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

```
Mounted at /content/drive
```

```
import pandas as pd
import os
import glob as gb
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
```

```
TRAIN_DIR = "/content/drive/MyDrive/Emotion detection/train"
TEST_DIR = "/content/drive/MyDrive/Emotion detection/test"
BATCH_SIZE=64
```

```
for folder in os.listdir(TRAIN_DIR):
    files = gb.glob(pathname= str(TRAIN_DIR+ '/'+ folder + '/*.jpg'))
    print(f'For training data, found {len(files)} in folder {folder}')
```

```
For training data, found 10 in folder Fear
For training data, found 10 in folder Happiness
For training data, found 10 in folder Sadness
For training data, found 10 in folder Surprise
For training data, found 10 in folder Disgust
For training data, found 10 in folder Anger
```

```
for folder in os.listdir(TEST_DIR):
    files = gb.glob(pathname= str(TEST_DIR+ '/'+ folder + '/*.jpg'))
    print(f'For testing data, found {len(files)} in folder {folder}')
```

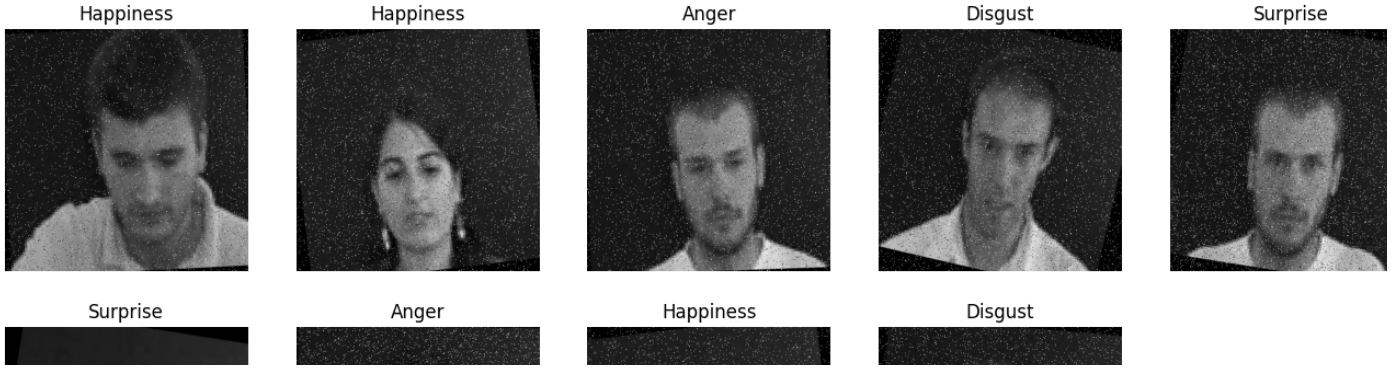
```
For testing data, found 10 in folder Surprise
For testing data, found 10 in folder Sadness
For testing data, found 10 in folder Happiness
For testing data, found 10 in folder Fear
For testing data, found 10 in folder Anger
For testing data, found 10 in folder Disgust
```

```
import random
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

def view_random_image(target_dir, target_class):
    # We will view images from here
    target_folder = target_dir + target_class

    # Get a random image path
    random_image = random.sample(os.listdir(target_folder), 1)

    # read in the image and plot it using matplolib
    img = mpimg.imread(target_folder+'/'+random_image[0])
    plt.imshow(img)
    plt.title(target_class)
    plt.axis('off')
    print(f"Image shape {img.shape}")

    return img
```

```
class_names = ['Anger','Disgust','Fear','Happiness','Sadness','Surprise']
```

```
plt.figure(figsize=(20,10))
for i in range(10):
    plt.subplot(3, 6, i+1)
    class_name = random.choice(class_names)
    img =view_random_image(target_dir="/content/drive/MyDrive/Emotion detection/train/", target_class=class_name)

```

```
Image shape (416, 416, 3)
Image shape (416, 416, 3)
Image shape (416, 416, 3)
Image shape (416, 416, 3)
Image shape (416, 416, 3)
Image shape (416, 416, 3)
Image shape (416, 416, 3)
Image shape (416, 416, 3)
Image shape (416, 416, 3)
Image shape (416, 416, 3)
```




```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)

training_set = train_datagen.flow_from_directory(TRAIN_DIR,
                                                 target_size = (128, 128),
                                                 batch_size = BATCH_SIZE,
                                                 class_mode = 'categorical')

test_set = test_datagen.flow_from_directory(TEST_DIR,
                                            target_size = (128, 128),
                                            batch_size = BATCH_SIZE,
                                            class_mode = 'categorical')
```

```
Found 60 images belonging to 6 classes.
Found 60 images belonging to 6 classes.
```

```
classifier = Sequential()

# Step 1 - Convolution
classifier.add(Conv2D(16, (3, 3), input_shape = (128, 128, 3), activation = 'relu'))

# Step 2 - Pooling
classifier.add(MaxPooling2D(pool_size = (2, 2)))
classifier.add(Conv2D(32, (3, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))



# Step 3 - Flattening
classifier.add(Flatten())

# Step 4 - Full connection
classifier.add(Dense(units = 128, activation = 'relu'))

classifier.add(Dense(units = 6, activation = 'softmax'))

# Compiling the CNN
classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

```
classifier.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 16)      448
```

```
 max_pooling2d (MaxPooling2  (None, 63, 63, 16)        0
 D)

 conv2d_1 (Conv2D)           (None, 61, 61, 32)        4640

 max_pooling2d_1 (MaxPoolin  (None, 30, 30, 32)        0
 g2D)

 flatten (Flatten)           (None, 28800)             0

 dense (Dense)               (None, 128)               3686528

 dense_1 (Dense)             (None, 6)                 774

=================================================================
Total params: 3692390 (14.09 MB)
Trainable params: 3692390 (14.09 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
history = classifier.fit(training_set,
                         epochs = 10,
                         validation_data = test_set)
classifier.save('model1.h5')
```

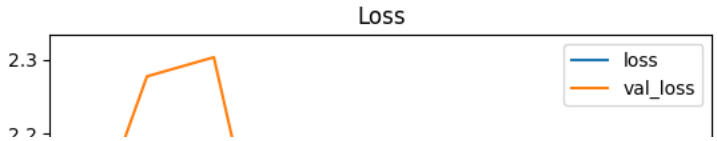
```
Epoch 1/10
1/1 [==============================] - 31s 31s/step - loss: 1.7927 - accuracy: 0.1667 - val_loss: 2.0238 - val_accuracy: 0.1667
Epoch 2/10
1/1 [==============================] - 3s 3s/step - loss: 1.8955 - accuracy: 0.2000 - val_loss: 2.2774 - val_accuracy: 0.1333
Epoch 3/10
1/1 [==============================] - 2s 2s/step - loss: 2.0151 - accuracy: 0.1667 - val_loss: 2.3034 - val_accuracy: 0.1667
Epoch 4/10
1/1 [==============================] - 2s 2s/step - loss: 2.0303 - accuracy: 0.1667 - val_loss: 1.9034 - val_accuracy: 0.1667
Epoch 5/10
1/1 [==============================] - 2s 2s/step - loss: 1.7657 - accuracy: 0.2500 - val_loss: 1.8760 - val_accuracy: 0.1667
Epoch 6/10
1/1 [==============================] - 2s 2s/step - loss: 1.7559 - accuracy: 0.2500 - val_loss: 1.8628 - val_accuracy: 0.2333
Epoch 7/10
1/1 [==============================] - 2s 2s/step - loss: 1.7694 - accuracy: 0.3000 - val_loss: 1.8622 - val_accuracy: 0.2500
Epoch 8/10
1/1 [==============================] - 3s 3s/step - loss: 1.7492 - accuracy: 0.2667 - val_loss: 1.8316 - val_accuracy: 0.2333
Epoch 9/10
1/1 [==============================] - 2s 2s/step - loss: 1.7226 - accuracy: 0.3500 - val_loss: 1.8112 - val_accuracy: 0.2000
Epoch 10/10
1/1 [==============================] - 2s 2s/step - loss: 1.6942 - accuracy: 0.3833 - val_loss: 1.8000 - val_accuracy: 0.1833
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model as an HDF5 file via `
  saving_api.save_model(
```

```
classifier.evaluate(test_set)
```

```
1/1 [==============================] - 0s 415ms/step - loss: 1.8000 - accuracy: 0.1833
[1.800031304359436, 0.18333333730697632]
```

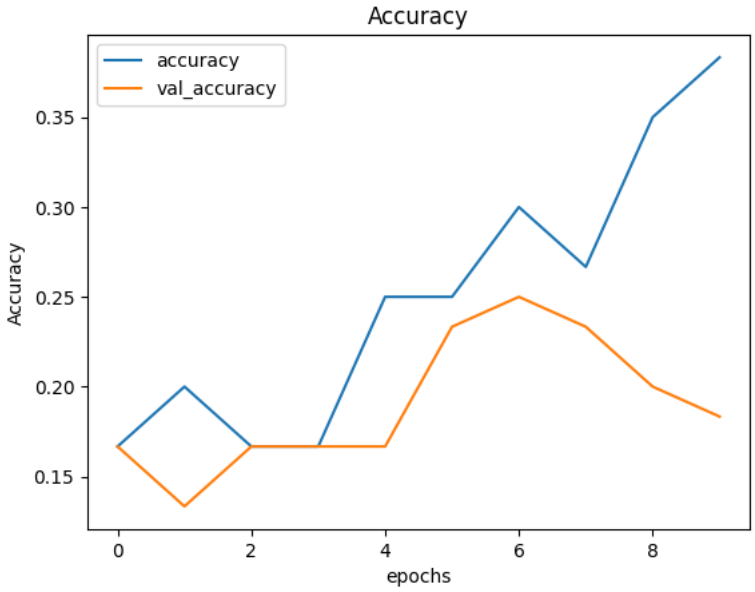
```
pd.DataFrame(history.history)[['loss','val_loss']].plot()
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('Loss')
```

```
Text(0, 0.5, 'Loss')
```




```
pd.DataFrame(history.history)[['accuracy','val_accuracy']].plot()
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('Accuracy')
```

```
Text(0, 0.5, 'Accuracy')
```




```
model_path = "model1.h5"
loaded_model = keras.models.load_model(model_path)

import matplotlib.pyplot as plt
import numpy as np
import cv2
from PIL import Image

image = cv2.imread("/content/drive/MyDrive/Emotion detection/test/Anger/myvideo_frames5_jpg.rf.f1726ea78f0bb1d5f16e3403e2d92468.jpg")

image_fromarray = Image.fromarray(image, 'RGB')
resize_image = image_fromarray.resize((128, 128))
expand_input = np.expand_dims(resize_image,axis=0)
input_data = np.array(expand_input)
input_data = input_data/255

pred = loaded_model.predict(input_data)
result = pred.argmax()
result
```

```
1/1 [==============================] - 0s 112ms/step
1
```

```
training_set.class_indices
```

```
{'Anger': 0,
 'Disgust': 1,
 'Fear': 2,
 'Happiness': 3,
 'Sadness': 4,
 'Surprise': 5}
```