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

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

```
import matplotlib.pyplot as plt
import seaborn as sns

import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import adam_v2

from sklearn.metrics import classification_report,confusion_matrix

import tensorflow as tf

import cv2
import os

import numpy as np
```

```
labels = ['OSTRICH', 'PEACOCK']
img_size = 224
def get_data(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        print(path)
        for img in os.listdir(path):
            try:

                img_arr = cv2.imread(os.path.join(path, img))[...,::-1] #convert BGR to RGB f
                print(img_arr)
                resized_arr = cv2.resize(img_arr, (img_size, img_size)) # Reshaping images to
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data)
```

```
#Now we can easily fetch our train and validation data.
train = get_data('/content/drive/MyDrive/input/train')

val = get_data('/content/drive/MyDrive/input/test')
```

```
 [[147 161  40]
  [150 164  43]
  [159 173  50]
  ...
  [ 43 152  73]
  [ 31 128  57]
  [ 52 144  77]]]
[[[160 151 134]
  [130 127 108]
  [100 105  83]
  ...
  [ 84  95  53]
  [109 112  83]
  [ 83  83  59]]

 [[100  93  77]
  [120 117  98]
  [137 142 120]
  ...
  [115 123  82]
  [100 103  72]
  [105 108  81]]

 [[102 100  85]
  [ 82  83  65]
  [118 123 101]
  ...
  [114 121  79]
  [ 85  91  55]
  [ 85  91  57]]

 ...

 [[117 141  57]
  [104 129  45]
  [108 131  49]
  ...
  [125 148  76]
  [ 95 119  45]
  [ 97 121  47]]

 [[ 91 117  30]
  [101 129  44]
  [107 132  49]
  ...
  [118 140  67]
  [ 90 112  37]
  [110 133  55]]


 [[ 94 122  35]
  [ 90 118  33]
  [ 84 112  28]
  ...
  [ 81 103  30]
```

```
        [ 96 119  41]
        [107 130  50]]]
```

```
x_train = []
y_train = []
x_val = []
y_val = []

for feature, label in train:
  x_train.append(feature)
  y_train.append(label)

for feature, label in val:
  x_val.append(feature)
  y_val.append(label)

# Normalize the data
x_train = np.array(x_train) / 255
x_val = np.array(x_val) / 255

x_train.reshape(-1, img_size, img_size, 1)
y_train = np.array(y_train)

x_val.reshape(-1, img_size, img_size, 1)
y_val = np.array(y_val)
```

```
datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over the dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs by std of the dataset
        samplewise_std_normalization=False,  # divide each input by its std
        zca_whitening=False,  # apply ZCA whitening
        rotation_range = 30,  # randomly rotate images in the range (degrees, 0 to 180)
        zoom_range = 0.2, # Randomly zoom image
        width_shift_range=0.1,  # randomly shift images horizontally (fraction of total width
        height_shift_range=0.1,  # randomly shift images vertically (fraction of total height
        horizontal_flip = True,  # randomly flip images
        vertical_flip=False)  # randomly flip images


datagen.fit(x_train)
```

```
model = Sequential()
model.add(Conv2D(32,3,padding="same", activation="relu", input_shape=(224,224,3)))
model.add(MaxPool2D())

model.add(Conv2D(32, 3, padding="same", activation="relu"))
model.add(MaxPool2D())
```

```
model.add(Conv2D(64, 3, padding="same", activation="relu"))
model.add(MaxPool2D())
model.add(Dropout(0.4))

model.add(Flatten())
model.add(Dense(128,activation="relu"))
model.add(Dense(2, activation="softmax"))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 32)      896

 max_pooling2d (MaxPooling2D  (None, 112, 112, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 112, 112, 32)      9248

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 56, 56, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 28, 28, 64)       0
 2D)

 dropout (Dropout)           (None, 28, 28, 64)        0

 flatten (Flatten)           (None, 50176)             0

 dense (Dense)               (None, 128)               6422656

 dense_1 (Dense)             (None, 2)                 258

=================================================================
Total params: 6,451,554
Trainable params: 6,451,554
Non-trainable params: 0
_________________________________________________________________
```

```
from keras.optimizer_v2.adam import Adam
opt = Adam(learning_rate=0.000001)
model.compile(optimizer = opt , loss = tf.keras.losses.SparseCategoricalCrossentropy(from_log
```

```
history = model.fit(x_train,y_train,epochs = 150 , validation_data = (x_val, y_val))
```

```
Epoch 123/150
```

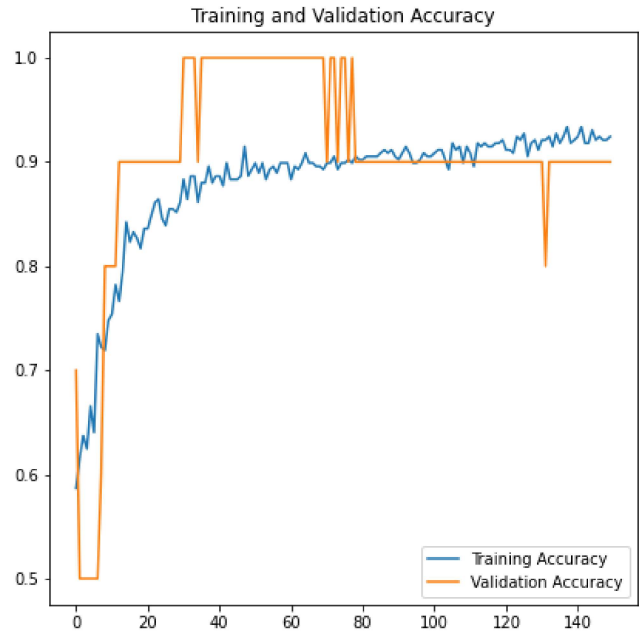
```
10/10 [==============================] - 20s 2s/step - loss: 0.2585 - accuracy: 0.908
Epoch 124/150
10/10 [==============================] - 20s 2s/step - loss: 0.2541 - accuracy: 0.924
Epoch 125/150
10/10 [==============================] - 20s 2s/step - loss: 0.2529 - accuracy: 0.921
Epoch 126/150
10/10 [==============================] - 20s 2s/step - loss: 0.2527 - accuracy: 0.927
Epoch 127/150
10/10 [==============================] - 20s 2s/step - loss: 0.2482 - accuracy: 0.905
Epoch 128/150
10/10 [==============================] - 20s 2s/step - loss: 0.2497 - accuracy: 0.918
Epoch 129/150
10/10 [==============================] - 20s 2s/step - loss: 0.2460 - accuracy: 0.921
Epoch 130/150
10/10 [==============================] - 20s 2s/step - loss: 0.2477 - accuracy: 0.911
Epoch 131/150
10/10 [==============================] - 20s 2s/step - loss: 0.2475 - accuracy: 0.921
Epoch 132/150
10/10 [==============================] - 20s 2s/step - loss: 0.2453 - accuracy: 0.921
Epoch 133/150
10/10 [==============================] - 20s 2s/step - loss: 0.2420 - accuracy: 0.924
Epoch 134/150
10/10 [==============================] - 20s 2s/step - loss: 0.2421 - accuracy: 0.914
Epoch 135/150
10/10 [==============================] - 20s 2s/step - loss: 0.2349 - accuracy: 0.927
Epoch 136/150
10/10 [==============================] - 20s 2s/step - loss: 0.2350 - accuracy: 0.918
Epoch 137/150
10/10 [==============================] - 20s 2s/step - loss: 0.2417 - accuracy: 0.924
Epoch 138/150
10/10 [==============================] - 20s 2s/step - loss: 0.2320 - accuracy: 0.933
Epoch 139/150
10/10 [==============================] - 20s 2s/step - loss: 0.2365 - accuracy: 0.918
Epoch 140/150
10/10 [==============================] - 20s 2s/step - loss: 0.2325 - accuracy: 0.921
Epoch 141/150
10/10 [==============================] - 20s 2s/step - loss: 0.2296 - accuracy: 0.924
Epoch 142/150
10/10 [==============================] - 20s 2s/step - loss: 0.2325 - accuracy: 0.933
Epoch 143/150
10/10 [==============================] - 20s 2s/step - loss: 0.2336 - accuracy: 0.918
Epoch 144/150
10/10 [==============================] - 20s 2s/step - loss: 0.2274 - accuracy: 0.918
Epoch 145/150
10/10 [==============================] - 20s 2s/step - loss: 0.2264 - accuracy: 0.930
Epoch 146/150
10/10 [==============================] - 20s 2s/step - loss: 0.2290 - accuracy: 0.921
Epoch 147/150
10/10 [==============================] - 20s 2s/step - loss: 0.2277 - accuracy: 0.924
Epoch 148/150
10/10 [==============================] - 20s 2s/step - loss: 0.2247 - accuracy: 0.921
Epoch 149/150
10/10 [==============================] - 20s 2s/step - loss: 0.2285 - accuracy: 0.921
Epoch 150/150
10/10 [==============================] - 20s 2s/step - loss: 0.2212 - accuracy: 0.924
```

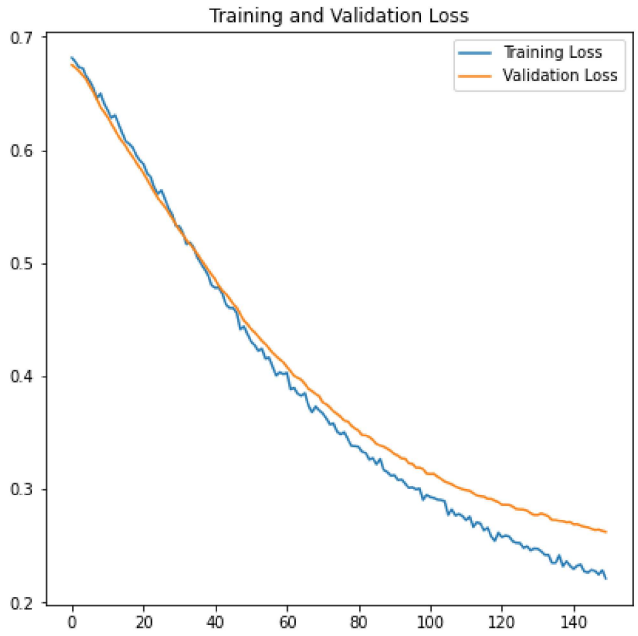
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(150)

plt.figure(figsize=(15, 15))
plt.subplot(2, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(2, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```






```
predictions = np.argmax(model.predict(x_val), axis=-1),
predictions = np.asarray(predictions)
predictions = predictions.reshape(1,-1)[0]
print(classification_report(y_val, predictions, target_names = ['Ostrich (Class 0)','Peacock
```

```
                    precision    recall  f1-score   support

Ostrich (Class 0)       1.00      0.80      0.89         5
Peacock (Class 1)       0.83      1.00      0.91         5

          accuracy                           0.90        10
         macro avg       0.92      0.90      0.90        10
      weighted avg       0.92      0.90      0.90        10
```

Task 1: Run the above code with given dataset.

Task 2: Run the code with different dataset