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_dataset.zip

##########################################
### 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_matrix
from    a_selectRandomFolders     import selectRandomFolders
from    b_viewClasses             import viewRandomClasses, viewRandomClass
from    c_selectData              import selectData, classesToInt, countLabels
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 names
trainPathWin = 'C:/Users/mzlarosa/OneDrive - unime.it/Learning/CdL Informatica/Anno
testPathWin = 'C:/Users/mzlarosa/OneDrive - unime.it/Learning/CdL Informatica/Anno
trainPathMac = '/Users/mau/OneDrive - unime.it/Learning/CdL Informatica/Anno II - D
testPathMac = '/Users/mau/OneDrive - unime.it/Learning/CdL Informatica/Anno II - De
validPathMac = '/Users/mau/OneDrive - unime.it/Learning/CdL Informatica/Anno II - D
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:
['SAYS PHOEBE', 'SAND MARTIN', 'SPANGLED COTINGA', 'MARABOU STORK', 'ASIAN DOLLARD
BIRD', 'COCK OF THE  ROCK', 'BIRD OF PARADISE', 'AZURE BREASTED PITTA', 'LITTLE AU
K', 'CHATTERING LORY', 'AZURE TIT', 'PUFFIN', 'TOUCHAN', 'CREAM COLORED WOODPECKE
R', 'ORIENTAL BAY OWL', 'SMITHS LONGSPUR', 'AZURE TANAGER', 'STRIPPED MANAKIN', 'T
RUMPTER SWAN', 'SPOON BILED SANDPIPER', 'ANDEAN GOOSE', 'HOUSE SPARROW', 'CAPE MAY
WARBLER', 'CALIFORNIA GULL', 'BALD EAGLE', 'AFRICAN OYSTER CATCHER', 'COPPERY TAIL
ED COUCAL', 'FIERY MINIVET', 'COMMON STARLING', 'RUDY KINGFISHER', 'ASHY STORM PET
REL', 'SNOWY EGRET', 'ALBERTS TOWHEE', 'EMU', 'BEARDED BARBET', 'CRESTED NUTHATC
H', 'VIOLET GREEN SWALLOW', 'BOBOLINK', 'ORNATE HAWK EAGLE', 'SNOWY OWL', 'BLACK S
KIMMER', 'FAIRY BLUEBIRD', 'CAMPO FLICKER', 'INCA TERN', 'GREEN BROADBILL', 'ENGGA
NO MYNA', 'PARADISE TANAGER', 'NORTHERN JACANA', 'RED WISKERED BULBUL', 'WOODLAND
KINGFISHER'][...]
```

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 [4]:
```
### 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 184 image files in /content/reshaped/train/SPOTTED WHISTLI
NG DUCK
There are 0 folders and 156 image files in /content/reshaped/train/BLACK VULTURE
There are 0 folders and 182 image files in /content/reshaped/train/VERDIN
There are 0 folders and 154 image files in /content/reshaped/train/PHILIPPINE EAGL
E
There are 0 folders and 155 image files in /content/reshaped/train/AFRICAN OYSTER
CATCHER
There are 0 folders and 163 image files in /content/reshaped/train/VISAYAN HORNBIL
L
There are 0 folders and 140 image files in /content/reshaped/train/RAINBOW LORIKEE
T
There are 0 folders and 159 image files in /content/reshaped/train/RED NAPED TROGO
N
There are 0 folders and 139 image files in /content/reshaped/train/DEMOISELLE CRAN
E
There are 0 folders and 143 image files in /content/reshaped/train/KOOKABURRA
There are 0 folders and 162 image files in /content/reshaped/train/RED WISKERED BU
LBUL
There are 0 folders and 149 image files in /content/reshaped/train/HORNED SUNGEM
There are 0 folders and 208 image files in /content/reshaped/train/MILITARY MACAW
There are 0 folders and 137 image files in /content/reshaped/train/NORTHERN FULMAR
There are 0 folders and 169 image files in /content/reshaped/train/RED LEGGED HONE
YCREEPER
```

Plot pictures from sample species

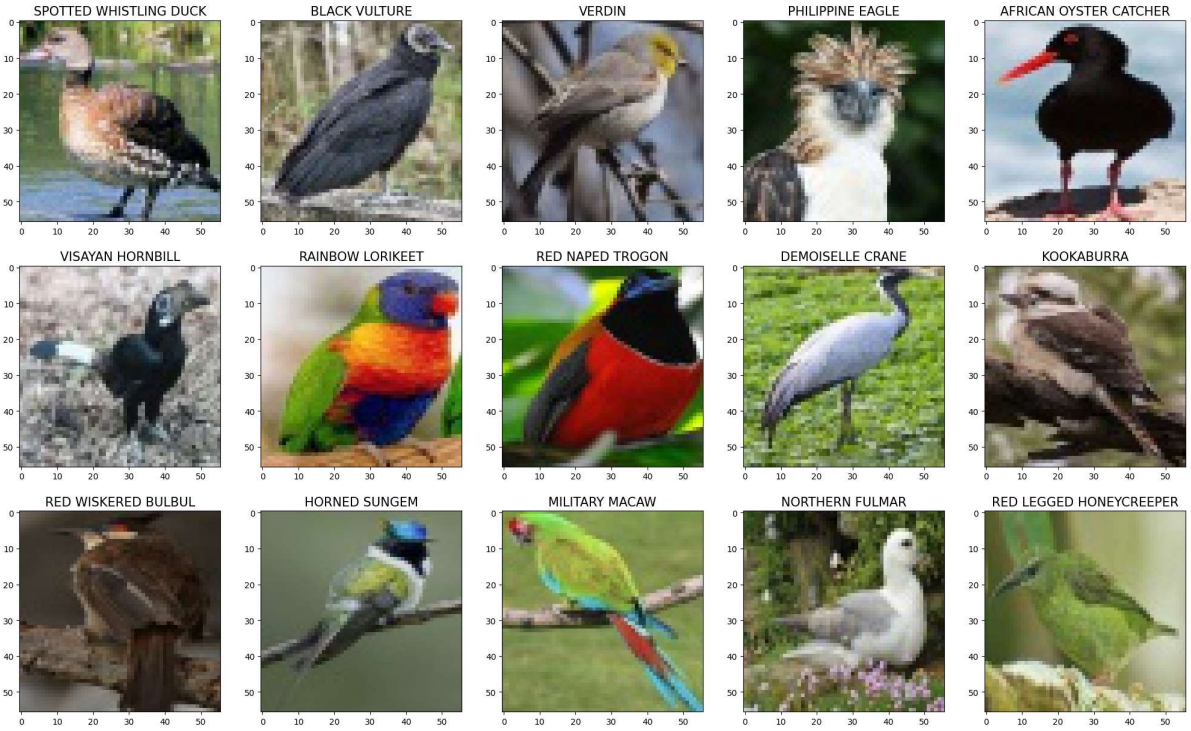
Plot one picture from 15 randomly selected species

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 [5]:
```
### 3) plot one random image from each bird class
###    modules: b_viewClasses
randomClasses = viewRandomClasses(trainPath, targetClasses[0])
plt.show()
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Sample of 15 bird classes

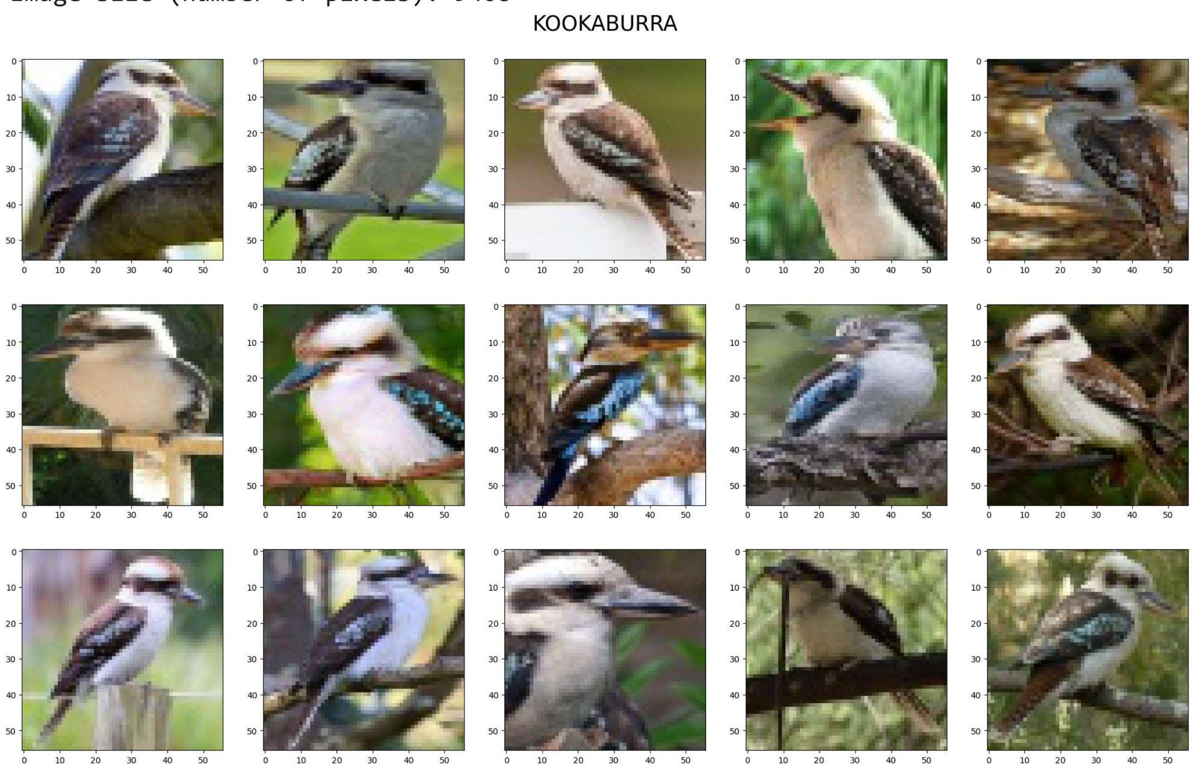


Plot 15 pictures from one of the previously selected random species

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

In [6]:
```
### 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')
```

```
KOOKABURRA
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
```




```
I conclude that, although there is a varying number of images for each bird class,
in our sample of 15 classes there is a minimum of 137 images, and a maximum of 208
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 [3]:
```
### 5) load the train and test data and labels into memory
trainData, trainClasses = selectData(trainPath)
testData, testClasses = selectData(testPath)
#train data list into numpy array
npTrainData = np.array(trainData)
#test data list into numpy array
npTestData = np.array(testData)

print('Our train data array has', npTrainData.ndim, 'dimensions, and a shape of',
      npTrainData.shape, 'for a total number of elements of', npTrainData.size, '.
print('Our test data array has', npTestData.ndim, 'dimensions, and a shape of',
      npTestData.shape, 'for a total number of elements of', npTestData.size, '.')
```

```
Our train data array has 4 dimensions, and a shape of (82724, 56, 56, 3) for a tot
al number of elements of 778267392 .
Our test data array has 4 dimensions, and a shape of (2575, 56, 56, 3) for a total
number 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 [4]:
```
### 6) turn the train and test labels into categorical arrays
# train labels

trainClasses = classesToInt(trainClasses)
npTrainClasses = np.array(trainClasses)
npTrainClasses = np.expand_dims(npTrainClasses, axis = 1) # add dimension to array
npTrainLabels = to_categorical(npTrainClasses)
# test labels
testClasses = classesToInt(testClasses)
npTestClasses = np.array(testClasses)
npTestClasses = np.expand_dims(npTestClasses, axis = 1) # add dimension to array
npTestLabels = to_categorical(npTestClasses)

trainCount = countLabels(npTrainClasses) # count train labels
testCount = countLabels(npTestClasses) # count test labels
if trainCount == testCount:
    print('There are', npTrainClasses.size, 'categories (it means that each image'
          'belongs to a category) for a set of', trainCount, 'categories.')
else:
    print('ERROR: train labels count and test labels count don\'t match.')

# free up memory
del trainData, testData, trainClasses, npTrainClasses, testClasses, npTestClasses
```

```
There are 82724 categories (it means that each image belongs to a category) 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 [5]:
```
### 7) normalize the data
npTrainData = np.array(npTrainData / npTrainData.max(), dtype = np.float16)
npTestData = np.array(npTestData / npTestData.max(), dtype = np.float16)
```

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.

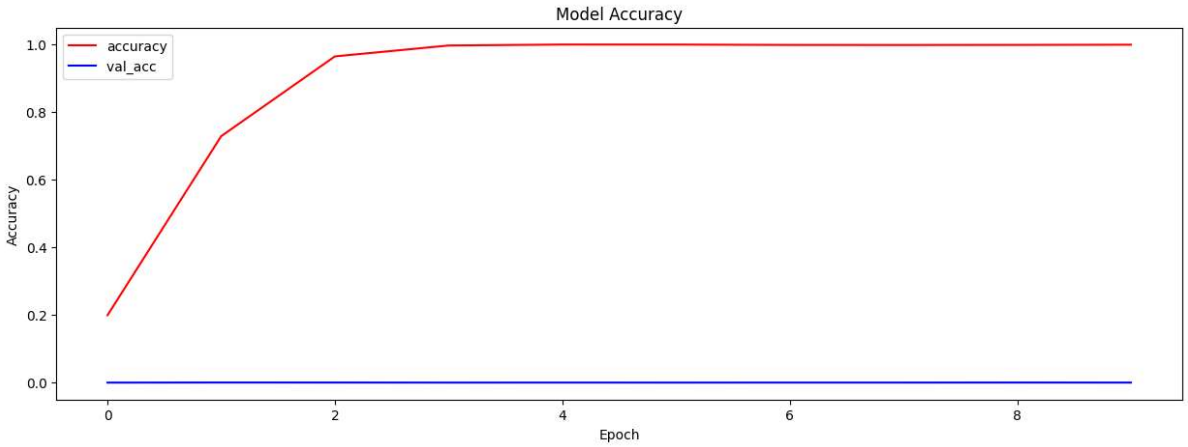
In [6]:
```
### 8) call the sequential model
batch_size = 64 # batch size to be used in model.fit
test_datagen = ImageDataGenerator(rescale = 1. / 255)
num_of_test_samples = npTestData.shape[0] # number of images in test folder

validation_generator = test_datagen.flow_from_directory(testPath,
                                                        target_size=(56, 56),
                                                        batch_size = 64,
                                                        class_mode = 'categorical'

myModel, Y_pred = seqModel(npTrainData, npTrainLabels, npTestData, npTestLabels, ba
```

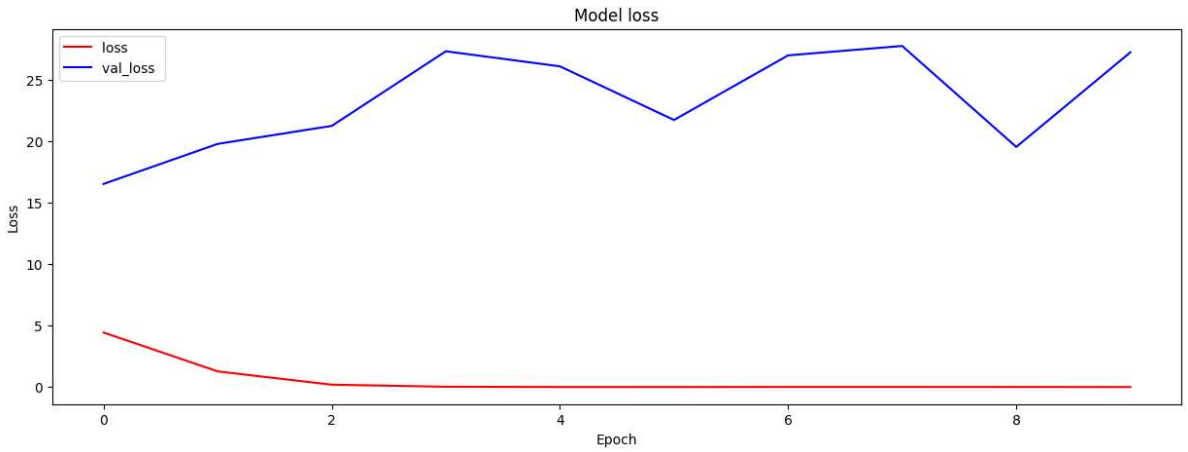
```
Found 2575 images belonging to 515 classes.
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 54, 54, 32)        896

 ReLU (Activation)           (None, 54, 54, 32)        0

 flatten (Flatten)           (None, 93312)             0

 FC2 (Dense)                 (None, 515)               48056195

 Softmax (Activation)        (None, 515)               0

=================================================================
Total params: 48,057,091
Trainable params: 48,057,091
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
1035/1035 [==============================] - 31s 20ms/step - loss: 4.4389 - accura
cy: 0.1992 - val_loss: 16.5208 - val_accuracy: 1.2088e-04
Epoch 2/10
1035/1035 [==============================] - 18s 18ms/step - loss: 1.2770 - accura
cy: 0.7285 - val_loss: 19.7740 - val_accuracy: 4.2309e-04
Epoch 3/10
1035/1035 [==============================] - 18s 18ms/step - loss: 0.1975 - accura
cy: 0.9642 - val_loss: 21.2366 - val_accuracy: 3.6265e-04
Epoch 4/10
1035/1035 [==============================] - 20s 20ms/step - loss: 0.0300 - accura
cy: 0.9963 - val_loss: 27.3042 - val_accuracy: 1.8132e-04
Epoch 5/10
1035/1035 [==============================] - 18s 18ms/step - loss: 0.0074 - accura
cy: 0.9992 - val_loss: 26.0787 - val_accuracy: 2.4176e-04
Epoch 6/10
1035/1035 [==============================] - 18s 18ms/step - loss: 0.0065 - accura
cy: 0.9991 - val_loss: 21.7170 - val_accuracy: 1.8132e-04
Epoch 7/10
1035/1035 [==============================] - 20s 19ms/step - loss: 0.0116 - accura
cy: 0.9980 - val_loss: 26.9711 - val_accuracy: 2.4176e-04
Epoch 8/10
1035/1035 [==============================] - 18s 18ms/step - loss: 0.0117 - accura
cy: 0.9978 - val_loss: 27.7351 - val_accuracy: 2.4176e-04
Epoch 9/10
1035/1035 [==============================] - 18s 18ms/step - loss: 0.0099 - accura
cy: 0.9981 - val_loss: 19.5271 - val_accuracy: 3.6265e-04
Epoch 10/10
1035/1035 [==============================] - 18s 18ms/step - loss: 0.0066 - accura
cy: 0.9987 - val_loss: 27.2174 - val_accuracy: 2.4176e-04

Test loss: 9.90447998046875
Test accuracy: 0.26213592290878296
41/41 [==============================] - 1s 28ms/step
```

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

In [7]:
```python
### 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')
```

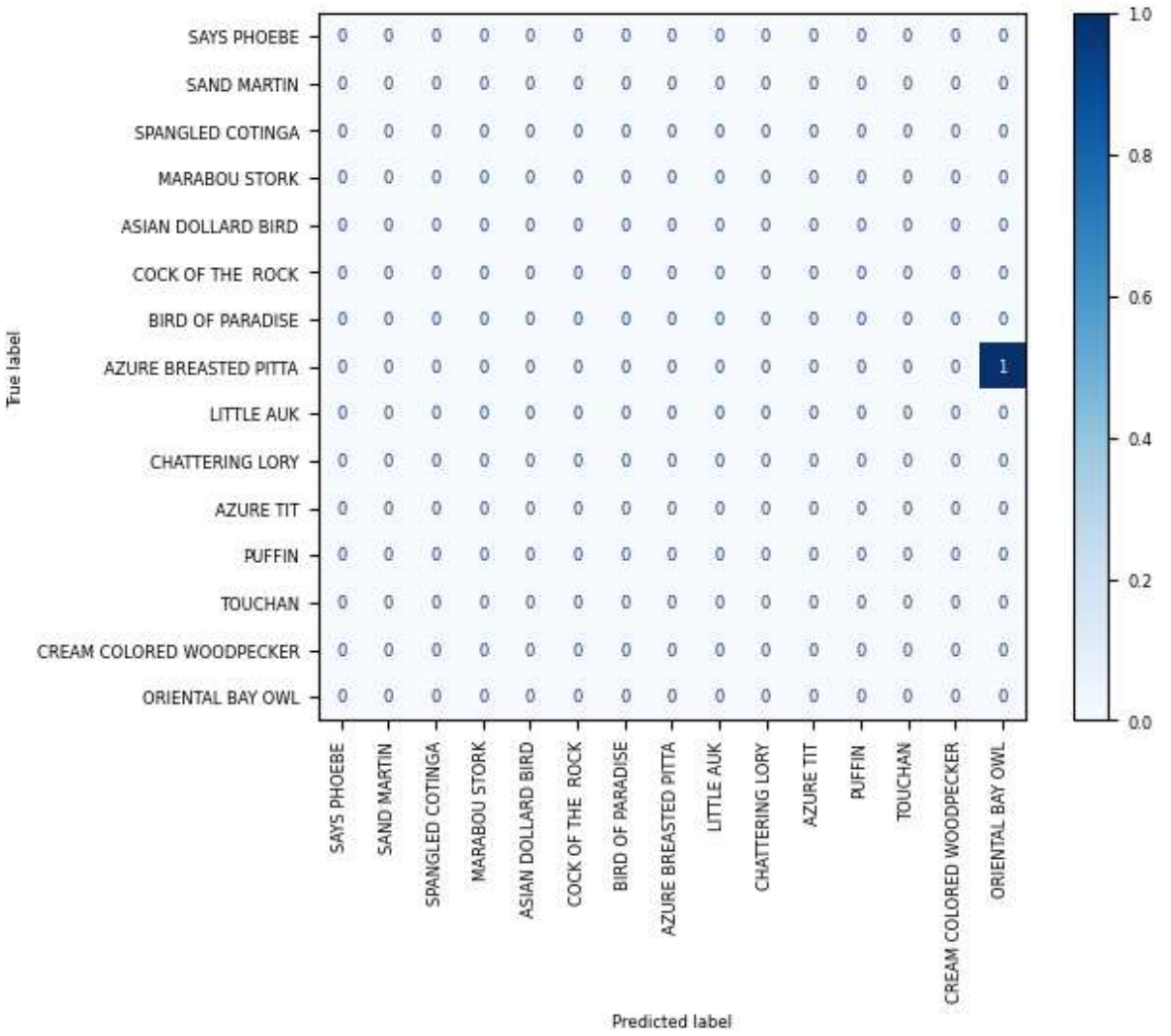
```
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()

# 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()
```





In [8]:
```
### 10) Confution Matrix and Classification Report
y_pred = np.argmax(Y_pred, axis=1)
cm = confusion_matrix(validation_generator.classes, 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(validation_generator.classes, y_pred, target_
print(class_report[0 : 1000], (' ' * 18) + '[...]\n') # print a few lines of the c
```

```
Confusion Matrix (small sample)

[[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 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 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 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]
 [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 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 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]]


Classification Report (small sample)

                               precision    recall  f1-score   support

                  SAYS PHOEBE       0.00      0.00      0.00         5
                  SAND MARTIN       0.00      0.00      0.00         5
             SPANGLED COTINGA       0.00      0.00      0.00         5
                MARABOU STORK       0.00      0.00      0.00         5
            ASIAN DOLLARD BIRD      0.00      0.00      0.00         5
              COCK OF THE  ROCK     0.00      0.00      0.00         5
             BIRD OF PARADISE       0.00      0.00      0.00         5
         AZURE BREASTED PITTA       0.00      0.00      0.00         5
                   LITTLE AUK       0.00      0.00      0.00         5
              CHATTERING LORY       0.00      0.00      0.00         5
                    AZURE TIT       0.00      0.00      0.00         5
                       PUFFIN       0.00      0.00      0.00         5
                      TOUCHAN       0.00      0.00      0.00         5
                        [...]

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` 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: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` 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: U
ndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
<Figure size 600x600 with 0 Axes>
```


SAYS PHOEBE
SAND MARTIN
SPANGLED COTINGA
MARABOU STORK
ASIAN DOLLARD BIRD
COCK OF THE ROCK
BIRD OF PARADISE
AZURE BREASTED PITTA
LITTLE AUK
CHATTERING LORY
AZURE TIT
PUFFIN
TOUCHAN
CREAM COLORED WOODPECKER
ORIENTAL BAY OWL
True label
Predicted label
1.0
0.8
0.6
0.4
0.2
0.0

In [ ]: