Devices and circuits for Artificial Intelligence

-- 540438 - Maurizio La Rosa --

Final exam project

In this notebook I introduce the project for the final exam of the course *Devices and circuits for artificial intelligence* from the Data Analysis degree of the University of Messina. The project consists in building a machine learning model for image classification.

The dataset to be used is hosted at kaggle, and is called BIRDS 525 SPECIES- IMAGE CLASSIFICATION. The dataset currently contains images from 525 bird species to be classified by the model. I downloaded the dataset on April, 17th, 2023, and that version contains 515 bird species.

It is useful to note that images in the dataset should have all the exact same shape (224, 224, 3), while I found that all images of the 'PLUSH CRESTED JAY' species and one image from the 'DON'T REMEMBER THE SPECIES, FILL INFO WITH FUTURE COMMIT' species have variable shapes. Hence, in my code, I check for images' shapes and remove images that don't match the common shape. This is important because imported images have the shape of 3D Numpy (np, when imported) arrays and I need to transform the list of images into a 4D Numpy array. The function np.array() can do it automatically when fed a list of 3D Numpy arrays, but images must have all the same shape.

The following cell is for importing necessary modules in the file used for describing the data, the model and the results. I previously uploaded data for bird species classification on Colab, in the *content/kaggle_data* folder.

In [1]:
```
#########################################
### allow importing from Google Drive  ###
###  after uploading the needed files  ###
#########################################
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
import sys
sys.path.insert(0,'/content/drive/MyDrive/Colab Notebooks/da_dcAI_project

#########################################
###  unzip entire dataset from Google  ###
###  Drive to colab's content folder   ###
#########################################
!unzip '/content/drive/MyDrive/Colab Notebooks/da_dcAI_project/reshaped_d

#########################################
### import necessary or useful modules ###
#########################################
import os
import numpy as np
#import tensorflow as tf
from    matplotlib                 import pyplot as plt
from    matplotlib                 import image as mpimg
from    tensorflow.keras.utils     import to_categorical
from    keras.preprocessing.image import ImageDataGenerator
from    sklearn.metrics            import classification_report, confusion
from    a_selectRandomFolders      import selectRandomFolders
from    b_viewClasses              import viewRandomClasses, viewRandomCla
from    c_selectData               import selectData, classesToInt, countL
from    d_sequentialModel          import seqModel
from    e_plotConfusionMatrix      import plotCM
```

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

Retrieve data folders

The following code cell is used for retrieving the system folders where the data are located.

In [2]:
```
### 1) set path to the directory of the dataset (this is the targetFolder
###    and show how many bird classes there are in the path and their nam
trainPathWin = 'C:/Users/mzlarosa/OneDrive - unime.it/Learning/CdL Inform
testPathWin = 'C:/Users/mzlarosa/OneDrive - unime.it/Learning/CdL Informa
validPathWin = 'C:/Users/mzlarosa/OneDrive - unime.it/Learning/CdL Inform
trainPathMac = '/Users/mau/OneDrive - unime.it/Learning/CdL Informatica/A
testPathMac = '/Users/mau/OneDrive - unime.it/Learning/CdL Informatica/An
validPathMac = '/Users/mau/OneDrive - unime.it/Learning/CdL Informatica/A
trainPath = '/content/reshaped/train'
testPath = '/content/reshaped/test'
validPath = '/content/reshaped/valid'
classes = os.listdir(trainPath)
if '.DS_Store' in classes:
    classes.remove('.DS_Store')
nOfClasses = len(classes)
print('\nThere are', nOfClasses, 'classes in the dataset.\n' +
      '\nHere is a list of the first 50 classes:')
print(classes[0 : 50], end = '')
print('[...]')
```

```
There are 515 classes in the dataset.

Here is a list of the first 50 classes:
['WHITE CRESTED HORNBILL', 'GOLDEN CHLOROPHONIA', 'EASTERN BLUEBIRD', 'N
ORTHERN GANNET', 'ANDEAN LAPWING', 'RED HEADED WOODPECKER', 'MALEO', 'EN
GGANO MYNA', 'ALPINE CHOUGH', 'GILDED FLICKER', 'CAPUCHINBIRD', 'INDIAN
ROLLER', 'RED KNOT', 'CRESTED NUTHATCH', 'CINNAMON FLYCATCHER', 'MYNA',
'GREY PLOVER', 'BANDED BROADBILL', 'WILLOW PTARMIGAN', 'RED BROWED FINCH
', 'SMITHS LONGSPUR', 'BLUE COAU', 'NORTHERN BEARDLESS TYRANNULET', 'COM
MON HOUSE MARTIN', 'RED WISKERED BULBUL', 'GUINEAFOWL', 'RED BILLED TROP
ICBIRD', 'TURKEY VULTURE', 'DEMOISELLE CRANE', 'BEARDED BELLBIRD', 'NICO
BAR PIGEON', 'ASHY STORM PETREL', 'ROADRUNNER', 'DUSKY LORY', 'GOLDEN PA
RAKEET', 'FAIRY BLUEBIRD', 'KAGU', 'OSTRICH', 'YELLOW CACIQUE', 'KIWI',
'HARLEQUIN DUCK', 'VARIED THRUSH', 'BLACK COCKATO', 'WHITE CHEEKED TURAC
O', 'ABBOTTS BOOBY', 'SPOON BILED SANDPIPER', 'WOOD DUCK', 'ROSE BREASTE
D GROSBEAK', 'BLONDE CRESTED WOODPECKER', 'GREAT JACAMAR'][...]
```

Introduce the data

The number of available pictures varies with the class.
I select a random sample of 15 bird species from the train data and show how many images are available for each sampled species.

In [3]:
```
### 2) select a random sample of n (15) subfolders from the targetFolder
###    and show their content (subfolders represent bird classes)
###    modules: os, random, selectRandomFolders
targetClasses = selectRandomFolders(trainPath, 15)
```

```
There are 0 folders and 163 image files in /content/reshaped/train/CREST
ED KINGFISHER
There are 0 folders and 140 image files in /content/reshaped/train/POMAR
INE JAEGER
There are 0 folders and 133 image files in /content/reshaped/train/RED H
EADED WOODPECKER
There are 0 folders and 157 image files in /content/reshaped/train/HORNE
D GUAN
There are 0 folders and 162 image files in /content/reshaped/train/CARMI
NE BEE-EATER
There are 0 folders and 144 image files in /content/reshaped/train/KAGU
There are 0 folders and 204 image files in /content/reshaped/train/JACOB
IN PIGEON
There are 0 folders and 162 image files in /content/reshaped/train/VIOLE
T TURACO
There are 0 folders and 160 image files in /content/reshaped/train/SPANG
LED COTINGA
There are 0 folders and 160 image files in /content/reshaped/train/BORNE
AN LEAFBIRD
There are 0 folders and 144 image files in /content/reshaped/train/SPOON
BILED SANDPIPER
There are 0 folders and 149 image files in /content/reshaped/train/CAPE
MAY WARBLER
There are 0 folders and 159 image files in /content/reshaped/train/MALEO
There are 0 folders and 168 image files in /content/reshaped/train/LUCIF
ER HUMMINGBIRD
There are 0 folders and 135 image files in /content/reshaped/train/CRANE
HAWK
```

Plot pictures from sample species

Plot one picture from 15 randomly selected species

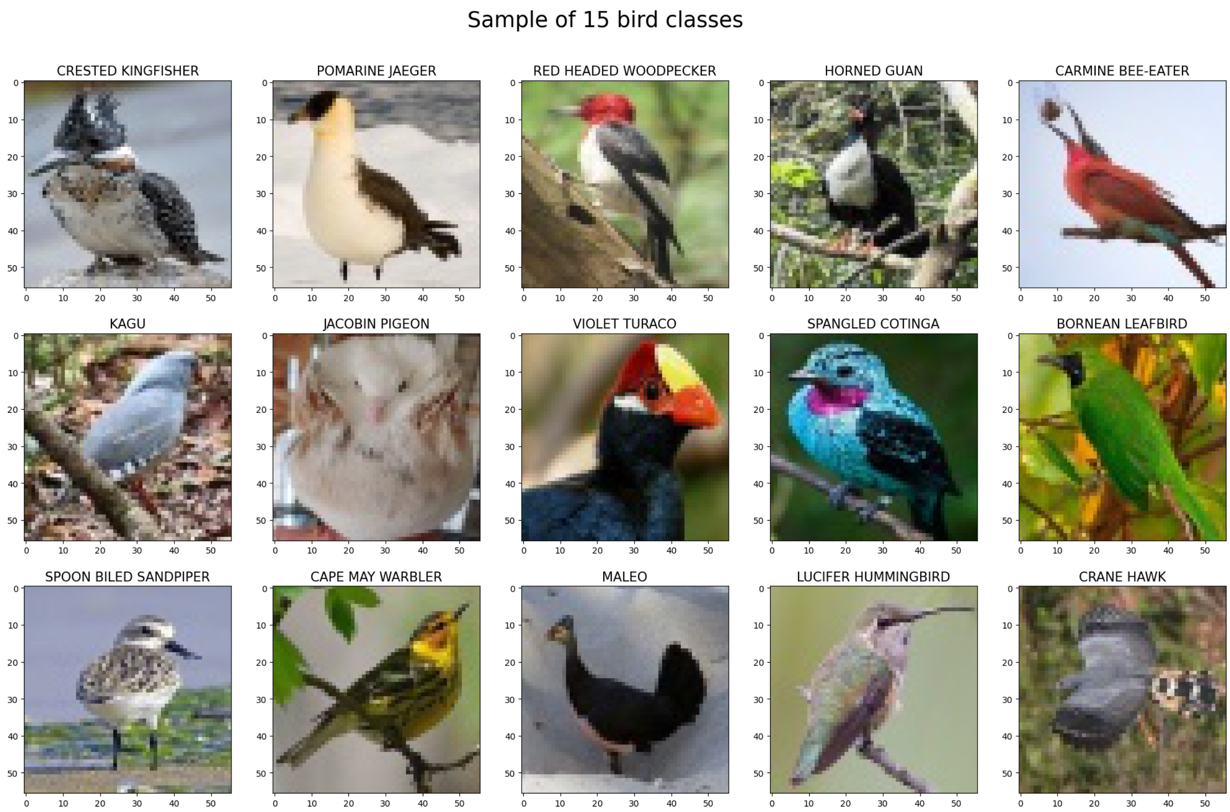
Then I draw a random picture from each class and show their shape and size. A picture's shape shows tipically three dimensions. The first two dimensions build a 2D matrix of n rows by m columns. The number of rows represents the image height in pixels, while the number of columns represents the image width in pixels. So each matrix coordinate point represents the intensity value of a single pixel. The third dimension refers to the number of color planes (or channels). There is one plane (2D matrix) for each RGB color, so the value of the third dimension is 3. A picture's size shows its total number of pixels, which results by multiplying the dimensions of the 2D matrices by themselves and by the number of color planes.

In [4]:
```
### 3) plot one random image from each bird class
###    modules: b_viewClasses
randomClasses = viewRandomClasses(trainPath, targetClasses[0])
plt.show()
```

```
Class: CRESTED KINGFISHER
```

```
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408

Class: POMARINE JAEGER
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408

Class: RED HEADED WOODPECKER
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408

Class: HORNED GUAN
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408

Class: CARMINE BEE-EATER
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408

Class: KAGU
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408

Class: JACOBIN PIGEON
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408

Class: VIOLET TURACO
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408

Class: SPANGLED COTINGA
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408

Class: BORNEAN LEAFBIRD
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408

Class: SPOON BILED SANDPIPER
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408

Class: CAPE MAY WARBLER
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408

Class: MALEO
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408

Class: LUCIFER HUMMINGBIRD
Image shape (rows, columns, channels): (56, 56, 3)
```

```
Image size (number of pixels): 9408

Class: CRANE HAWK
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408
```



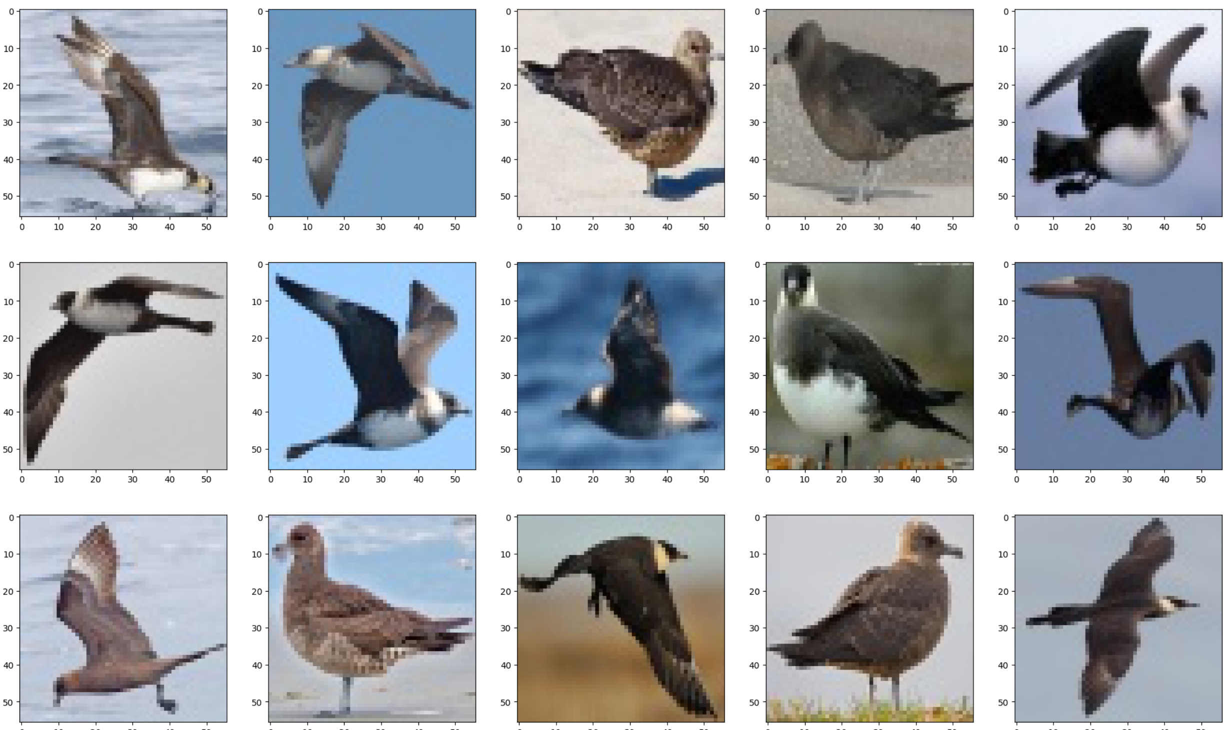
Plot 15 pictures from one of the previously selected random species

Finally, we print 15 random images from one of the previously chosen species and show their shape and size as before defined.

In [5]:
```
### 4) plot 15 random images from one of the classes
###    modules: b_viewClasses
randomClass = viewRandomClass(trainPath, targetClasses[0])
plt.show()
print('\nI conclude that, although there is a varying number of images',
      'for each bird class,')
print('in our sample of 15 classes there is a minimum',
      'of', min(targetClasses[2]), 'images, and a maximum of',
      max(targetClasses[2]), 'images.\n')
```

```
POMARINE JAEGER
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408
Image shape (rows, columns, channels): (56, 56, 3)
Image size (number of pixels): 9408
```

POMARINE JAEGER



```
I conclude that, although there is a varying number of images for each b
ird class,
in our sample of 15 classes there is a minimum of 133 images, and a maxi
mum of 204 images.
```

Data preparation

The following code allows me to load each image of the train and test sets into Python lists. The lists containing the images are then turned into numerical arrays, with 4 dimensions: the first one represents the number of images, while the other three represent the images' shape, which has been rendered homogeneous (56, 56, 3) by removing those not matching the common shape within the execution of the custom *selectData* function.

In [6]:
```
### 5) load the train, test and validation data and labels into memory
trainData, trainClasses = selectData(trainPath)
testData, testClasses = selectData(testPath)
validData, validClasses = selectData(validPath)
#train, test and validation data lists into numpy arrays
npTrainData = np.array(trainData)
npTestData = np.array(testData)
npValidData = np.array(validData)

print('Our train data array has', npTrainData.ndim, 'dimensions, and a sh
      npTrainData.shape, 'for a total number of elements of', npTrainData
print('Our test data array has', npTestData.ndim, 'dimensions, and a shap
      npTestData.shape, 'for a total number of elements of', npTestData.s
print('Our validation data array is equivalent in number to the test data
      npValidData.ndim, 'dimensions, and a shape of', npValidData.shape,
      'for a total number of elements of', npValidData.size, '.')
```

```
Our train data array has 4 dimensions, and a shape of (82724, 56, 56, 3)
for a total number of elements of 778267392 .
Our test data array has 4 dimensions, and a shape of (2575, 56, 56, 3) f
or a total number of elements of 24225600 .
Our validation data array is equivalent in number to the test data array
. It has 4 dimensions, and a shape of (2575, 56, 56, 3) for a total numb
er of elements of 24225600 .
```

The data labels represent the bird species to which the images belong. They are first converted into 2D arrays and finally turned into categorical data: bird classes are substituted by numerical categories.
Bird species should be homogeneous across train and test data, so I check if it's actually the case.

In [7]:
```
### 6) turn the train, test and validation labels into categorical arrays
# train labels
trainClasses = classesToInt(trainClasses)
npTrainClasses = np.array(trainClasses)
npTrainClasses = np.expand_dims(npTrainClasses, axis = 1) # add dimension
npTrainLabels = to_categorical(npTrainClasses)
# test labels
testClasses = classesToInt(testClasses)
npTestClasses = np.array(testClasses)
npTestClasses = np.expand_dims(npTestClasses, axis = 1) # add dimension t
npTestLabels = to_categorical(npTestClasses)
# validation labels
validClasses = classesToInt(validClasses)
npValidClasses = np.array(validClasses)
npValidClasses = np.expand_dims(npValidClasses, axis = 1) # add dimension
npValidLabels = to_categorical(npValidClasses)

trainCount = countLabels(npTrainClasses) # count train labels
testCount = countLabels(npTestClasses)   # count test labels
validCount = countLabels(npValidClasses) # count validation labels
if trainCount == testCount and trainCount == validCount:
    print('There are', npTrainClasses.size, 'categories (it means that ea
          'belongs to a category) for a set of', trainCount, 'categories.
else:
    print('ERROR: train labels count, test labels count and validation la

# free up memory
del trainData, testData, validData #trainClasses, npTrainClasses, testCla
```

```
There are 82724 categories (it means that each image belongs to a catego
ry) for a set of 515 categories.
```

Data normalization

At this point I normalize the data by dividing them for the maximum value they can assume. In this way the data range from 0 to 1 and allow for better speed and prediction results.

In [8]:
```
### 7) normalize the data
npTrainData = np.array(npTrainData / npTrainData.max(), dtype = np.float1
npTestData = np.array(npTestData / npTestData.max(), dtype = np.float16)
npValidData = np.array(npValidData / npValidData.max(), dtype = np.float1
```

Call the sequential model

I call the function executing the sequential model from the file d_sequentialModel. The function returns the model history, which is used to train the model and plot the loss function and the model accuracy.

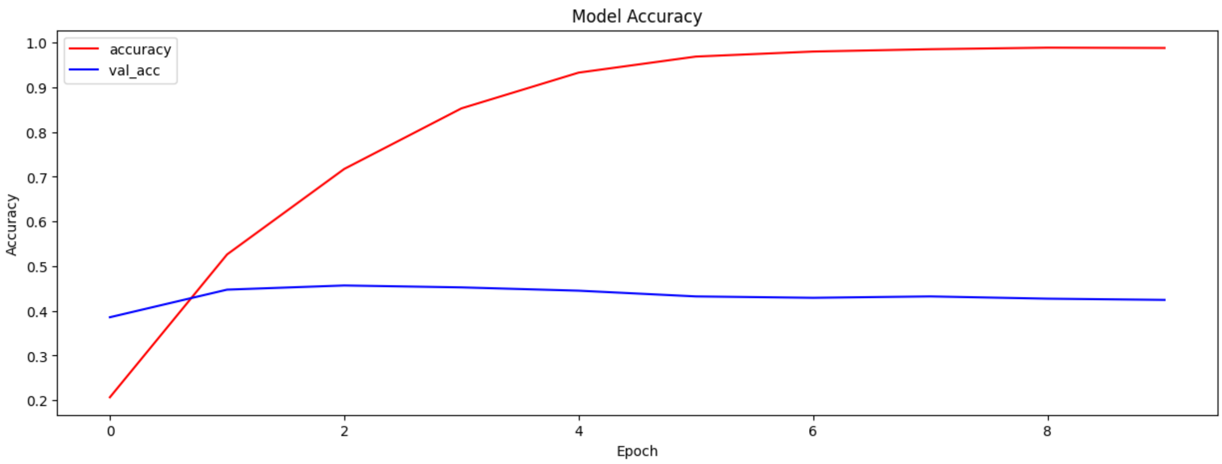
In [9]:
```
### 8) call the sequential model
myModel, Y_pred = seqModel(npTrainData, npTrainLabels, npTestData, npTest
```

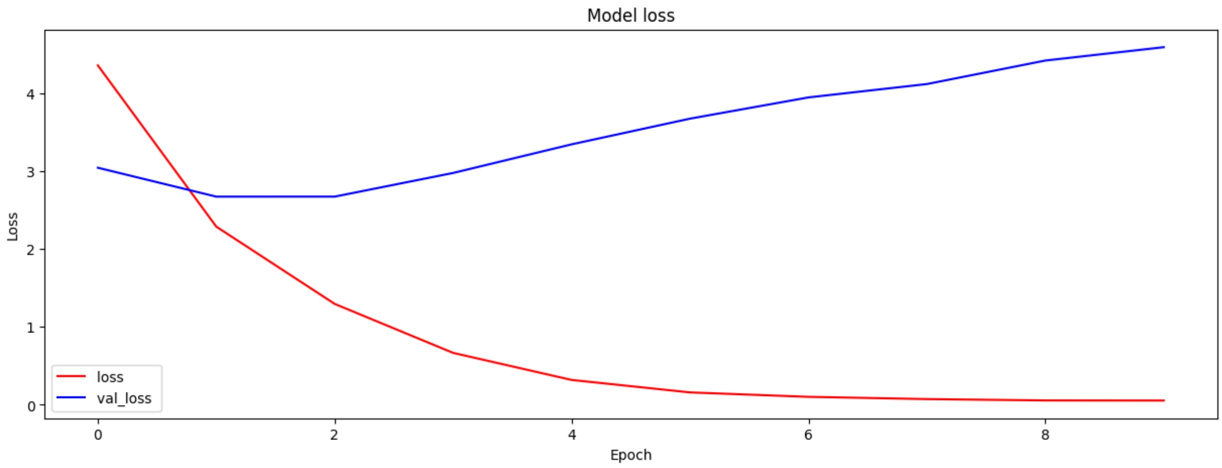
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 54, 54, 32)        896       
                                                                 
 ReLU (Activation)           (None, 54, 54, 32)        0         
                                                                 
 max_pooling2d (MaxPooling2D  (None, 18, 18, 32)       0         
 )                                                               
                                                                 
 flatten (Flatten)           (None, 10368)             0         
                                                                 
 FC2 (Dense)                 (None, 515)               5340035   
                                                                 
 Softmax (Activation)        (None, 515)               0         
                                                                 
=================================================================
Total params: 5,340,931
Trainable params: 5,340,931
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
1293/1293 [==============================] - 19s 7ms/step - loss: 4.3615
- accuracy: 0.2066 - val_loss: 3.0464 - val_accuracy: 0.3856
Epoch 2/10
1293/1293 [==============================] - 8s 6ms/step - loss: 2.2892
- accuracy: 0.5261 - val_loss: 2.6749 - val_accuracy: 0.4474
Epoch 3/10
1293/1293 [==============================] - 9s 7ms/step - loss: 1.2963
- accuracy: 0.7173 - val_loss: 2.6752 - val_accuracy: 0.4567
Epoch 4/10
1293/1293 [==============================] - 8s 6ms/step - loss: 0.6669
- accuracy: 0.8528 - val_loss: 2.9794 - val_accuracy: 0.4524
Epoch 5/10
1293/1293 [==============================] - 8s 6ms/step - loss: 0.3197
- accuracy: 0.9328 - val_loss: 3.3466 - val_accuracy: 0.4450
Epoch 6/10
1293/1293 [==============================] - 8s 6ms/step - loss: 0.1593
- accuracy: 0.9687 - val_loss: 3.6770 - val_accuracy: 0.4322
Epoch 7/10
1293/1293 [==============================] - 8s 6ms/step - loss: 0.1023
- accuracy: 0.9799 - val_loss: 3.9499 - val_accuracy: 0.4291
Epoch 8/10
1293/1293 [==============================] - 8s 6ms/step - loss: 0.0734
- accuracy: 0.9851 - val_loss: 4.1216 - val_accuracy: 0.4322
Epoch 9/10
1293/1293 [==============================] - 9s 7ms/step - loss: 0.0561
- accuracy: 0.9886 - val_loss: 4.4239 - val_accuracy: 0.4272
Epoch 10/10
```

```
1293/1293 [==============================] - 8s 6ms/step - loss: 0.0547
- accuracy: 0.9879 - val_loss: 4.5953 - val_accuracy: 0.4245

Test loss: 4.191368103027344
Test accuracy: 0.4485436975955963
81/81 [==============================] - 0s 2ms/step
```

I finally plot the loss function for the model and the model accuracy.

In [10]:
```
### 9) plot model accuracy and loss function
# accuracy
plt.figure(figsize=(15, 5))
plt.plot(myModel.history['accuracy'], 'r', label='accuracy')
plt.plot(myModel.history['val_accuracy'], 'b', label='val_acc ')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()

# loss
plt.figure(figsize=(15, 5))
plt.plot(myModel.history['loss'], 'r', label='loss ')
plt.plot(myModel.history['val_loss'], 'b', label='val_loss ')
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend()

plt.show()
```





Plot confusion matrix and classification report

I finally plot the confusion matrix and the classification report for a small sample of bird species.

In [13]:
```
### 10) Confution Matrix and Classification Report
y_pred = np.argmax(Y_pred, axis=1)
cm = confusion_matrix(testClasses, y_pred)
thresh = cm.max() / 2.
tick_marks = np.arange(len(classes))
target_names = classes
print('\nConfusion Matrix (small sample)\n')
print(cm[0 : 15, 0 : 15])
plotCM(cm[0 : 15, 0 : 15], classes[0 : 15])
print('\n\nClassification Report (small sample)\n')
class_report = classification_report(testClasses, y_pred, target_names =
print(class_report[0 : 1000], (' ' * 18) + '[...]\n') # print a few lines
```

```
Confusion Matrix (small sample)

[[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 4 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 4 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 4 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 2 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 2 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 3 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 3 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 2 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 2 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 3]]


Classification Report (small sample)

                               precision    recall  f1-score   support

       WHITE CRESTED HORNBILL       0.33      0.20      0.25         5
          GOLDEN CHLOROPHONIA       0.67      0.80      0.73         5
             EASTERN BLUEBIRD       0.67      0.80      0.73         5
              NORTHERN GANNET       0.25      0.20      0.22         5
               ANDEAN LAPWING       1.00      0.80      0.89         5
         RED HEADED WOODPECKER       0.67      0.40      0.50         5
                        MALEO       1.00      0.40      0.57         5
                 ENGGANO MYNA       0.60      0.60      0.60         5
                ALPINE CHOUGH       0.00      0.00      0.00         5
               GILDED FLICKER       0.14      0.20      0.17         5
                 CAPUCHINBIRD       1.00      0.20      0.33         5
                INDIAN ROLLER       0.75      0.60      0.67         5
                     RED KNOT       0.67      0.40      0.50         5
                        [...]
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.
py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divis
ion` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.
py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divis
ion` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.
py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divis
ion` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
<Figure size 600x600 with 0 Axes>
```