```
import os

# walk through the list of directories and number of files
total_image = 0
for dirpath, dirnames, filenames in os.walk("F:\SmartBridge-AI-Assignments\Assignment-3"):
    total_image = total_image + int(len(filenames))
print(total_image)
```

```
309
```

```
num_of_bird_groups = len(os.listdir("F:/SmartBridge-AI-Assignments/Assignment-3/train_data/train_data"))
num_of_bird_groups
```

```
16
```

```
import pathlib
import numpy as np

data_dir = pathlib.Path("F:/SmartBridge-AI-Assignments/Assignment-3/train_data/train_data")
class_names = np.array(sorted([item.name for item in data_dir.glob("*")])) # creating a list of class names from subdirectory
print(class_names)
```

```
['blasti' 'bonegl' 'brhkyt' 'cbrtsh' 'cmnmyn' 'gretit' 'hilpig' 'himbul'
 'himgri' 'hsparo' 'indvul' 'jglowl' 'lbicrw' 'mgprob' 'rebimg' 'wcrsrt']
```

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

def view_random_image(target_dir, target_class):
  # setting up the image directory
  target_folder = target_dir + target_class

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

  #read image and plotting it
  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
```

```
mg = view_random_image(target_dir = "F:/SmartBridge-AI-Assignments/Assignment-3/train_data/train_data/",
                        target_class = "himbul")
```

```
Image shape: (2160, 2880, 3)
```

```
import tensorflow as tf
img.shape #(width, height, colour channels)
```

```
(2160, 2880, 3)
```

```
plt.figure(figsize = (15,7))
plt.subplot(1,3,1)
steak_image = view_random_image("F:/SmartBridge-AI-Assignments/Assignment-3/train_data/train_data/", "blasti")
plt.subplot(1,3,2)
pizza_image = view_random_image("F:/SmartBridge-AI-Assignments/Assignment-3/train_data/train_data/", "gretit")
plt.subplot(1,3,3)
pizza_image = view_random_image("F:/SmartBridge-AI-Assignments/Assignment-3/train_data/train_data/", "indvul")
```

```
Image shape: (4000, 6016, 3)
Image shape: (2160, 2880, 3)
Image shape: (4000, 6016, 3)
```

blasti

gretit

indvul

```
import tensorflow as tf
import PIL
from keras.preprocessing.image import ImageDataGenerator
from pathlib import Path
from PIL import UnidentifiedImageError
from PIL import ImageFile
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

```
datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   validation_split = 0.2,
                                   horizontal_flip = True)
training_set = datagen.flow_from_directory("F:/SmartBridge-AI-
Assignments/Assignment-3/train_data/train_data",
                                                 target_size = (256, 256),
                                                 batch_size = 32,
                                                 subset='training',
                                                 class_mode = 'categorical')
```

```
Found 124 images belonging to 16 classes.
```

```
test_set = datagen.flow_from_directory("F:/SmartBridge-AI-
Assignments/Assignment-3/test_data/test_data",
                                                 target_size = (256, 256),
                                                 batch_size = 32,
                                                 subset='validation',
                                                 class_mode = 'categorical')
```

```
Found 26 images belonging to 16 classes.
```

```
cnn = tf.keras.models.Sequential()
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu',
input_shape=[256,256, 3]))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Flatten())
cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))
cnn.add(tf.keras.layers.Dropout(0.25))
cnn.add(tf.keras.layers.Dense(units=16, activation='softmax'))
```

```

cnn.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_2 (Conv2D)           (None, 254, 254, 32)      896       
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 127, 127, 32)     0         
 2D)                                                             
                                                                 
 conv2d_3 (Conv2D)           (None, 125, 125, 32)      9248      
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 62, 62, 32)       0         
 2D)                                                             
                                                                 
 flatten_1 (Flatten)         (None, 123008)            0         
                                                                 
 dense_2 (Dense)             (None, 128)               15745152  
                                                                 
 dropout_1 (Dropout)         (None, 128)               0         
                                                                 
 dense_3 (Dense)             (None, 16)                2064      
                                                                 
=================================================================
Total params: 15,757,360
Trainable params: 15,757,360
Non-trainable params: 0
```

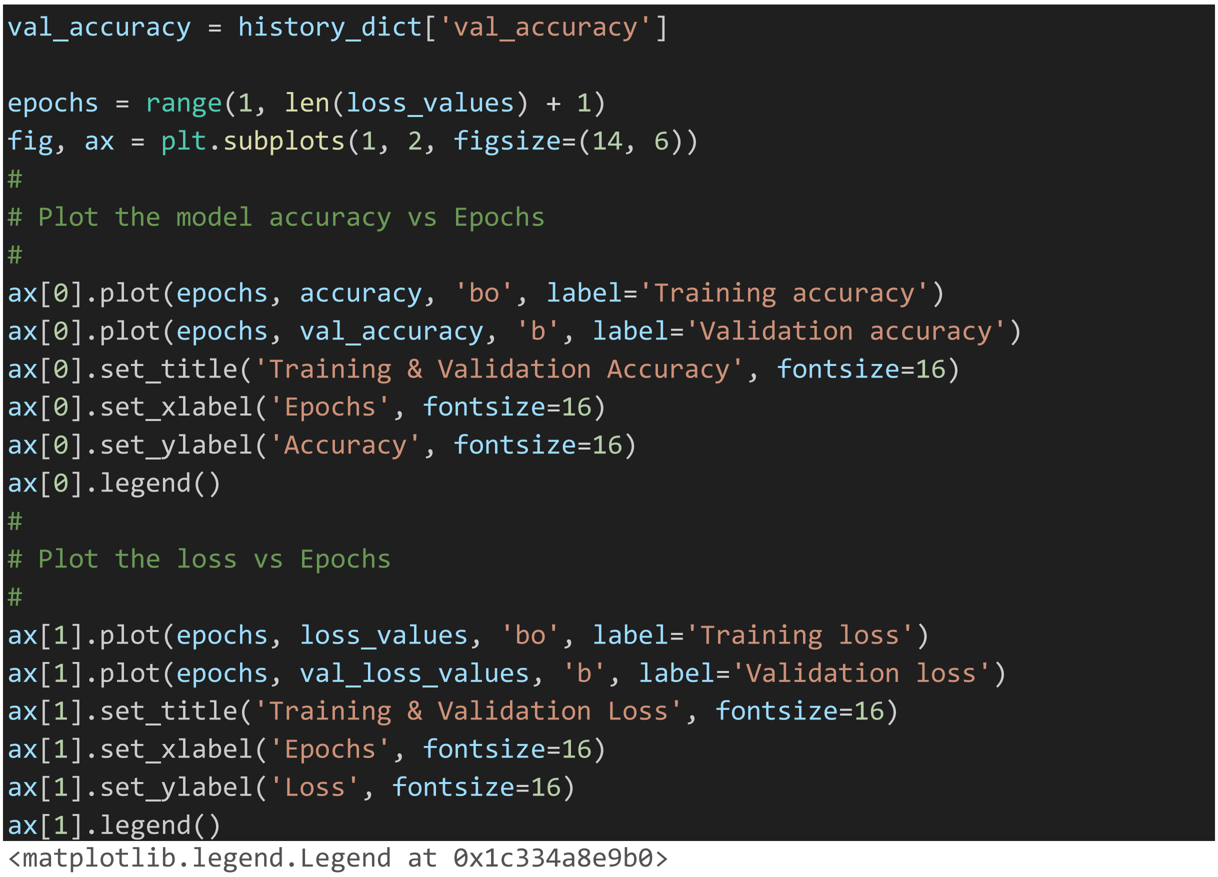
```
cnn.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics =
['accuracy'])
```

```
model=cnn.fit(x = training_set, validation_data = test_set, epochs = 40)
```

```
Epoch 1/40
4/4 [==============================] - 40s 12s/step - loss: 6.5691 - accuracy: 0.0726 - val_loss: 2.7876 - val_accuracy: 0.1154
Epoch 2/40
4/4 [==============================] - 34s 9s/step - loss: 2.9211 - accuracy: 0.1452 - val_loss: 2.6452 - val_accuracy: 0.1923
Epoch 3/40
4/4 [==============================] - 28s 8s/step - loss: 2.4738 - accuracy: 0.1855 - val_loss: 2.8303 - val_accuracy: 0.2308
Epoch 4/40
4/4 [==============================] - 27s 7s/step - loss: 2.4448 - accuracy: 0.1935 - val_loss: 2.6812 - val_accuracy: 0.3846
Epoch 5/40
4/4 [==============================] - 25s 7s/step - loss: 2.3176 - accuracy: 0.2903 - val_loss: 2.5915 - val_accuracy: 0.3077
Epoch 6/40
4/4 [==============================] - 26s 7s/step - loss: 2.1500 - accuracy: 0.3145 - val_loss: 2.5684 - val_accuracy: 0.3077
Epoch 7/40
4/4 [==============================] - 25s 6s/step - loss: 2.1298 - accuracy: 0.3548 - val_loss: 2.6096 - val_accuracy: 0.2308
Epoch 8/40
4/4 [==============================] - 27s 7s/step - loss: 1.9532 - accuracy: 0.3790 - val_loss: 2.4386 - val_accuracy: 0.3846
Epoch 9/40
4/4 [==============================] - 26s 7s/step - loss: 1.8426 - accuracy: 0.4032 - val_loss: 2.5029 - val_accuracy: 0.3077
Epoch 10/40
4/4 [==============================] - 25s 7s/step - loss: 1.6272 - accuracy: 0.5323 - val_loss: 2.2119 - val_accuracy: 0.3846
Epoch 11/40
4/4 [==============================] - 26s 7s/step - loss: 1.7367 - accuracy: 0.4919 - val_loss: 2.3173 - val_accuracy: 0.3846
Epoch 12/40
4/4 [==============================] - 25s 7s/step - loss: 1.5291 - accuracy: 0.5000 - val_loss: 2.4833 - val_accuracy: 0.6154
Epoch 13/40
...
Epoch 39/40
4/4 [==============================] - 27s 7s/step - loss: 0.1883 - accuracy: 0.9677 - val_loss: 4.2374 - val_accuracy: 0.5385
Epoch 40/40
4/4 [==============================] - 25s 6s/step - loss: 0.1344 - accuracy: 0.9839 - val_loss: 4.4630 - val_accuracy: 0.5769
```

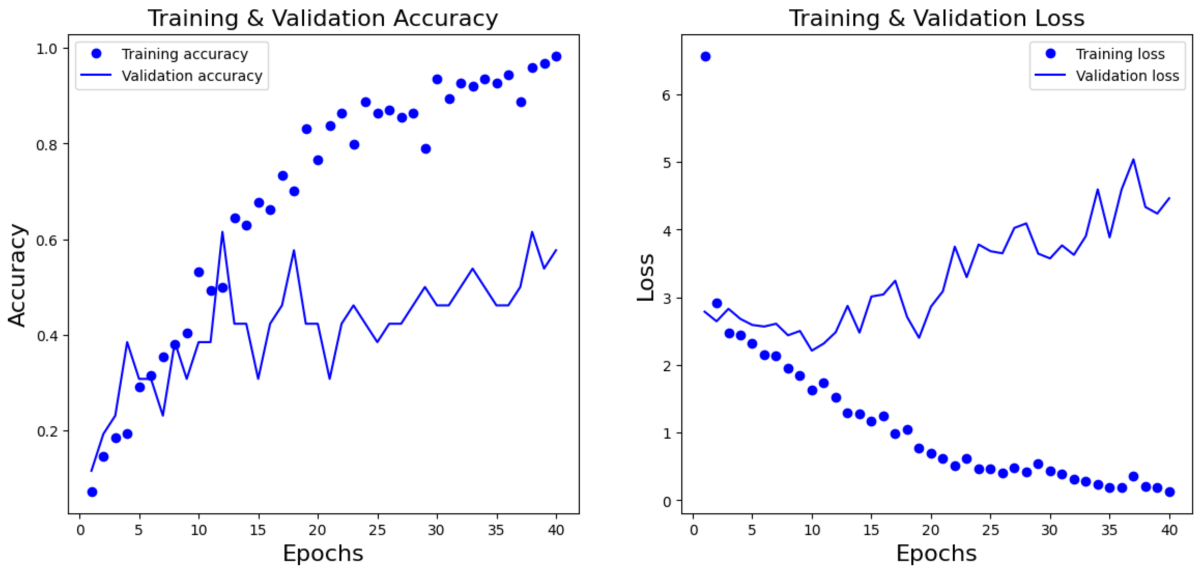
```
cnn.save('image_classifier.h5')
```

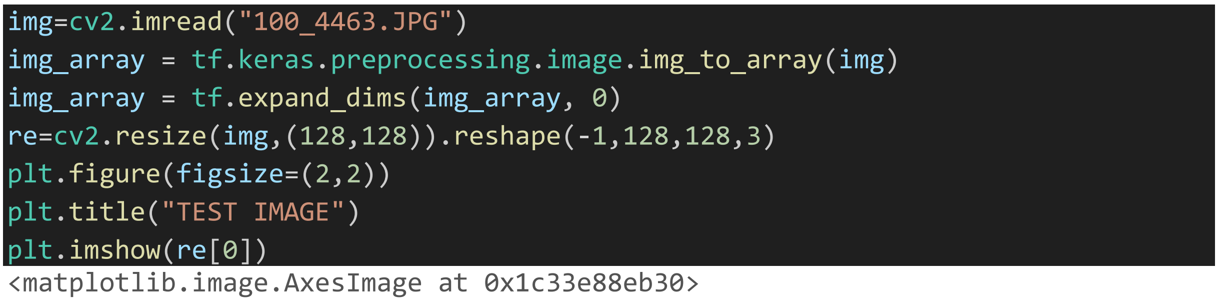
```
import matplotlib.pyplot as plt

history_dict = model.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
accuracy = history_dict['accuracy']
```



```
val_accuracy = history_dict['val_accuracy']

epochs = range(1, len(loss_values) + 1)
fig, ax = plt.subplots(1, 2, figsize=(14, 6))
#
# Plot the model accuracy vs Epochs
#
ax[0].plot(epochs, accuracy, 'bo', label='Training accuracy')
ax[0].plot(epochs, val_accuracy, 'b', label='Validation accuracy')
ax[0].set_title('Training & Validation Accuracy', fontsize=16)
ax[0].set_xlabel('Epochs', fontsize=16)
ax[0].set_ylabel('Accuracy', fontsize=16)
ax[0].legend()
#
# Plot the loss vs Epochs
#
ax[1].plot(epochs, loss_values, 'bo', label='Training loss')
ax[1].plot(epochs, val_loss_values, 'b', label='Validation loss')
ax[1].set_title('Training & Validation Loss', fontsize=16)
ax[1].set_xlabel('Epochs', fontsize=16)
ax[1].set_ylabel('Loss', fontsize=16)
ax[1].legend()
```

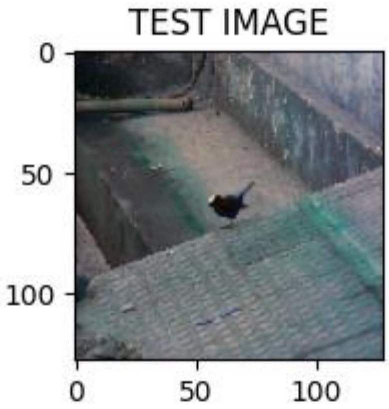
```
<matplotlib.legend.Legend at 0x1c334a8e9b0>
```

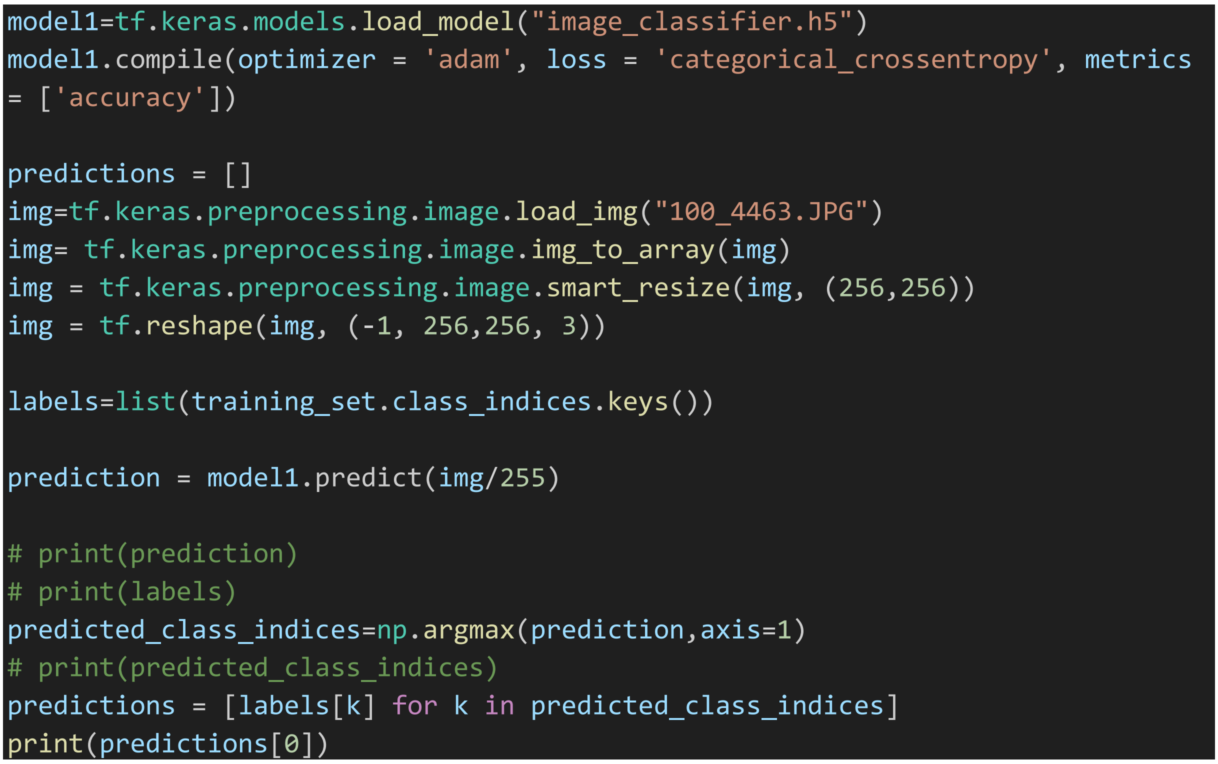




```
img=cv2.imread("100_4463.JPG")
img_array = tf.keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0)
re=cv2.resize(img,(128,128)).reshape(-1,128,128,3)
plt.figure(figsize=(2,2))
plt.title("TEST IMAGE")
plt.imshow(re[0])
```

```
<matplotlib.image.AxesImage at 0x1c33e88eb30>
```





```
model1=tf.keras.models.load_model("image_classifier.h5")
model1.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics
= ['accuracy'])

predictions = []
img=tf.keras.preprocessing.image.load_img("100_4463.JPG")
img= tf.keras.preprocessing.image.img_to_array(img)
img = tf.keras.preprocessing.image.smart_resize(img, (256,256))
img = tf.reshape(img, (-1, 256,256, 3))

labels=list(training_set.class_indices.keys())

prediction = model1.predict(img/255)

# print(prediction)
# print(labels)
predicted_class_indices=np.argmax(prediction,axis=1)
# print(predicted_class_indices)
predictions = [labels[k] for k in predicted_class_indices]
print(predictions[0])
```

```
1/1 [==============================] - 0s 182ms/step
hsparo
```