exp6fighting_expression_342.jpg
  inflating: MMAFEDB/valid/neutral/8876Exp6fighting_expression_346.jpg
  inflating: MMAFEDB/valid/neutral/8878Exp6fighting_expression_395.jpg
  inflating: MMAFEDB/valid/neutral/8880Exp6fighting_expression_413.jpg
  inflating: MMAFEDB/valid/neutral/8881Exp6fighting_expression_45.jpg
  inflating: MMAFEDB/valid/neutral/8882Exp6fighting_expression_46.jpg
  inflating: MMAFEDB/valid/neutral/8887Exp6fighting_expression_578.jpg
  inflating: MMAFEDB/valid/neutral/8888Exp6fighting_expression_58.jpg
  inflating: MMAFEDB/valid/neutral/8891Exp6fighting_expression_592.jpg
  inflating: MMAFEDB/valid/neutral/8892Exp6fighting_expression_601.jpg
  inflating: MMAFEDB/valid/neutral/8894Exp6fighting_expression_623.jpg
  inflating: MMAFEDB/valid/neutral/8895Exp6fighting_expression_623.jpg
  inflating: MMAFEDB/valid/neutral/8898Exp6fighting_expression_642.jpg
  inflating: MMAFEDB/valid/neutral/8899Exp6fighting_expression_651.jpg
  inflating: MMAFEDB/valid/neutral/889Exp6angry_european_197.jpg
  inflating: MMAFEDB/valid/neutral/88Exp6angry_actor_611.jpg
  inflating: MMAFEDB/valid/neutral/8900Exp6fighting_expression_667.jpg
  inflating: MMAFEDB/valid/neutral/8901Exp6fighting_expression_679.jpg
  inflating: MMAFEDB/valid/neutral/8904Exp6fighting_expression_712.jpg
  inflating: MMAFEDB/valid/neutral/8905Exp6fighting_expression_716.jpg
  inflating: MMAFEDB/valid/neutral/8906Exp6fighting_expression_716.jpg
  inflating: MMAFEDB/valid/neutral/890Exp6angry_european_197.jpg
  inflating: MMAFEDB/valid/neutral/8910Exp6fighting_expression_752.jpg
  inflating: MMAFEDB/valid/neutral/8911Exp6fighting_expression_752.jpg
  inflating: MMAFEDB/valid/neutral/8912Exp6fighting_expression_762.jpg
  inflating: MMAFEDB/valid/neutral/8913Exp6fighting_expression_762.jpg
  inflating: MMAFEDB/valid/neutral/8915Exp6fighting_expression_782.jpg
  inflating: MMAFEDB/valid/neutral/8917Exp6fighting_expression_807.jpg
  inflating: MMAFEDB/valid/neutral/8918Exp6fighting_expression_818.jpg
  inflating: MMAFEDB/valid/neutral/8919Exp6fighting_expression_827.jpg
  inflating: MMAFEDB/valid/neutral/891Exp6angry_european_197.jpg
  inflating: MMAFEDB/valid/neutral/8920Exp6fighting_expression_869.jpg
  inflating: MMAFEDB/valid/neutral/8922Exp6fighting_expression_90.jpg
  inflating: MMAFEDB/valid/neutral/8923Exp6fighting_face_105.jpg
  inflating: MMAFEDB/valid/neutral/8924Exp6fighting_face_108.jpg
  inflating: MMAFEDB/valid/neutral/8926Exp6fighting_face_128.jpg
  inflating: MMAFEDB/valid/neutral/8928Exp6fighting_face_153.jpg
  inflating: MMAFEDB/valid/neutral/892Exp6angry_european_197.jpg
  inflating: MMAFEDB/valid/neutral/8930Exp6fighting_face_156.jpg
  inflating: MMAFEDB/valid/neutral/8931Exp6fighting_face_162.jpg
  inflating: MMAFEDB/valid/neutral/8932Exp6fighting_face_174.jpg
  inflating: MMAFEDB/valid/neutral/8934Exp6fighting_face_190.jpg
  inflating: MMAFEDB/valid/neutral/8935Exp6fighting_face_197.jpg
  inflating: MMAFEDB/valid/neutral/8937Exp6fighting_face_223.jpg
  inflating: MMAFEDB/valid/neutral/8938Exp6fighting_face_23.jpg
  inflating: MMAFEDB/valid/neutral/8939Exp6fighting_face_239.jpg
  inflating: MMAFEDB/valid/neutral/893Exp6angry_european_197.jpg
  inflating: MMAFEDB/valid/neutral/8940Exp6fighting_face_242.jpg
  inflating: MMAFEDB/valid/neutral/8941Exp6fighting_face_246.jpg
  inflating: MMAFEDB/valid/neutral/8942Exp6fighting_face_267.jpg
  inflating: MMAFEDB/valid/neutral/8945Exp6fighting_face_281.jpg
  inflating: MMAFEDB/valid/neutral/8946Exp6fighting_face_282.jpg
```

```
# testing
import random
import keras
from tensorflow.keras import layers
from keras.models import Sequential
import tensorflow as tf
import tensorflow_datasets as tfds
from keras.layers.core import Dense, Activation, Dropout, Flatten
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.optimizers import SGD, RMSprop, adam
from keras.utils import np_utils
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
import matplotlib.pyplot as plt
import numpy as np
import os
import cv2
from PIL import Image
```

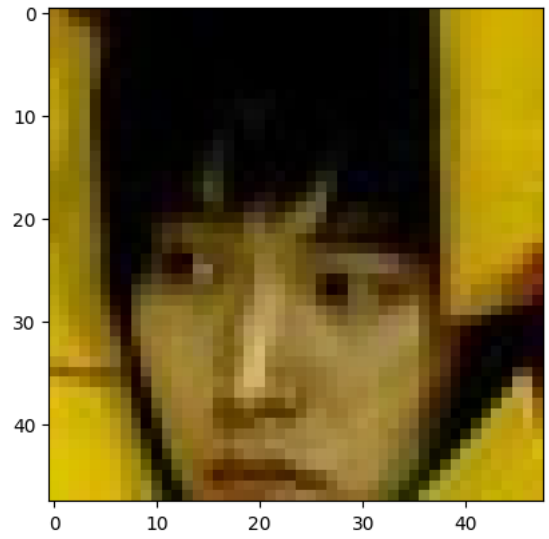
```
import matplotlib.pyplot as plt
from keras.utils import np_utils
from sklearn.model_selection import train_test_split
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import VGG16
from keras.models import Model
from tensorflow.keras import layers
from tensorflow.keras.applications.vgg16 import preprocess_input
import tensorflow_datasets as tfds


# setting up training data
path_train = "/content/MMAFEDB/train"

training = []
EMOTIONS = ["angry", "disgust", "fear", "happy", "neutral", "sad", "surprise"]
for emotion in EMOTIONS:
    path = os.path.join(path_train, emotion)
    class_num = EMOTIONS.index(emotion)
    count = 0
    for img in os.listdir(path):
        if count > 3200:
          break
        count+=1
        img_array = cv2.imread(os.path.join(path, img))
        img_array = cv2.cvtColor(img_array,cv2.COLOR_RGB2BGR)
        training.append([img_array, class_num])

# randomizes data
random.shuffle(training)
plt.imshow(training[0][0])
```

```
<matplotlib.image.AxesImage at 0x7fba195293f0>
```




```
!ls
```

```
gdrive  MMAFEDB  sample_data
```

```
# assign labels
X = []
y = []

for features, label in training:
    X.append(features)
    y.append(label)
X = np.array(X).reshape(-1, 48, 48, 3)
```

```
# normalizing x
X = X.astype('float32')
X /= 255
Y = np_utils.to_categorical(y, 7)

print(Y[100])
print(Y.shape)
```

```
[0. 0. 0. 1. 0. 0. 0.]
(22407, 7)
```

```
# splitting X and Y
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 7)


# model1
input_shape = (48, 48, 3)

model1 = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),

        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Dropout(0.2),


        layers.Conv2D(128, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),

        layers.Conv2D(256, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),

        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(100, activation="sigmoid"),
        layers.Dense(7, activation="sigmoid"),
    ]
)

model1.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 46, 46, 32)        896

 max_pooling2d_8 (MaxPooling  (None, 23, 23, 32)       0
 2D)

 conv2d_9 (Conv2D)           (None, 21, 21, 64)        18496

 max_pooling2d_9 (MaxPooling  (None, 10, 10, 64)       0
 2D)

 dropout_4 (Dropout)         (None, 10, 10, 64)        0

 conv2d_10 (Conv2D)          (None, 8, 8, 128)         73856

 max_pooling2d_10 (MaxPoolin  (None, 4, 4, 128)        0
 g2D)

 conv2d_11 (Conv2D)          (None, 2, 2, 256)         295168

 max_pooling2d_11 (MaxPoolin  (None, 1, 1, 256)        0
 g2D)

 flatten_2 (Flatten)         (None, 256)               0

 dropout_5 (Dropout)         (None, 256)               0

 dense_4 (Dense)             (None, 100)               25700

 dense_5 (Dense)             (None, 7)                 707

=================================================================
Total params: 414,823
```

```
Trainable params: 414,823
Non-trainable params: 0
_________________________________________________________________
```

```
batch_size = 32
epochs = 15

model1.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])

model1.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, validation_data =(X_test,y_test))
```

```
Epoch 1/15
561/561 [==============================] - 6s 8ms/step - loss: 1.8916 - accuracy: 0.2015 - val_loss: 1.8450 - val_accuracy: (
Epoch 2/15
561/561 [==============================] - 4s 8ms/step - loss: 1.7733 - accuracy: 0.2789 - val_loss: 1.7126 - val_accuracy: (
Epoch 3/15
561/561 [==============================] - 4s 7ms/step - loss: 1.7070 - accuracy: 0.3135 - val_loss: 1.6551 - val_accuracy: (
Epoch 4/15
561/561 [==============================] - 4s 7ms/step - loss: 1.6396 - accuracy: 0.3507 - val_loss: 1.5834 - val_accuracy: (
Epoch 5/15
561/561 [==============================] - 4s 7ms/step - loss: 1.5771 - accuracy: 0.3821 - val_loss: 1.5267 - val_accuracy: (
Epoch 6/15
561/561 [==============================] - 4s 8ms/step - loss: 1.5153 - accuracy: 0.4099 - val_loss: 1.5074 - val_accuracy: (
Epoch 7/15
561/561 [==============================] - 3s 6ms/step - loss: 1.4699 - accuracy: 0.4311 - val_loss: 1.4586 - val_accuracy: (
Epoch 8/15
561/561 [==============================] - 3s 6ms/step - loss: 1.4377 - accuracy: 0.4460 - val_loss: 1.4278 - val_accuracy: (
Epoch 9/15
561/561 [==============================] - 4s 8ms/step - loss: 1.3932 - accuracy: 0.4616 - val_loss: 1.4023 - val_accuracy: (
Epoch 10/15
561/561 [==============================] - 4s 7ms/step - loss: 1.3575 - accuracy: 0.4763 - val_loss: 1.4011 - val_accuracy: (
Epoch 11/15
561/561 [==============================] - 3s 6ms/step - loss: 1.3272 - accuracy: 0.4899 - val_loss: 1.4026 - val_accuracy: (
Epoch 12/15
561/561 [==============================] - 4s 7ms/step - loss: 1.2958 - accuracy: 0.4979 - val_loss: 1.3896 - val_accuracy: (
Epoch 13/15
561/561 [==============================] - 4s 7ms/step - loss: 1.2603 - accuracy: 0.5194 - val_loss: 1.3873 - val_accuracy: (
Epoch 14/15
561/561 [==============================] - 4s 6ms/step - loss: 1.2323 - accuracy: 0.5291 - val_loss: 1.4128 - val_accuracy: (
Epoch 15/15
561/561 [==============================] - 4s 7ms/step - loss: 1.2029 - accuracy: 0.5444 - val_loss: 1.4260 - val_accuracy: (
<keras.callbacks.History at 0x7fb9c815bd30>
```

```
path = "/content/gdrive/MyDrive/EmotionRecognition"
model1.save(path)
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convo
```