In [1]:

```
import keras
from keras import backend as K
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras import optimizers
import pandas as pd
import numpy as np
from keras.utils import to_categorical

from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from keras.utils import np_utils
```

```
Using TensorFlow backend.
```

In [3]:

```
img_width, img_height = 128, 128
TrainData = pd.read_csv('TrainAnnotations.csv')

Annotations = TrainData['annotation'].tolist()
#display(Image(FileNames[0]))
#TrainData.shape[0]

Annotations_one_hot = to_categorical(Annotations)

print(Annotations_one_hot)


FileNames = ['TrainData\\'+ fname for fname in TrainData['file_name'].tolist()]
Train_image = []

for i in range(TrainData.shape[0]):
    #img = resize(FileNames[i],(32,32,3))
    img = keras.preprocessing.image.load_img(FileNames[i],target_size = (128,128))
    img = image.img_to_array(img)
    img = img/255
    Train_image.append(img)

X = np.array(Train_image)

# Check if the images are RGB and change the channels likewise
if K.image_data_format() == 'channels_first':
  input_shape= (3, img_width, img_height)
else:
  input_shape = (img_width, img_height, 3)

Y = np.array(Annotations)



X_train, X_val, Y_train, Y_val = train_test_split(X,Y, test_size = 0.2, random_state =
10 )
Y_train = np_utils.to_categorical(Y_train, 5)
Y_val = np_utils.to_categorical(Y_val, 5)
```

```
[[1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 ...
 [0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 1.]]
```

In [4]:

```
def get_model(learning_rate = 0.0001):

  model = Sequential()
  model.add(Conv2D(32,(3,3), input_shape = input_shape)) #32
  model.add(Activation('relu'))
  model.add(MaxPooling2D(pool_size = (2,2)))

  model.add(Conv2D(64,(3,3))) #64
  model.add(Activation('relu'))
  model.add(MaxPooling2D(pool_size = (2,2)))

  model.add(Conv2D(128,(2,2))) #128
  model.add(Activation('relu'))
  model.add(MaxPooling2D(pool_size = (2,2)))



  model.add(Flatten())

  model.add(Dense(64))
  model.add(Activation('relu'))
  model.add(Dropout(0.5))
  model.add(Dense(5))
  model.add(Activation('softmax'))

  optimizer = optimizers.adam(learning_rate)

  model.compile(loss = "categorical_crossentropy", optimizer = optimizer, metrics = ['a
ccuracy'])

  return model
```

In [5]:

```
batch_sizes = [16]

learning_rates = [0.001]

histories = []

epochs = 100

i = 0

for batch_size in batch_sizes:

  for learning_rate in learning_rates:

    print("Batch Size : {} Learning rate: {}".format(batch_size, learning_rate))

    #train_datagen = ImageDataGenerator(rescale = 1./255)

   # train_generator = train_datagen.flow_from_directory('TrainData\\', target_size =(i
mg_width, img_height),class_mode = 'categorical')

    #validation_generator = train_datagen.flow_from_directory( val_data_dir, target_siz
e =(img_width, img_height), batch_size = batch_size, class_mode ='categorical')

    model = get_model(learning_rate)

   # history = model.fit_generator(train_generator, steps_per_epoch = num_train // batc
h_size, epochs = epochs, validation_data = validation_generator, validation_steps = num
_val// batch_size)

    Train_fit = model.fit(X_train,Y_train,batch_size = 16, epochs = 50, validation_data
= (X_val, Y_val))

    model.save_weights("model_new"+ str(i+1) +".h5")

    histories.append(model.history.history)

    i = i + 1
```

```
Batch Size : 16 Learning rate: 0.001
Train on 1020 samples, validate on 255 samples
Epoch 1/50
1020/1020 [==============================] - 13s 13ms/step - loss: 1.4951
- accuracy: 0.3667 - val_loss: 1.4182 - val_accuracy: 0.3412
Epoch 2/50
1020/1020 [==============================] - 4s 4ms/step - loss: 1.3163 -
accuracy: 0.4873 - val_loss: 1.4759 - val_accuracy: 0.3529
Epoch 3/50
1020/1020 [==============================] - 4s 4ms/step - loss: 1.1469 -
accuracy: 0.5676 - val_loss: 1.0552 - val_accuracy: 0.5373
Epoch 4/50
1020/1020 [==============================] - 4s 4ms/step - loss: 1.0038 -
accuracy: 0.6049 - val_loss: 0.8736 - val_accuracy: 0.6392
Epoch 5/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.8648 -
accuracy: 0.6422 - val_loss: 0.8541 - val_accuracy: 0.6588
Epoch 6/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.7651 -
accuracy: 0.6873 - val_loss: 0.8033 - val_accuracy: 0.6510
Epoch 7/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.6728 -
accuracy: 0.7314 - val_loss: 0.8069 - val_accuracy: 0.7137
Epoch 8/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.6609 -
accuracy: 0.7382 - val_loss: 0.6619 - val_accuracy: 0.7569
Epoch 9/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.5881 -
accuracy: 0.7441 - val_loss: 0.6668 - val_accuracy: 0.7647
Epoch 10/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.4952 -
accuracy: 0.7902 - val_loss: 0.6809 - val_accuracy: 0.8039
Epoch 11/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.4313 -
accuracy: 0.8422 - val_loss: 0.5879 - val_accuracy: 0.8078
Epoch 12/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.3997 -
accuracy: 0.8431 - val_loss: 0.7515 - val_accuracy: 0.7647
Epoch 13/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.4076 -
accuracy: 0.8569 - val_loss: 0.9445 - val_accuracy: 0.7608
Epoch 14/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.3595 -
accuracy: 0.8627 - val_loss: 0.7781 - val_accuracy: 0.7804
Epoch 15/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.2975 -
accuracy: 0.8745 - val_loss: 0.6739 - val_accuracy: 0.8235
Epoch 16/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.2877 -
accuracy: 0.9088 - val_loss: 0.6374 - val_accuracy: 0.8353
Epoch 17/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.3005 -
accuracy: 0.8853 - val_loss: 0.8437 - val_accuracy: 0.8157
Epoch 18/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.3058 -
accuracy: 0.8961 - val_loss: 0.7123 - val_accuracy: 0.7922
Epoch 19/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.3026 -
accuracy: 0.8843 - val_loss: 0.8293 - val_accuracy: 0.8196
Epoch 20/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.2248 -
```

```
accuracy: 0.9186 - val_loss: 0.9640 - val_accuracy: 0.8000
Epoch 21/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.2155 -
accuracy: 0.9265 - val_loss: 1.0649 - val_accuracy: 0.8118
Epoch 22/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.2179 -
accuracy: 0.9235 - val_loss: 0.6653 - val_accuracy: 0.8157
Epoch 23/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.2386 -
accuracy: 0.9010 - val_loss: 0.9076 - val_accuracy: 0.8275
Epoch 24/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.2083 -
accuracy: 0.9324 - val_loss: 0.8432 - val_accuracy: 0.8431
Epoch 25/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.1786 -
accuracy: 0.9353 - val_loss: 1.1811 - val_accuracy: 0.8118
Epoch 26/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.1595 -
accuracy: 0.9324 - val_loss: 0.9141 - val_accuracy: 0.8392
Epoch 27/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.1507 -
accuracy: 0.9412 - val_loss: 0.9489 - val_accuracy: 0.8196
Epoch 28/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.1636 -
accuracy: 0.9382 - val_loss: 1.0894 - val_accuracy: 0.8431
Epoch 29/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.1580 -
accuracy: 0.9392 - val_loss: 1.1401 - val_accuracy: 0.8275
Epoch 30/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.1421 -
accuracy: 0.9461 - val_loss: 0.8718 - val_accuracy: 0.8510
Epoch 31/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.1351 -
accuracy: 0.9461 - val_loss: 1.0302 - val_accuracy: 0.8353
Epoch 32/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.1646 -
accuracy: 0.9363 - val_loss: 1.0261 - val_accuracy: 0.8235
Epoch 33/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.1478 -
accuracy: 0.9422 - val_loss: 1.1444 - val_accuracy: 0.8196
Epoch 34/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.1725 -
accuracy: 0.9275 - val_loss: 1.2011 - val_accuracy: 0.8118
Epoch 35/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.1217 -
accuracy: 0.9569 - val_loss: 1.0544 - val_accuracy: 0.8392
Epoch 36/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.1197 -
accuracy: 0.9608 - val_loss: 1.4117 - val_accuracy: 0.8196
Epoch 37/50
1020/1020 [==============================] - 4s 4ms/step - loss: 0.1506 -
accuracy: 0.9343 - val_loss: 1.1679 - val_accuracy: 0.8392
Epoch 38/50
1020/1020 [==============================] - 5s 5ms/step - loss: 0.1619 -
accuracy: 0.9382 - val_loss: 0.7688 - val_accuracy: 0.8471
Epoch 39/50
1020/1020 [==============================] - 6s 6ms/step - loss: 0.1566 -
accuracy: 0.9451 - val_loss: 1.1478 - val_accuracy: 0.8431
Epoch 40/50
1020/1020 [==============================] - 6s 6ms/step - loss: 0.1478 -
accuracy: 0.9382 - val_loss: 0.9320 - val_accuracy: 0.8392
```

```
Epoch 41/50
1020/1020 [==============================] - 6s 6ms/step - loss: 0.1206 -
accuracy: 0.9431 - val_loss: 1.2933 - val_accuracy: 0.8314
Epoch 42/50
1020/1020 [==============================] - 2s 2ms/step - loss: 0.1150 -
accuracy: 0.9539 - val_loss: 1.4205 - val_accuracy: 0.8157
Epoch 43/50
1020/1020 [==============================] - 2s 2ms/step - loss: 0.1547 -
accuracy: 0.9441 - val_loss: 1.5334 - val_accuracy: 0.8118
Epoch 44/50
1020/1020 [==============================] - 2s 2ms/step - loss: 0.1368 -
accuracy: 0.9324 - val_loss: 1.3388 - val_accuracy: 0.8353
Epoch 45/50
1020/1020 [==============================] - 1s 1ms/step - loss: 0.1185 -
accuracy: 0.9510 - val_loss: 1.1561 - val_accuracy: 0.8588
Epoch 46/50
1020/1020 [==============================] - 1s 1ms/step - loss: 0.1055 -
accuracy: 0.9627 - val_loss: 1.1825 - val_accuracy: 0.8510
Epoch 47/50
1020/1020 [==============================] - 1s 1ms/step - loss: 0.1067 -
accuracy: 0.9549 - val_loss: 1.0223 - val_accuracy: 0.8275
Epoch 48/50
1020/1020 [==============================] - 1s 1ms/step - loss: 0.1479 -
accuracy: 0.9412 - val_loss: 1.4110 - val_accuracy: 0.8235
Epoch 49/50
1020/1020 [==============================] - 1s 1ms/step - loss: 0.1353 -
accuracy: 0.9461 - val_loss: 1.2110 - val_accuracy: 0.8510
Epoch 50/50
1020/1020 [==============================] - 1s 1ms/step - loss: 0.1016 -
accuracy: 0.9549 - val_loss: 1.0653 - val_accuracy: 0.8431
```

In [6]:

```
import numpy
import sklearn.metrics as metrics
import glob
img_width, img_height = 128, 128
batch_size = 8
epochs = 100
Testpath = "TestData/*.jpg"
Testfiles = glob.glob(Testpath)
Test_image = []
for i in range (len(Testfiles)):
    img = keras.preprocessing.image.load_img(Testfiles[i],target_size = (128,128))
    img = image.img_to_array(img)
    img = img/255
    Test_image.append(img)


X_test = np.array(Test_image)
YPredict_test = model.predict(X_test)
Predict_test = np.argmax(YPredict_test, axis=1)
Predict_one_hot = to_categorical(Predict_test)

print(Predict_one_hot)
set(Predict_test)
result = pd.DataFrame(Predict_one_hot)
result.to_csv("predict.csv",header=False, index=False)
'''
test_set_dir = "/content/drive/My Drive/NN-ProjectC/Project_C1/Test/"

num_test = len(os.listdir(test_set_dir))

print ("Number of images in test set: ", num_test)

model = get_model(0.0001)

model.load_weights("/content/model_new1.h5")

test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(test_set_dir, target_size =(img_widt
h, img_height),
                                                  batch_size = batch_size, class_mode =
None, shuffle = False)

test_steps_per_epoch = numpy.math.ceil(test_generator.samples / test_generator.batch_si
ze)

predictions = model.predict_generator(test_generator, steps = test_steps_per_epoch)

print("PREDICTIONS: --->")
print(predictions)

predicted_classes = numpy.argmax(predictions, axis=1)

print("PREDICTED CLASSES: --->")
print (predicted_classes)
'''
```

```
[[0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
```

```
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[1. 0. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[1. 0. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
```

```
[1. 0. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[1. 0. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[1. 0. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 0. 0. 0. 1.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[0. 0. 0. 0. 1.]
[0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 0. 0. 0. 1.]
```

```
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]]
```

Out[6]:

```
'\ntest_set_dir = "/content/drive/My Drive/NN-ProjectC/Project_C1/Test/"\n
\nnum_test = len(os.listdir(test_set_dir))\n\nprint ("Number of images in
test set: ", num_test)\n\nmodel = get_model(0.0001)\n\nmodel.load_weights
("/content/model_new1.h5")\n  \ntest_datagen = ImageDataGenerator(rescale=
1./255)\n\ntest_generator = test_datagen.flow_from_directory(test_set_dir,
target_size =(img_width, img_height), \n
batch_size = batch_size, class_mode =None, shuffle = False) \n\ntest_steps
_per_epoch = numpy.math.ceil(test_generator.samples / test_generator.batch
_size)\n\npredictions = model.predict_generator(test_generator, steps = te
st_steps_per_epoch)\n\nprint("PREDICTIONS: --->")\nprint(predictions)\n\np
redicted_classes = numpy.argmax(predictions, axis=1)\n\nprint("PREDICTED C
LASSES: --->")\nprint (predicted_classes)\n'
```

In [ ]: