```
!nvidia-smi
```

```
Fri Apr 22 15:44:09 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   72C    P8    71W / 149W |      0MiB / 11441MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

```
USE_GPU = 1
```

```
# Import TensorFlow
import tensorflow as tf

# Print the installed TensorFlow version
print(f'TensorFlow version: {tf.__version__}\n')

# Get all GPU devices on this server
gpu_devices = tf.config.list_physical_devices('GPU')

# Print the name and the type of all GPU devices
print('Available GPU Devices:')
for gpu in gpu_devices:
    print('', gpu.name, gpu.device_type)
# Set only the GPU specified as USE_GPU to be visible
# tf.config.set_visible_devices(gpu_devices[USE_GPU], 'GPU')

# Get all visible GPU  devices on this server
visible_devices = tf.config.get_visible_devices('GPU')

# Print the name and the type of all visible GPU devices
print('\nVisible GPU Devices:')
for gpu in visible_devices:
    print('', gpu.name, gpu.device_type)
# Set the visible device(s) to not allocate all available memory at once,
# but rather let the memory grow whenever needed
for gpu in visible_devices:
    tf.config.experimental.set_memory_growth(gpu, True)
```

```
TensorFlow version: 2.8.0

Available GPU Devices:
  /physical_device:GPU:0 GPU

Visible GPU Devices:
  /physical_device:GPU:0 GPU
```

```
%matplotlib inline
```

```
import os
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

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

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

```
path = os.getcwd()
print(path)
```

```
/content
```

```
class_names = ['agricultural', 'airplane', 'baseballdiamond', 'beach', 'buildings',
               'chaparral', 'denseresidential', 'forest', 'freeway', 'golfcours
               'harbor', 'intersection', 'mediumresidential', 'mobilehomepark',
               'overpass', 'parkinglot', 'river', 'runway', 'sparseresidential'
               'storagetanks', 'tenniscourt']
```

```
os.chdir("drive/MyDrive/Challenge_dataset/train/")
```

```
print(os.getcwd())
print(os.path.join(os.path.dirname("__file__"),os.path.pardir))
path = os.path.abspath(os.path.join(os.path.dirname("__file__"),os.path.pardir))
os.chdir(path)
print(os.getcwd())
```

```
/content/drive/MyDrive/Challenge_dataset/train
..
/content/drive/MyDrive/Challenge_dataset
```

```
os.chdir(os.getcwd()+ "/train")
```

```
import cv2 as cv
from skimage import io
```

```
X = []
y = []
error = []
counter = 0

for c in class_names:
    os.chdir(os.getcwd() + "/" + c)
    print(os.getcwd())
    files = os.listdir(os.getcwd())
    print(files)
    for file in files: #遍历文件夹
        img = cv.imread(file)
        if(img.shape == (256, 256, 3)):
            X.append(img)
            y.append(counter)
        else:
            error.append(c + file)
    os.chdir(os.path.abspath(os.path.join(os.path.dirname("__file__"),os.path.pardir)))
    print(os.getcwd())
    counter += 1
```

```
/content/drive/MyDrive/Challenge_dataset/train/baseballdiamond
['baseballdiamond01.tif', 'baseballdiamond02.tif', 'baseballdiamond03.tif', 'baseballdiamo
/content/drive/MyDrive/Challenge_dataset/train
/content/drive/MyDrive/Challenge_dataset/train/beach
['beach00.tif', 'beach01.tif', 'beach02.tif', 'beach03.tif', 'beach04.tif', 'beach05.tif',
/content/drive/MyDrive/Challenge_dataset/train
/content/drive/MyDrive/Challenge_dataset/train/buildings
['buildings00.tif', 'buildings01.tif', 'buildings02.tif', 'buildings03.tif', 'buildings04.
/content/drive/MyDrive/Challenge_dataset/train
/content/drive/MyDrive/Challenge_dataset/train/chaparral
['chaparral00.tif', 'chaparral01.tif', 'chaparral02.tif', 'chaparral03.tif', 'chaparral04.
/content/drive/MyDrive/Challenge_dataset/train
/content/drive/MyDrive/Challenge_dataset/train/denseresidential
['denseresidential01.tif', 'denseresidential02.tif', 'denseresidential03.tif', 'denseresid
/content/drive/MyDrive/Challenge_dataset/train
/content/drive/MyDrive/Challenge_dataset/train/forest
['forest01.tif', 'forest00.tif', 'forest02.tif', 'forest03.tif', 'forest04.tif', 'forest05
/content/drive/MyDrive/Challenge_dataset/train
/content/drive/MyDrive/Challenge_dataset/train/freeway
['freeway01.tif', 'freeway00.tif', 'freeway02.tif', 'freeway03.tif', 'freeway04.tif', 'fre
/content/drive/MyDrive/Challenge_dataset/train
/content/drive/MyDrive/Challenge_dataset/train/golfcourse
['golfcourse00.tif', 'golfcourse01.tif', 'golfcourse02.tif', 'golfcourse03.tif', 'golfcour
/content/drive/MyDrive/Challenge_dataset/train
/content/drive/MyDrive/Challenge_dataset/train/harbor
['harbor01.tif', 'harbor02.tif', 'harbor03.tif', 'harbor04.tif', 'harbor05.tif', 'harbor06
/content/drive/MyDrive/Challenge_dataset/train
/content/drive/MyDrive/Challenge_dataset/train/intersection
['intersection01.tif', 'intersection02.tif', 'intersection03.tif', 'intersection04.tif', '
/content/drive/MyDrive/Challenge_dataset/train
/content/drive/MyDrive/Challenge_dataset/train/mediumresidential
['mediumresidential00.tif', 'mediumresidential01.tif', 'mediumresidential02.tif', 'mediumr
/content/drive/MyDrive/Challenge_dataset/train
/content/drive/MyDrive/Challenge_dataset/train/mobilehomepark
['mobilehomepark00.tif', 'mobilehomepark01.tif', 'mobilehomepark02.tif', 'mobilehomepark03
/content/drive/MyDrive/Challenge_dataset/train
/content/drive/MyDrive/Challenge_dataset/train/overpass
```

```
/content/drive/MyDrive/Challenge_dataset/train/overpass
['overpass00.tif', 'overpass01.tif', 'overpass02.tif', 'overpass03.tif', 'overpass04.tif',
/content/drive/MyDrive/Challenge_dataset/train
/content/drive/MyDrive/Challenge_dataset/train/parkinglot
['parkinglot01.tif', 'parkinglot02.tif', 'parkinglot03.tif', 'parkinglot04.tif', 'parkingl
/content/drive/MyDrive/Challenge_dataset/train
/content/drive/MyDrive/Challenge_dataset/train/river
['river00.tif', 'river01.tif', 'river02.tif', 'river03.tif', 'river04.tif', 'river05.tif',
/content/drive/MyDrive/Challenge_dataset/train
/content/drive/MyDrive/Challenge_dataset/train/runway
['runway01.tif', 'runway02.tif', 'runway03.tif', 'runway04.tif', 'runway05.tif', 'runway06
/content/drive/MyDrive/Challenge_dataset/train
/content/drive/MyDrive/Challenge_dataset/train/sparseresidential
['sparseresidential00.tif', 'sparseresidential01.tif', 'sparseresidential02.tif', 'sparser
/content/drive/MyDrive/Challenge_dataset/train
/content/drive/MyDrive/Challenge_dataset/train/storagetanks
['storagetanks00.tif', 'storagetanks01.tif', 'storagetanks02.tif', 'storagetanks03.tif', '
/content/drive/MyDrive/Challenge_dataset/train
/content/drive/MyDrive/Challenge_dataset/train/tenniscourt
['tenniscourt00.tif', 'tenniscourt01.tif', 'tenniscourt02.tif', 'tenniscourt03.tif', 'tenn
/content/drive/MyDrive/Challenge_dataset/train
```

```
X = np.array(X)
y = np.array(y)
print(X.shape, y.shape)
```

```
(1076, 256, 256, 3) (1076,)
```

```
print(X[1].shape)
```

```
(256, 256, 3)
```

```
os.chdir(os.path.abspath(os.path.join(os.path.dirname("__file__"),os.path.pardir)))
os.getcwd()
```

```
'/content/drive/MyDrive/Challenge_dataset'
```

```
os.getcwd()
```

```
'/content/drive/MyDrive/Challenge_dataset'
```

```
with open('train.npy', 'wb') as f:
    np.save(f, X)
    np.save(f, y)
```

```
os.chdir(os.getcwd()+"/test")
```

```
print(os.getcwd())
```

```
/content/drive/MyDrive/Challenge_dataset/test
```

```
X_test = []
y_test = []
counter = 0
error = []

for c in class_names:
    os.chdir(os.getcwd() + "/" + c)
    print(os.getcwd())
    files = os.listdir(os.getcwd())
    print(files)
    for file in files: #遍历文件夹
        img = cv.imread(file)

        if(img.shape == (256, 256, 3)):
            X_test.append(img)
            y_test.append(counter)
        else:
            error.append(c + file)

    os.chdir(os.path.abspath(os.path.join(os.path.dirname("__file__"),os.pardir)))
    print(os.getcwd())
    counter += 1
```

```
['baseballdiamond85.tif', 'baseballdiamond86.tif', 'baseballdiamond87.tif', 'baseballdiamo
/content/drive/MyDrive/Challenge_dataset/test
/content/drive/MyDrive/Challenge_dataset/test/beach
['beach89.tif', 'beach90.tif', 'beach91.tif', 'beach92.tif', 'beach93.tif', 'beach94.tif',
/content/drive/MyDrive/Challenge_dataset/test
/content/drive/MyDrive/Challenge_dataset/test/buildings
['buildings86.tif', 'buildings87.tif', 'buildings88.tif', 'buildings89.tif', 'buildings90.
/content/drive/MyDrive/Challenge_dataset/test
/content/drive/MyDrive/Challenge_dataset/test/chaparral
['chaparral86.tif', 'chaparral87.tif', 'chaparral88.tif', 'chaparral89.tif', 'chaparral90.
/content/drive/MyDrive/Challenge_dataset/test
/content/drive/MyDrive/Challenge_dataset/test/denseresidential
['denseresidential85.tif', 'denseresidential86.tif', 'denseresidential87.tif', 'denseresid
/content/drive/MyDrive/Challenge_dataset/test
/content/drive/MyDrive/Challenge_dataset/test/forest
['forest86.tif', 'forest87.tif', 'forest88.tif', 'forest89.tif', 'forest90.tif', 'forest91
/content/drive/MyDrive/Challenge_dataset/test
/content/drive/MyDrive/Challenge_dataset/test/freeway
['freeway87.tif', 'freeway88.tif', 'freeway89.tif', 'freeway90.tif', 'freeway91.tif', 'fre
/content/drive/MyDrive/Challenge_dataset/test
/content/drive/MyDrive/Challenge_dataset/test/golfcourse
['golfcourse87.tif', 'golfcourse88.tif', 'golfcourse89.tif', 'golfcourse90.tif', 'golfcour
/content/drive/MyDrive/Challenge_dataset/test
/content/drive/MyDrive/Challenge_dataset/test/harbor
['harbor85.tif', 'harbor86.tif', 'harbor87.tif', 'harbor88.tif', 'harbor89.tif', 'harbor90
/content/drive/MyDrive/Challenge_dataset/test
/content/drive/MyDrive/Challenge_dataset/test/intersection

['intersection86.tif', 'intersection87.tif', 'intersection88.tif', 'intersection89.tif', '
/content/drive/MyDrive/Challenge_dataset/test
/content/drive/MyDrive/Challenge_dataset/test/mediumresidential
['mediumresidential86.tif', 'mediumresidential87.tif', 'mediumresidential88.tif', 'mediumr
/content/drive/MyDrive/Challenge_dataset/test
/content/drive/MyDrive/Challenge_dataset/test/mobilehomepark
['mobilehomepark87.tif', 'mobilehomepark88.tif', 'mobilehomepark89.tif', 'mobilehomepark90
```

```
/content/drive/MyDrive/Challenge_dataset/test
/content/drive/MyDrive/Challenge_dataset/test/overpass
['overpass88.tif', 'overpass89.tif', 'overpass90.tif', 'overpass91.tif', 'overpass92.tif',
/content/drive/MyDrive/Challenge_dataset/test
/content/drive/MyDrive/Challenge_dataset/test/parkinglot
['parkinglot87.tif', 'parkinglot88.tif', 'parkinglot89.tif', 'parkinglot90.tif', 'parkingl
/content/drive/MyDrive/Challenge_dataset/test
/content/drive/MyDrive/Challenge_dataset/test/river
['river87.tif', 'river88.tif', 'river89.tif', 'river90.tif', 'river91.tif', 'river92.tif',
/content/drive/MyDrive/Challenge_dataset/test
/content/drive/MyDrive/Challenge_dataset/test/runway
['runway89.tif', 'runway90.tif', 'runway91.tif', 'runway92.tif', 'runway93.tif', 'runway94
/content/drive/MyDrive/Challenge_dataset/test
/content/drive/MyDrive/Challenge_dataset/test/sparseresidential
['sparseresidential87.tif', 'sparseresidential88.tif', 'sparseresidential89.tif', 'sparser
/content/drive/MyDrive/Challenge_dataset/test
/content/drive/MyDrive/Challenge_dataset/test/storagetanks
['storagetanks87.tif', 'storagetanks88.tif', 'storagetanks89.tif', 'storagetanks90.tif', '
/content/drive/MyDrive/Challenge_dataset/test
/content/drive/MyDrive/Challenge_dataset/test/tenniscourt
['tenniscourt91.tif', 'tenniscourt92.tif', 'tenniscourt93.tif', 'tenniscourt94.tif', 'tenn
/content/drive/MyDrive/Challenge_dataset/test
```

```
X_test  =  np.array(X_test)
y_test  =  np.array(y_test)
print(X_test.shape,  y_test.shape)
```

```
(276, 256, 256, 3) (276,)
```

```
print(X_test[1].shape)
print(y_test[1])
```

```
(256, 256, 3)
0
```

```
os.chdir(os.path.abspath(os.path.join(os.path.dirname("__file__"),os.path.pardir)))
os.getcwd()
```

```
'/content/drive/MyDrive/Challenge_dataset'
```

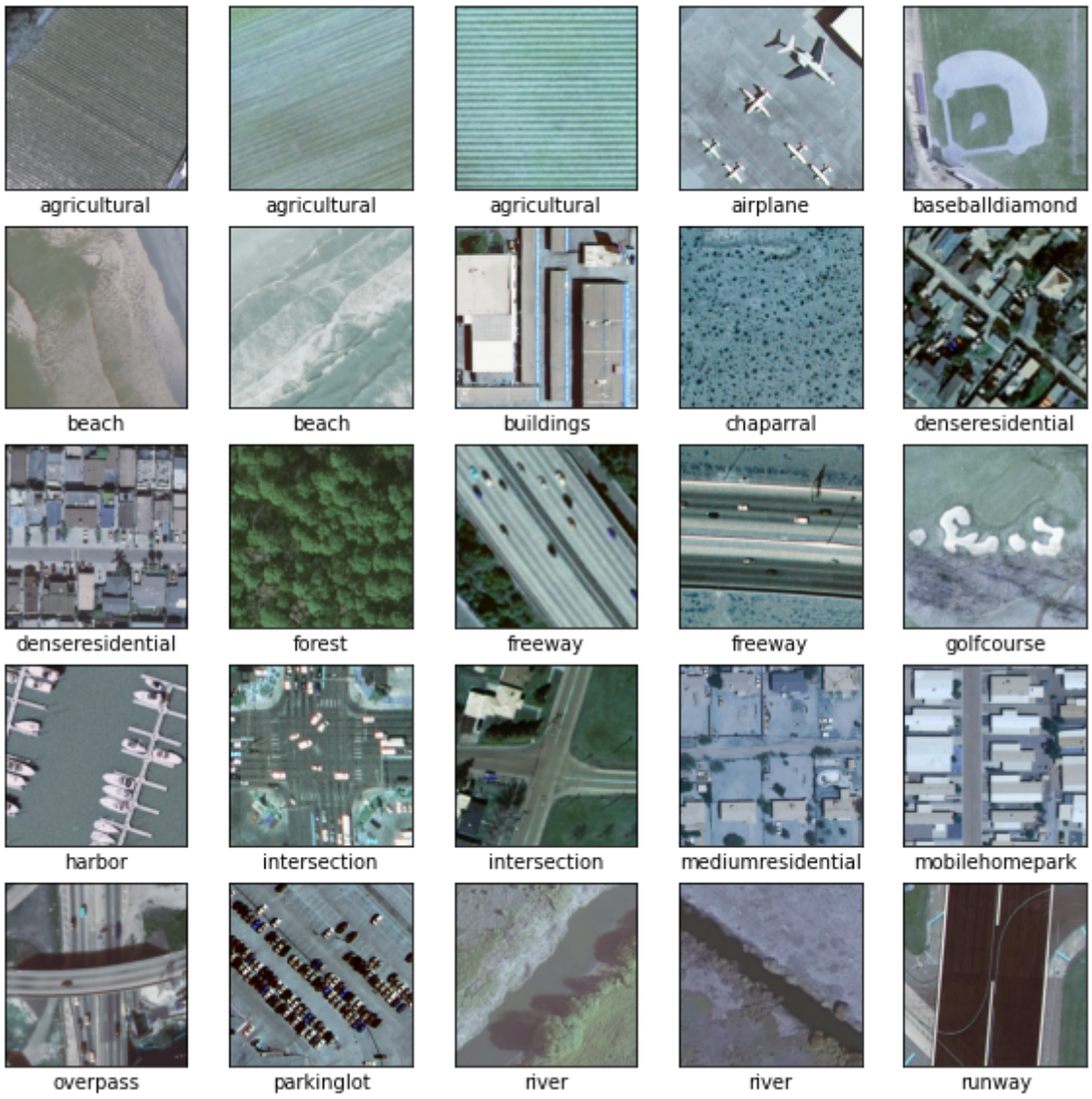
```
with  open('test.npy',  'wb')  as  f:
    np.save(f,  X_test)
    np.save(f,  y_test)
```

```
print(X[0])
```

```
[[[ 45  41  44]
  [ 56  50  53]
  [ 50  44  47]
  ...
  [122 126 125]
  [130 137 137]
  [111 119 119]]
```

```
        [[ 42  38  41]
         [ 46  43  46]
         [ 45  41  44]
         ...
         [ 81  83  83]
         [118 124 124]
         [126 131 135]]

        [[ 43  39  42]
         [ 44  41  44]
         [ 44  40  43]
         ...
         [ 91  95  90]
         [ 98 103  99]
         [ 97  99 101]]

        ...

        [[175 173 175]
         [189 191 190]
         [221 223 223]
         ...
         [136 144 144]
         [126 135 135]
         [118 123 124]]

        [[120 113 120]
         [150 147 151]
         [165 166 168]
         ...
         [117 121 123]
         [117 121 126]
         [124 129 133]]

        [[127 118 128]
         [110 103 113]
         [118 114 122]
         ...
         [ 99 101 101]
         [ 94  95  99]
         [127 129 135]]]

##########################################


# Normalize pixel values to be between 0 and 1
X, X_test = X / 255.0, X_test / 255.0


plt.figure(figsize=(10,10))
for i in range(0, 25):
        plt.subplot(5,5,i+1)
        plt.xticks([])
        plt.yticks([])
        plt.grid(False)
        plt.imshow(X[i*40])
        # The CIFAR labels happen to be arrays,
        # which is why you need the extra index
```

```
        plt.xlabel(class_names[y[i*40]])
plt.show()
```




```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)


# Instantiate the ResNet50 model
conv_base =tf.keras.applications.resnet.ResNet50(input_shape=(256, 256, 3),
                                                 include_top=False,
                                                 weights='imagenet')


# freeze the layers
for layer in conv_base.layers:
    layer.trainable = False


names = []
for layer in conv_base.layers:
        names.append(layer.name)
```

```
names[-1] # getting the name of the last conv layer
```

```
'conv5_block3_out'
```

```
last_layer = conv_base.get_layer('conv5_block3_out')
print('last layer output shape: ', last_layer.output_shape)
last_output = last_layer.output
```

```
last layer output shape: (None, 8, 8, 2048)
```

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten, ReLU, GlobalAveragePooling2D, Input


inputs = Input(shape = (256, 256, 3))
x = conv_base(inputs)
x = GlobalAveragePooling2D()(x)
x = Flatten()(x)
# Add a fully connected layer with 512 hidden units and ReLU activation
x = Dense(512, activation='relu')(x)
# Add a final sigmoid layer for classification
output = Dense(len(class_names), activation='softmax')(x)

model = Model(inputs, output)
```

```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 256, 256, 3)]     0

 resnet50 (Functional)       (None, 8, 8, 2048)        23587712

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 512)               1049088

 dense_1 (Dense)             (None, 21)                10773

=================================================================
Total params: 24,647,573
Trainable params: 1,059,861
Non-trainable params: 23,587,712
_________________________________________________________________
```

```
from • tensorflow.keras.optimizers • import • RMSprop

# • compile • the • model
model.compile(optimizer=RMSprop(lr=0.001),
```

```
              loss='SparseCategoricalCrossentropy',
              metrics=['accuracy'])
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/rmsprop.py:130: UserWarning: The `
  super(RMSprop, self).__init__(name, **kwargs)
```

```
X_train[1].shape
```

```
(256, 256, 3)
```

```
print(X_train.shape,  y_train.shape)
print(X_test.shape,  y_test.shape)
```

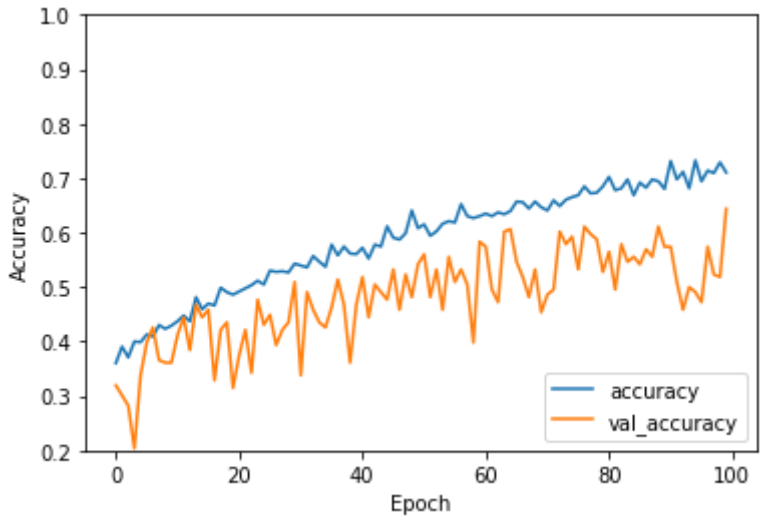
```
(860, 256, 256, 3) (860,)
(216, 256, 256, 3) (216,)
numpy.ndarray
```

```
history  =  model.fit(x  =  X_train,
                      y  =  y_train.reshape(860,1),
                      validation_data=(X_test,  y_test.reshape(216,1)),
                      epochs=100)
```

```
27/27 [==============================] - 11s 395ms/step - loss: 1.0142 - accuracy: 0.6593
Epoch 73/100
27/27 [==============================] - 11s 393ms/step - loss: 1.0401 - accuracy: 0.6488
Epoch 74/100
27/27 [==============================] - 11s 412ms/step - loss: 1.0356 - accuracy: 0.6605
Epoch 75/100
27/27 [==============================] - 11s 394ms/step - loss: 1.0121 - accuracy: 0.6651
Epoch 76/100
27/27 [==============================] - 11s 395ms/step - loss: 0.9968 - accuracy: 0.6686
Epoch 77/100
27/27 [==============================] - 11s 394ms/step - loss: 0.9732 - accuracy: 0.6849
Epoch 78/100
27/27 [==============================] - 11s 394ms/step - loss: 0.9994 - accuracy: 0.6721
Epoch 79/100
27/27 [==============================] - 11s 394ms/step - loss: 0.9640 - accuracy: 0.6733
Epoch 80/100
27/27 [==============================] - 11s 394ms/step - loss: 0.9717 - accuracy: 0.6849
Epoch 81/100
27/27 [==============================] - 11s 394ms/step - loss: 0.9484 - accuracy: 0.7023
Epoch 82/100
27/27 [==============================] - 11s 394ms/step - loss: 0.9472 - accuracy: 0.6779
Epoch 83/100
27/27 [==============================] - 11s 393ms/step - loss: 0.9878 - accuracy: 0.6814
Epoch 84/100
27/27 [==============================] - 11s 395ms/step - loss: 0.9136 - accuracy: 0.6977
Epoch 85/100
27/27 [==============================] - 11s 394ms/step - loss: 0.9648 - accuracy: 0.6686
Epoch 86/100
27/27 [==============================] - 11s 394ms/step - loss: 0.9383 - accuracy: 0.6919
Epoch 87/100
27/27 [==============================] - 11s 394ms/step - loss: 0.9338 - accuracy: 0.6826
Epoch 88/100
27/27 [==============================] - 11s 394ms/step - loss: 0.9585 - accuracy: 0.6977
```

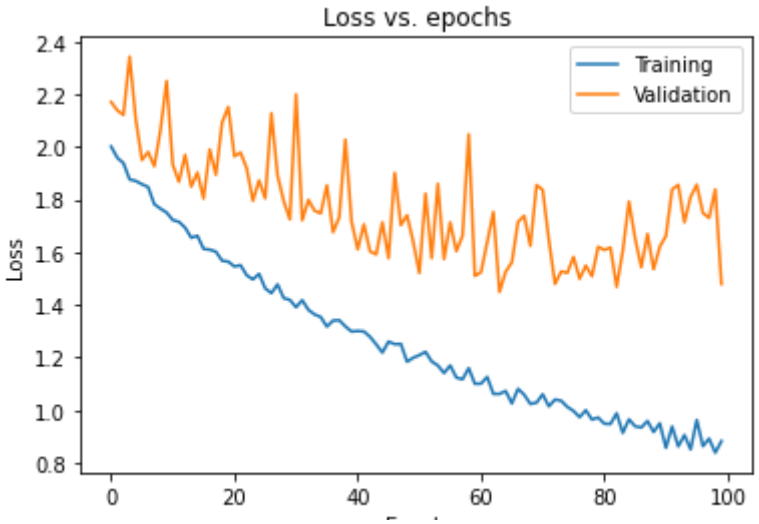
```
Epoch 89/100
27/27 [==============================] - 11s 394ms/step - loss: 0.9167 - accuracy: 0.6942
Epoch 90/100
27/27 [==============================] - 11s 395ms/step - loss: 0.9497 - accuracy: 0.6802
Epoch 91/100
27/27 [==============================] - 11s 394ms/step - loss: 0.8567 - accuracy: 0.7314
Epoch 92/100
27/27 [==============================] - 11s 394ms/step - loss: 0.9376 - accuracy: 0.6977
Epoch 93/100
27/27 [==============================] - 11s 412ms/step - loss: 0.8621 - accuracy: 0.7116
Epoch 94/100
27/27 [==============================] - 11s 394ms/step - loss: 0.9052 - accuracy: 0.6814
Epoch 95/100
27/27 [==============================] - 11s 395ms/step - loss: 0.8507 - accuracy: 0.7326
Epoch 96/100
27/27 [==============================] - 11s 394ms/step - loss: 0.9625 - accuracy: 0.6942
Epoch 97/100
27/27 [==============================] - 11s 412ms/step - loss: 0.8630 - accuracy: 0.7140
Epoch 98/100
27/27 [==============================] - 11s 395ms/step - loss: 0.8917 - accuracy: 0.7093
Epoch 99/100
27/27 [==============================] - 11s 393ms/step - loss: 0.8383 - accuracy: 0.7291
Epoch 100/100
27/27 [==============================] - 11s 394ms/step - loss: 0.8822 - accuracy: 0.7105
```

```
plt.plot(history.history['accuracy'],  label='accuracy')
plt.plot(history.history['val_accuracy'],  label  =  'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.2,  1])
plt.legend(loc='lower  right')
plt.show()
```




```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss  vs.  epochs')
plt.ylabel('Loss')
plt.xlabel('Epoch')
```

```
plt.legend(['Training', 'Validation'], loc='upper right')
plt.show()
```




```
# Save the weights
model.save_weights('./checkpoints/my_checkpoint')


inputs = Input(shape = (256, 256, 3))
x = conv_base(inputs)
x = GlobalAveragePooling2D()(x)
x = Flatten()(x)
# Add a fully connected layer with 512 hidden units and ReLU activation
x = Dense(512, activation='relu')(x)
# Add a final sigmoid layer for classification
output = Dense(len(class_names), activation='softmax')(x)

model2 = Model(inputs, output)


# Restore the weights
model2.load_weights('./checkpoints/my_checkpoint')
```

```
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7f0df7d63390>
```

```
with open("test.npy", "rb") as f:
    X_val = np.load(f)
    y_val = np.load(f)



X_val = X_val / 255.0


X_val[1].shape
```

```
(256, 256, 3)
```

```
classif_prob = model.predict(X_val)
print(classif_prob)
```

```
[[9.9079680e-01 1.8352343e-13 9.2491671e-04 ... 1.1319467e-07
```

```
        4.0909299e-08 7.9312094e-06]
       [5.9418303e-01 1.6796980e-09 4.7567502e-02 ... 6.5906934e-05
        5.9400845e-05 6.4122584e-03]
       [7.9076058e-01 4.1297371e-10 2.2103989e-02 ... 4.8845042e-05
        2.8628907e-05 3.1372164e-03]
       ...
       [8.1754662e-03 2.6119809e-04 3.2341129e-06 ... 3.5841912e-01
        2.4187183e-01 5.2144378e-02]
       [4.6716304e-04 7.2954346e-05 2.0611715e-02 ... 3.0913332e-04
        1.4982893e-01 4.7001913e-01]
       [1.3893687e-03 6.4356675e-05 1.4540068e-02 ... 5.2434229e-03
        9.7010687e-02 3.4245518e-01]]
```

```
pred_classes_argmax = np.argmax(classif_prob,axis=-1)
```

```
predicted_cls = pred_classes_argmax[0]
print("Predicted class:", predicted_cls)
print()
```

```
Predicted class: 0
```

```
values = model.evaluate(X_val, y_val)
print("{}:{}, {}:{}.".format(model.metrics_names[0],values[0],model.metrics_names[1],values[1]))
```

```
9/9 [==============================] - 4s 301ms/step - loss: 3.2318 - accuracy: 0.4167
loss:3.231823444366455,accuracy:0.4166666567325592.
```

✓ 6s completed at 6:40 PM