INSTITUTE FOR ADVANCED COMPUTING

AND

SOFTWARE DEVELOPMENT

AKURDI, PUNE

Documentation On

“Bird Species Classification using Deep Learning”

e-DBDA MAY 2021

Submitted By:

Group No: 16

Ankit S. Gawande- 1306

Rutuja D. Kardile- 1322

Mr. Prashant Karhale

Centre Coordinator

Mr. Akshay Tilekar

Project Guide

INDEX

Abstract

The recognition of bird species had a crucial role in identifying the belongingness of the bird to a certain species. Bird species travel throughout the world and we find wide variety of bird species in our day-today life. Each Bird species has different size, color, shape in different scenarios. The main aim of the proposed work is to develop an automated model which has capability of identifying the species of the bird where bird image is given as a test image from the dataset.\

We believe images show good variations to identify the bird species. A simple image recognition classifier has been created. This image recognition tool classifies various species of birds. An application CNN has been used in order to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data. Then application of some of the supervised algorithms is used to check and compare their accuracies against each other. In deep learning, convolutional neural network (CNN) is a class of deep neural network mostly used for analyzing visual images. Our project will be helpful in identifying the endangered species and help society in spreading awareness about the need of all the species for balance in the nature.

Chapter 1

Introduction

1.1 Image Processing:

Image Processing is a very powerful technique used today to convert an image into a form which is either digital or analog so that it can be used to extract some important and useful data. This process takes a raw image as an input and processes it and gives an improved modified image or characteristics associated with that image as an output. Image processing when used in ML algorithms such as CNN can be used to get very interesting results such as image recognition or creating a model to predict some feature from image. Mainly these three processes are involved in image processing. Fetching the required image by using any available tool. The fetched image is then analyzed and some necessary manipulation is done on it to find some significant patterns in it which are not visible to a naked human eye. The last stage is the output stage where the output is either an image or a report based.

1.2 Convolutional Neural Networks (CNN):

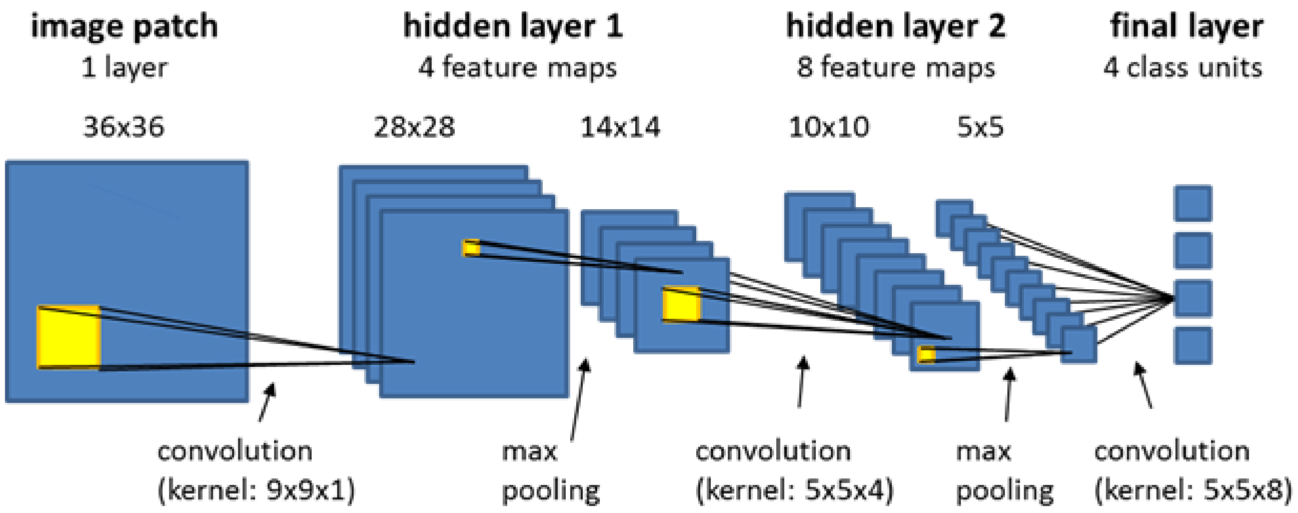


Figure 1: Convolutional Neural Network Architecture

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in Figure 1 a CNN classification example Dog Breed Prediction Using DL IACSD Akurdi, Pune 4 primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlaps to cover the entire visual area.

The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network, which has the wholesome understanding of images in the dataset, similar to how we would

1.2 Problem Statement

Build a model using deep learning algorithms and image processing to predict the species of 275 types of birds by taking their image as input. Use transfer learning to build a model that recognizes up to 275 different bird species.

1.3 Motivation:

This problem is not only challenging but also its solution is applicable to other fine-grained classification problems. For example, the methods used to solve this problem would also help identify breeds of cats and horses as well as breeds of dogs and plants or even models of cars. Any set of classes with relatively small variation within it can be solved as a fine-grained classification problem. In the real-world, an identifier like this could be used in bio-diversity studies, helping scientists save time and resources when conducting studies about the health and abundance of certain species populations. Ultimately, we found birds to be the most interesting class to experiment with due to their immense diversity, beautiful and abundance in photographs, but we also hope to expand our understanding of the fine-grained classification problem and provide a useful tool for scientists across disciplines.

1.4 Scope of the project:

1.4.1 Initial functional requirement will be:

• To build deep learning algorithms and predict the species of birds.

• To train the model on 275 classes of birds on over 39364 images.

• To use Transfer Learning to do feature extraction of training data and build a CNN model over it to predict species.

• Use Python, Keras and image processing tools to filter our data and cleaning of our input data.

1.4.2 Initial nonfunctional requirement will be:

• There is training set and a test set of images of birds. Each directory has a filename that is its name of birds. Each directory contains images of their respective class.

• train - the training set, there are images of birds. There are about 39364 images in the training data.

• validation – In the validation set, there are about 1375 images in the training data.

Chapter 2

Literature Review

2.1 What is Machine Learning?

Machine learning is a field of Artificial Intelligence, it is an approach that is based on the idea that machines can be given access to data along with the ability to learn from it. Machine can find patterns in the data and predict according to that. The data provided is divided to training and testing data and the accuracy of the algorithm is calculated on the basis of test score it obtains. Machine Learning is mainly divided into 3 categories:

a. **Supervised Machine Learning:**

Supervised Learning is a type of machine learning used to train models from labeled training data. It allows you to predict output for future or unseen data.
In supervised learning, algorithm is selected based on target variable
Types Of Supervised learning—

1. Classification—
If target variable is categorical (classes), then use classification algorithm.
classification is applied when the output has finite and discrete values.
2. Regression—
If target variable is a continuous numeric variable (100–2000), then use a regression algorithm.

b. **Unsupervised Machine Learning:**

This kind of algorithm is used when the input data is not classified as well as not labeled. These algorithms identify the patterns in the provided data and draw inferences from the data sets to identify the data which is unstructured and unlabeled. However, these algorithms are not capable of predicting very accurate results but still it provides quite efficient results base on the kind of unstructured data provided. These algorithms are more complex and difficult to understand than the supervised algorithms.

c. **Reinforcement Machine learning:**

Reinforcement learning is an area of machine learning it is generally depending on users' feedback how user can react on specific area or product. It recommends user upon their previous activity. in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it is bound to learn from its experience.

2.2 Transfer Learning:

Transfer learning generally refers to a process where a model trained on one problem is used in some way on a second related problem. In deep learning, transfer learning is a technique whereby a neural network model is first trained on a problem similar to the problem that is being solved. One or more layers from the trained model are then used in a new model trained on the problem of interest. Transfer learning has the benefit of decreasing the training time for a neural network model and can result in lower generalization error. The weights in re-used layers may be used as the starting point for the training process and adapted in response to the new problem. This usage treats transfer learning as a type of weight initialization scheme. This may be useful when the first related problem has a lot more labeled data than the problem of interest and the similarity in the structure of the problem may be useful in both contexts.

CHAPTER 3

SYSTEM DEVELOPMENT

3.1 System Environment:

System environment involves a process of designing a software prototype, testing it, converting it to complete model and again applying various testing algorithms to it and finally create the whole software.

3.2 Functional requirement specification:

Use Cases:

• To Use Transfer Learning for feature Extraction.

• To build deep learning algorithms and predict the species of bird.

Brief Description:

The neural network then identifies the patterns in that matrix to remember that image so that it can later itself recognize that image. This neural network will take the value of the pixels as a feature on which the ML algorithm will work to predict the output. It is almost impossible for a human eye to recognize the pattern which this network learns to identify that image. We can provide different weights and bias to the model to alter our results.

Initial Step by Step Description:

• First, we need to import the images then perform filtering and data augmentation.

• We are going to do feature extraction using pretrained models.

• Then we are going to train a deep learning model based on these extracted features for better accuracy.

3.3 Non-functional requirement:

Python libraries such as os, TensorFlow, scikit-learn, Keras, torchvision, PIL.

CHAPTER 4

Requirement Specification

4.1 External requirement specification:

The only link to an external system is the link to the model with JavaScript, html and CSS with the help of Django Framework.

4.2 Detailed Requirements:

4.2.1 Functional Requirement:

First of all, model should be successfully trained by developer.

4.2.2 Hardware Requirement:

Processor: Intel Dual Core (minimum), GPU

RAM: Minimum 4 GB

OS: Windows, Linux

4.2.3 Software Requirement:

Installing Anaconda on Windows10:

Download and install Anaconda (windows version).

• Select the default options when prompted during the installation of Anaconda.

• After you finished installing, open Anaconda Prompt. Type the command below to see that you can use a Jupyter (IPython) Notebook.

• If you didn't check the add Anaconda to path argument during the installation process, you will have to add python and conda to your environment variables.

• You know you need to do so if you open a command prompt (not anaconda prompt) and get the following messages.

• This step gives two options for adding python and conda to your path. If you don't know where your conda and/or python is, you type the following commands into your anaconda prompt.

Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Dog Breed Prediction Using DL IACSD Akurdi, Pune 11 Windows, macOS and Linux. To get Navigator, get the Navigator cheat sheet and install Anaconda.

The Navigator Getting started with Navigator section shows how to start Navigator from the shortcuts or from a terminal window.

Create a new conda environment and install following libraries of python:

Sklearn – Comes with Anaconda

To create a new conda environment – conda create -n my_TF2 --clone base

Upgrade your pip - pip install --upgrade pip

OS – pip install os

TensorFlow – conda install tensorflow

OpenCV2 - conda install -c conda-forge opencv

Keras - conda install -c conda-forge keras (Optional)

OpenCV is a computer vision and machine learning software library. Keras is an Deep Learning library for Python. It contains pretrained models also. TensorFlow 2.0 focus on simplicity and ease of use, featuring updates like:

• Easy model building with Keras and eager execution.

• Robust model deployment in production on any platform.

• Powerful experimentation for research.

• Simplifying the API by cleaning up deprecated APIs and reducing duplication.

When you install TensorFlow, Keras comes with it automatically. Keras is an extremely popular high-level API for building and training deep learning models.

CHAPTER 5

SYSTEM DESIGN

5.1 Flowchart of the System:

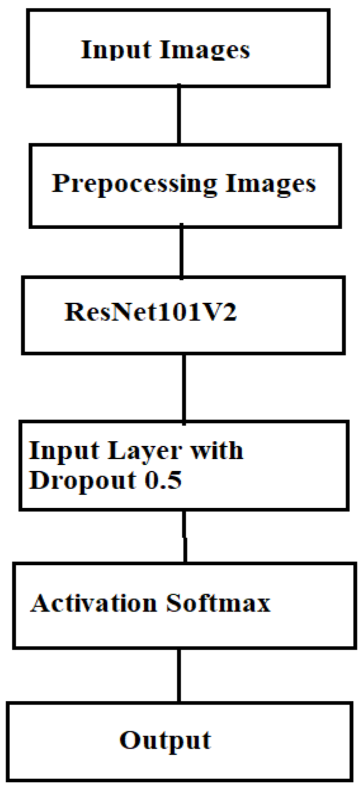


Figure 2: Flowchart of project

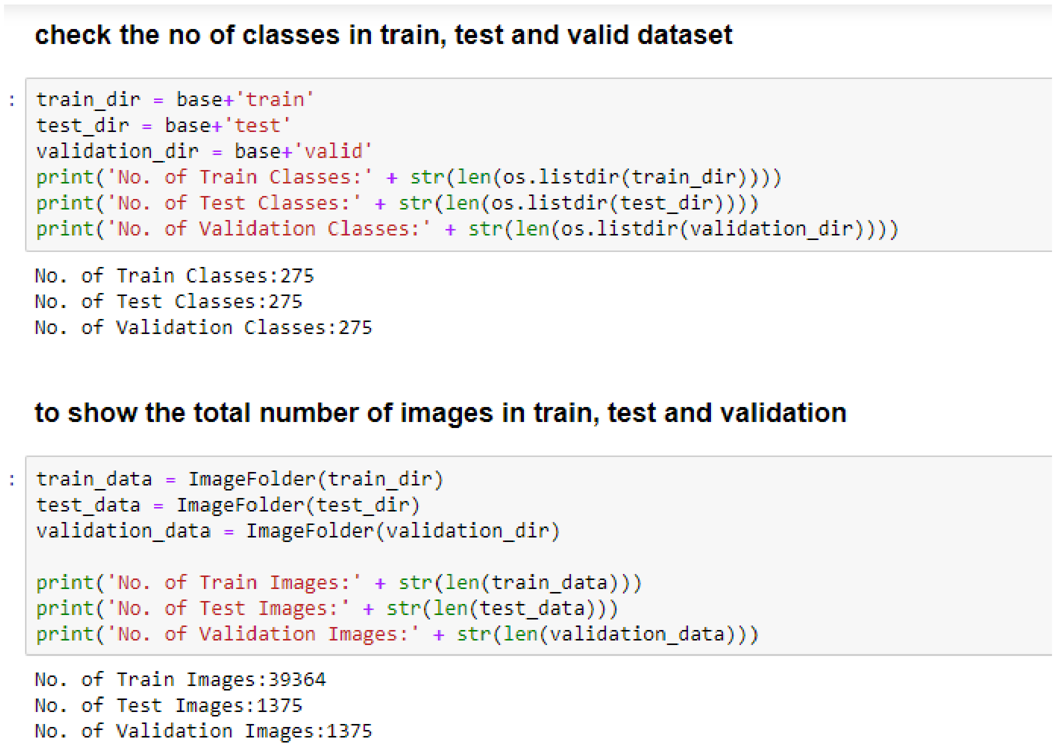
Steps involved in flow-chart:

1. The dataset is imported from the web and is broken into three parts i.e., training, validation and testing.
2. Reading all Bird images from training dataset.
3. We have 275 species so we have to use label encoding and set the position of the data.
4. We used ResNet101V2 model of keras application for processing data and used imagenet.
5. We used ResNet model and provide them as a input to our classifier Deep Learning model.
6. Now based on the training dataset and validation dataset the model is trained and validated. The weights are adjusted based on back propagation
7. Now, we plug in any image of the bird and check for the accuracy.

CHAPTER 6

MODEL BUILDING

6.1 Data Preprocessing:


check the no of classes in train, test and valid dataset

```
train_dir = base+'train'
test_dir = base+'test'
validation_dir = base+'valid'
print('No. of Train Classes:' + str(len(os.listdir(train_dir))))
print('No. of Test Classes:' + str(len(os.listdir(test_dir))))
print('No. of Validation Classes:' + str(len(os.listdir(validation_dir))))
```

```
No. of Train Classes:275
No. of Test Classes:275
No. of Validation Classes:275
```

to show the total number of images in train, test and validation

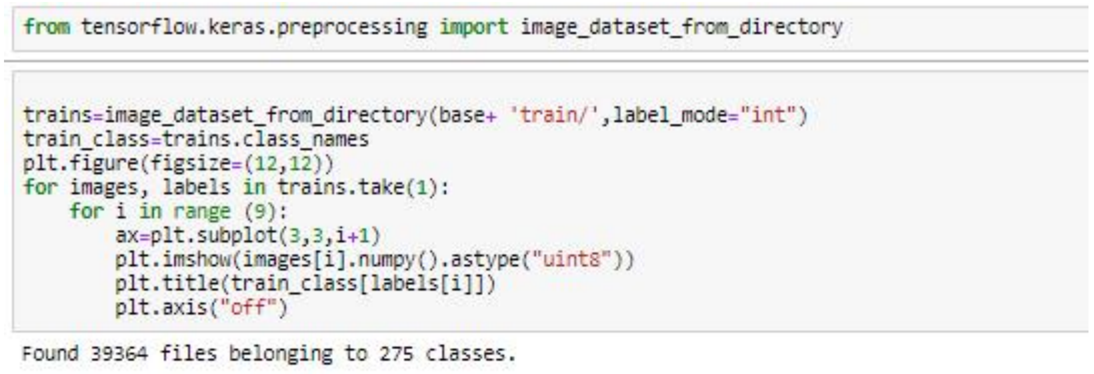
```
train_data = ImageFolder(train_dir)
test_data = ImageFolder(test_dir)
validation_data = ImageFolder(validation_dir)

print('No. of Train Images:' + str(len(train_data)))
print('No. of Test Images:' + str(len(test_data)))
print('No. of Validation Images:' + str(len(validation_data)))
```

```
No. of Train Images:39364
No. of Test Images:1375
No. of Validation Images:1375
```

Figure 3: Data preprocessing

The datasets consist of folders of train test and validation data, each datasets contains no of folders naming as their species name. Folder contains no of images of birds so for reading that image for specific folder. It is somehow critical to read all images for each folder so we use library for reading those data. For understanding data, we use image folder from torchvision library to check no of classes in the data and then check total number of images in each dataset. There are 275 types of classes present in each train test and validation datasets.

Total No of images in train datasets are 39364 and 1375 no of images in test and validation datasets.



```
from tensorflow.keras.preprocessing import image_dataset_from_directory
```

```
trains=image_dataset_from_directory(base+ 'train/',label_mode="int")
train_class=trains.class_names
plt.figure(figsize=(12,12))
for images, labels in trains.take(1):
    for i in range (9):
        ax=plt.subplot(3,3,i+1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(train_class[labels[i]])
        plt.axis("off")
```

```
Found 39364 files belonging to 275 classes.
```

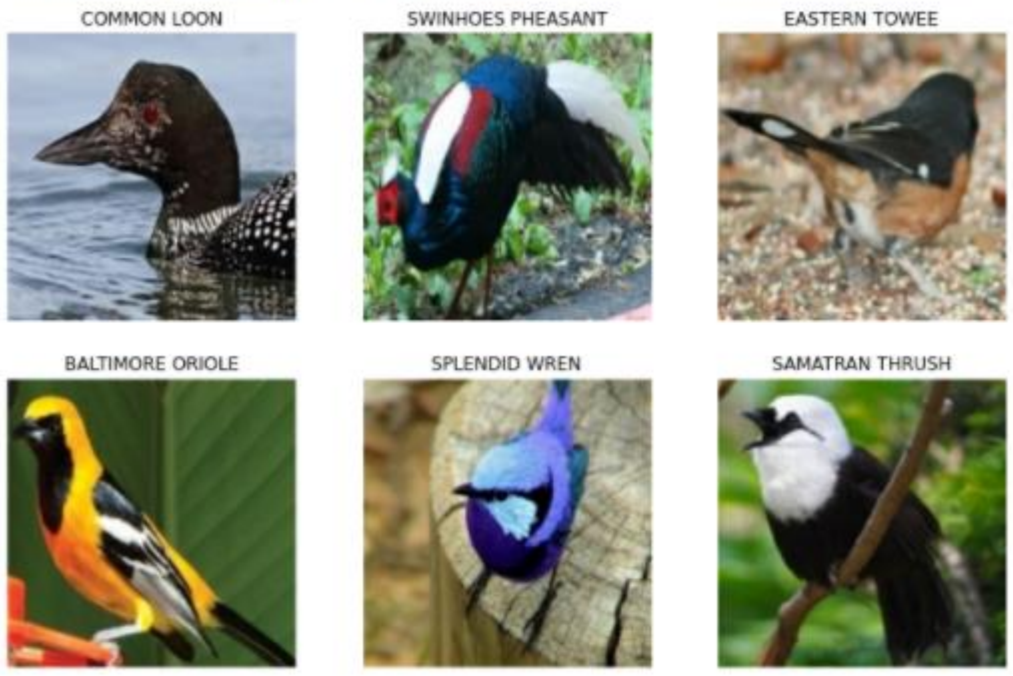
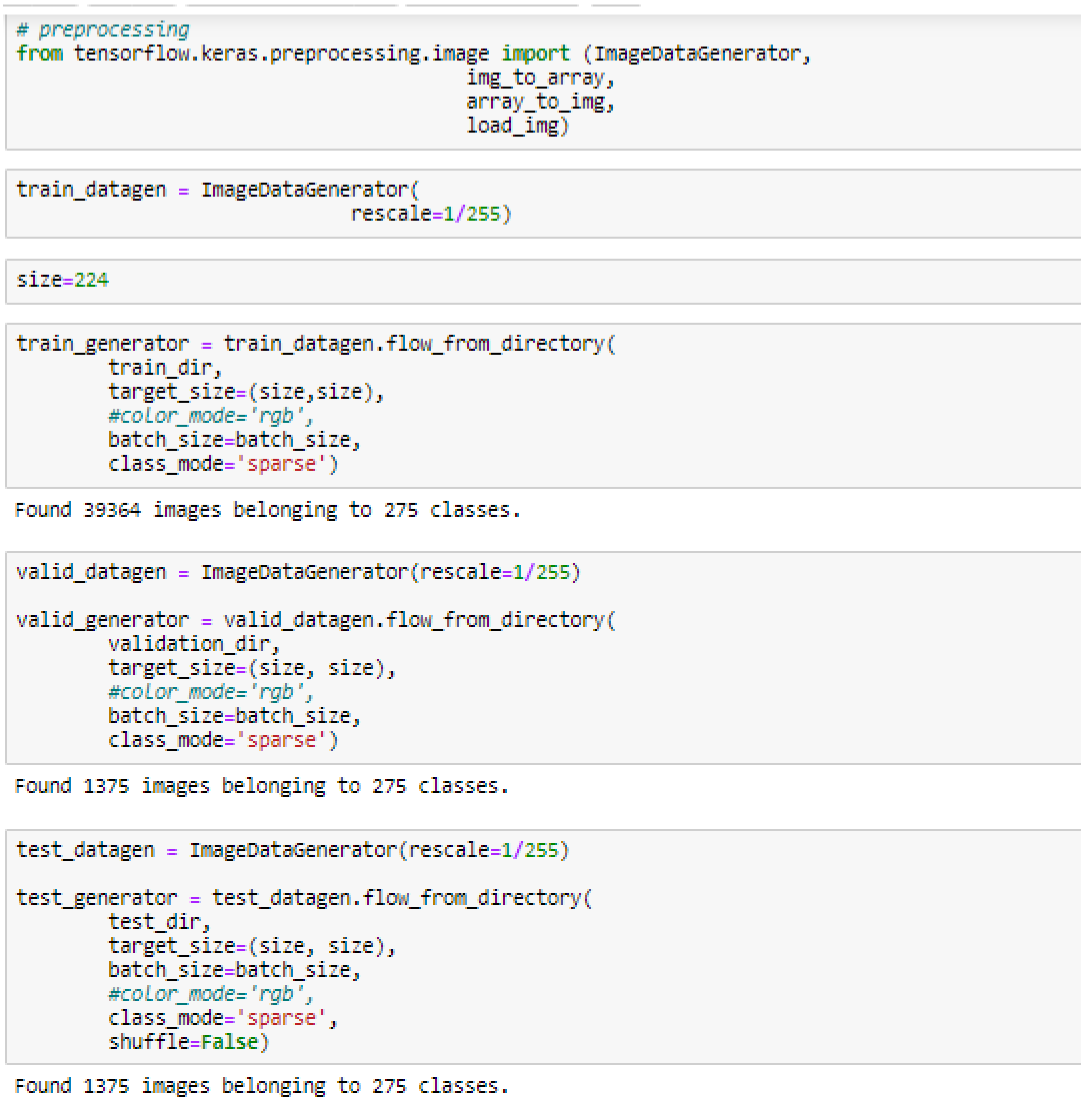


Figure 4: Bird Images

Here we are showing some bird images from training datasets. For showing images we use keras preprocessing library. It takes any random images from any classes. For visualize the images we use matplotlib library. It is showing their images and name of that respective species. We consider fig size as 12x12.



```
# preprocessing
from tensorflow.keras.preprocessing.image import (ImageDataGenerator,
                                                  img_to_array,
                                                  array_to_img,
                                                  load_img)
```

```
train_datagen = ImageDataGenerator(
                                   rescale=1/255)
```

```
size=224
```

```
train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(size,size),
        #color_mode='rgb',
        batch_size=batch_size,
        class_mode='sparse')
```

```
Found 39364 images belonging to 275 classes.
```

```
valid_datagen = ImageDataGenerator(rescale=1/255)

valid_generator = valid_datagen.flow_from_directory(
        validation_dir,
        target_size=(size, size),
        #color_mode='rgb',
        batch_size=batch_size,
        class_mode='sparse')
```

```
Found 1375 images belonging to 275 classes.
```

```
test_datagen = ImageDataGenerator(rescale=1/255)

test_generator = test_datagen.flow_from_directory(
        test_dir,
        target_size=(size, size),
        batch_size=batch_size,
        #color_mode='rgb',
        class_mode='sparse',
        shuffle=False)
```

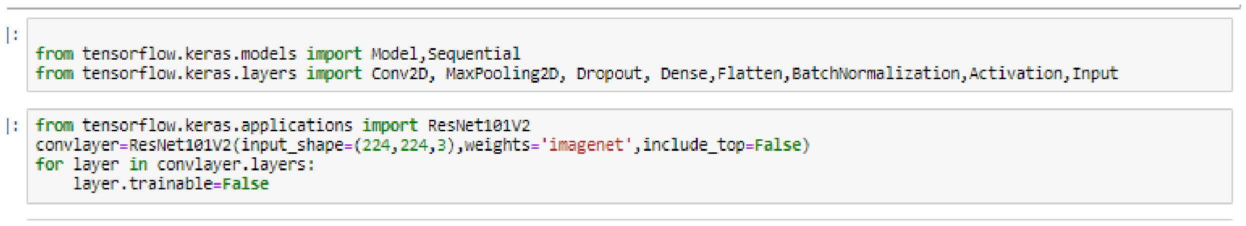
```
Found 1375 images belonging to 275 classes.
```

Figure 5: Data Augmentation

Description: The ImageDataGenerator class supports a number of pixel scaling methods, as well as a range of data augmentation techniques. We will focus on the pixel scaling techniques. The ImageDataGenerator class can be used to rescale pixel values from the range of 0-255 to the range 0-1 preferred for neural network models. Here we normalize the size of image.so it can be easy for further processing of the data. Rescale is a value by which we will multiply the data before any other processing. Our original images consist in RGB coefficients in the 0-255, but such values would be too high for our model to process (given a typical learning rate), so we target values between 0 and 1. We used a batch size of 32. This means that each of the train and test datasets of images are divided into groups of 32 images that will then be scaled when returned from the iterator. Batch size is simply No. of images to be yielded from the generator per batch.

We used class_mode as a sparse. This means 'sparse returns the output formatted in the in the way that’s required if your using 'spare_categorical_crossentropy' as a loss. In other words, it should return data that's been encoded as a 1-dimensional vector with integer labels [0,1,2,3, 4....n] where n is the number of classes, similar to the output from scikit-learn's LabelEncoder. We used shuffle as a False so it can sort the data in alphanumeric order. We applied all these arguments to train, test and validation data.

6.2 Data Modeling:



```
[ ]: from tensorflow.keras.models import Model,Sequential
     from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dropout, Dense,Flatten,BatchNormalization,Activation,Input
```

```
[ ]: from tensorflow.keras.applications import ResNet101V2
     convlayer=ResNet101V2(input_shape=(224,224,3),weights='imagenet',include_top=False)
     for layer in convlayer.layers:
         layer.trainable=False
```

Figure 6: Resnet Model

Description: We import ResNet101V2 from keras. application library for further analysis. ResNet is Residual Neural network which already implement in keras and we used that model for our analysis. It is called as residual because it skips the training of few layers using skip-connections or residual connections

In traditional neural networks, each layer feeds into the next layer. In a network with residual blocks, each layer feeds into the next layer and directly into the layers about 2–3 hops away. The skip connections in ResNet solve the problem of vanishing gradient in deep neural networks by allowing this alternate shortcut path for the gradient. You can skip the training of few layers using skip connections or residual connections. This is what we see in the image above. In fact, if you look closely, we can directly learn an identity function by relying on skip connections only. This is the exact reason why skip connections are also called identity shortcut connections too.

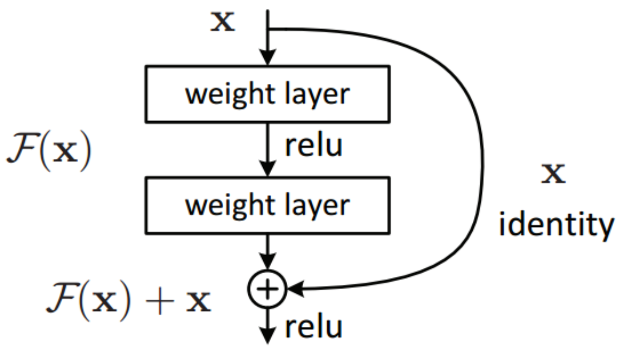
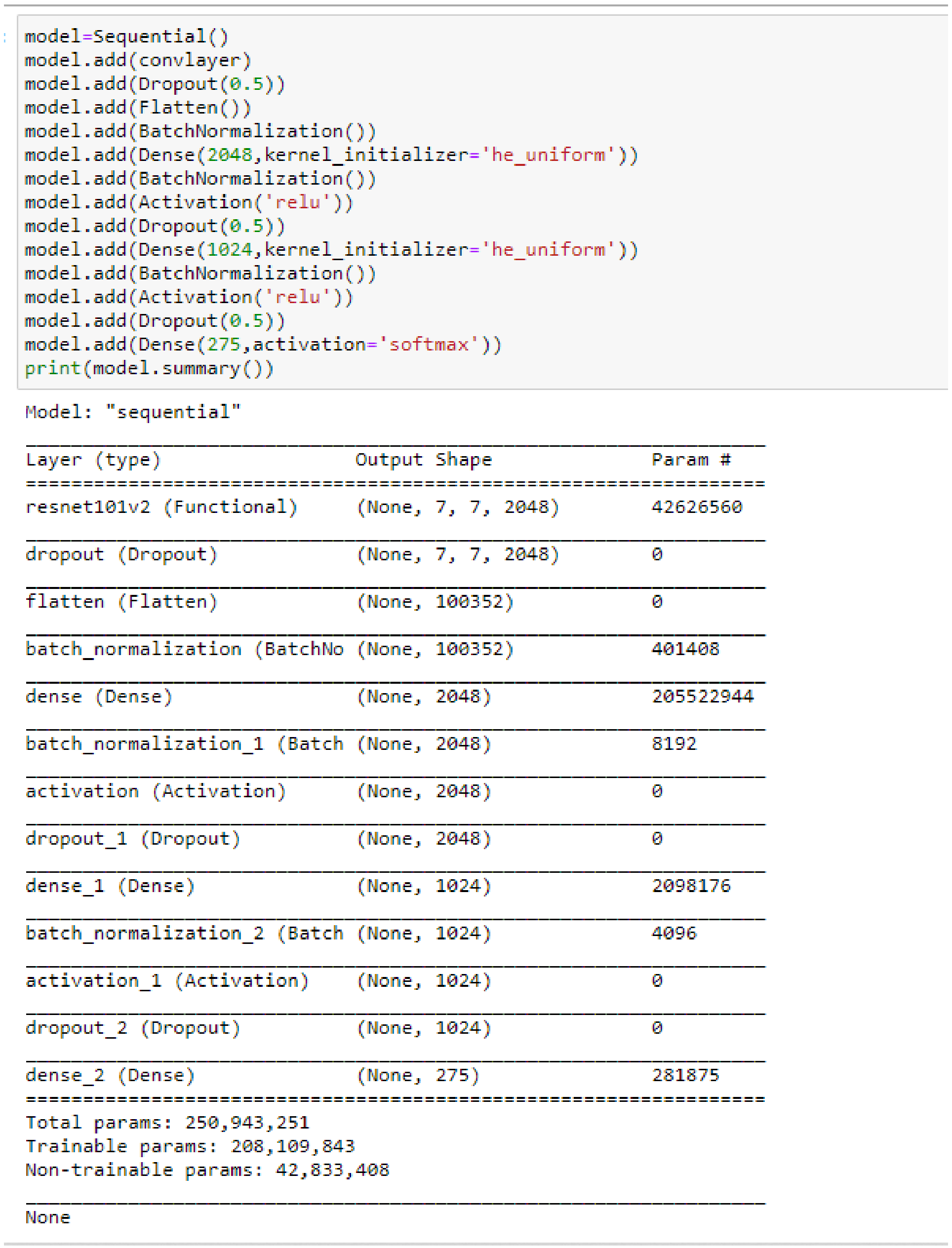

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
$\mathbf{x}$
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 7: Resnet Working

Above Diagram shows working of ResNet. Here input is x and weights are added in the layer sequentially. It uses activation function relu. Relu is Rectified linear Unit Activation function. It will take output the input directly if it is positive, otherwise, it will output zero (If the output is negative then it takes zero). ResNet consist convolution and pooling layers.



```
model=Sequential()
model.add(convlayer)
model.add(Dropout(0.5))
model.add(Flatten())
model.add(BatchNormalization())
model.add(Dense(2048,kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1024,kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(275,activation='softmax'))
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet101v2 (Functional)     (None, 7, 7, 2048)        42626560
_________________________________________________________________
dropout (Dropout)            (None, 7, 7, 2048)        0
_________________________________________________________________
flatten (Flatten)            (None, 100352)            0
_________________________________________________________________
batch_normalization (BatchNo (None, 100352)            401408
_________________________________________________________________
dense (Dense)                (None, 2048)              205522944
_________________________________________________________________
batch_normalization_1 (Batch (None, 2048)              8192
_________________________________________________________________
activation (Activation)      (None, 2048)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              2098176
_________________________________________________________________
batch_normalization_2 (Batch (None, 1024)              4096
_________________________________________________________________
activation_1 (Activation)    (None, 1024)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 275)               281875
=================================================================
Total params: 250,943,251
Trainable params: 208,109,843
Non-trainable params: 42,833,408
_________________________________________________________________
None
```

Figure 8: Model Build

Algorithm Description:

Bird Species Classifier consists of advanced convolution neural network. Algorithm design of convolution network uses an insight knowledge of machine learning as well as neural networks algorithms. Deep convolution neural networks use advanced filters to detect the patterns that are present within the image. We will make a filter, that will be randomized using various distributions like Gaussian, normal, uniform distribution. So, we place a randomized filter on an image and figure out the pattern. Multiple filters, max-pooling layers, fully connected pooling layers, and dense layers are used to detect the pattern in an image. Activation functions and dropout also played a role in the development of our deep convolution network system that is quite robust. We have constructed a model that contains a layer architecture and passed our dataset through the pre-trained model ResNet101V2 by using ImageNet weights. Basic elements for algorithm:

• Filter is basically used as an agent to detect the patterns within the images. Its main task is to do convolution by continuous movement on the image. There are several types of filters like horizontal filters, vertical filters, inclined filters. In our algorithm, we will use randomized filters that are picked from random probability distribution functions.

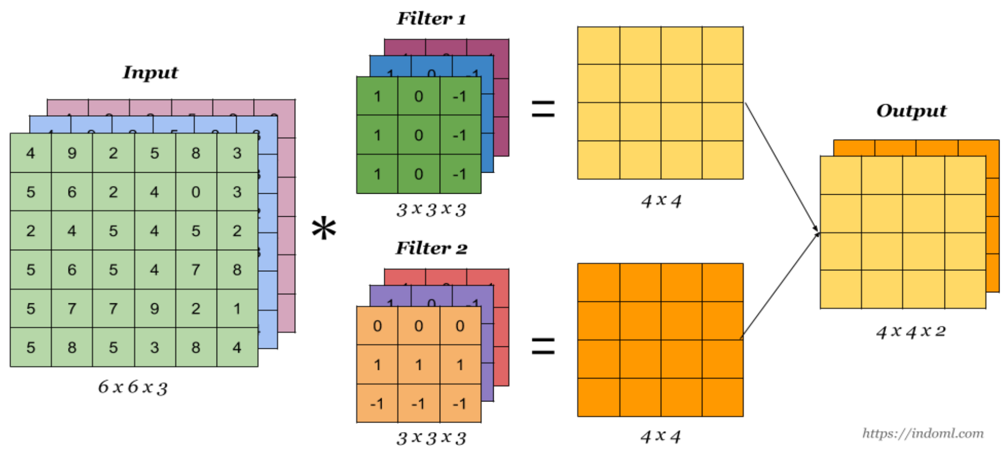


Figure 9: Convolution Neural Network

- **Convolution Layer:**

Convolution Layers plays a most important role in the classification of any type of images dataset. Convolution layers preserve the relationship between the neighboring pixels which is not possible in the case of dense layers neural networks that are mostly used in sentimental analysis and where there is no image data. Convolution layers are a major part for bird species classifier we have used many layers of convolution to detect the much more complex patterns that are not at all possible for humans to detect from their eyes. Convolution layer consists of many inputs that are necessary for the initialization of layer. Convolution layer consists of filter that is used to initialize number of filters, it contains strides meaning how much the filter has to move after it has completed the work with one set of pixels, it consists of activation function which in our case is mostly SoftMax activation function and padding is set to same so that it does not append any pixels if the convolution filter goes out of an image. The filter will be initialized from a random probability distribution to give out the best result as possible.

- **Activation Function:**

Activation functions are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network, and determines whether it should be activated or not, based on whether each neuron's input is relevant for the model prediction. In simple words, it is a transformation function that maps the input signals into output signals that are needed for the neural network to function. Activation functions help us to determine the output of the program by normalizing the output values in a range of 0 to 1.

In our case, we have mostly used a SoftMax activation function. The SoftMax regression is a form of logistic regression that normalizes an input value into a vector of values that follows a probability distribution whose total sums up to 1. The output values are between the range [0,1] which is nice because we are able to avoid binary classification and accommodate as many classes or dimensions in our neural network model.

- **Pooling Layer:**
  Pooling Layers are used as an intermediate layer between the convolution layer to decrease the height and width of an image that is passed through convolution layer. The main purpose is to make the image as deep as possible by using more and more filters on an image and using the pooling layers to reduce width and height of a passed image. We have several pooling layers but the major pooling layer which our used in our deep learning model are global max pooling layer and global average pooling layer

- **Stride and Padding:**
  Stride is the number of pixels shifts over the input matrix. When the stride is 1 then we move the filters to 1 pixel at a time. Stride and Padding are the two terms which play an important role in the convolution neural network. Stride is a parameter of the neural network's filter that modifies the amount of movement over the image. Padding is the space between an image or cell contents and its outside border. Padding specifies whether we have to add extra bits on the corner of an image or not.

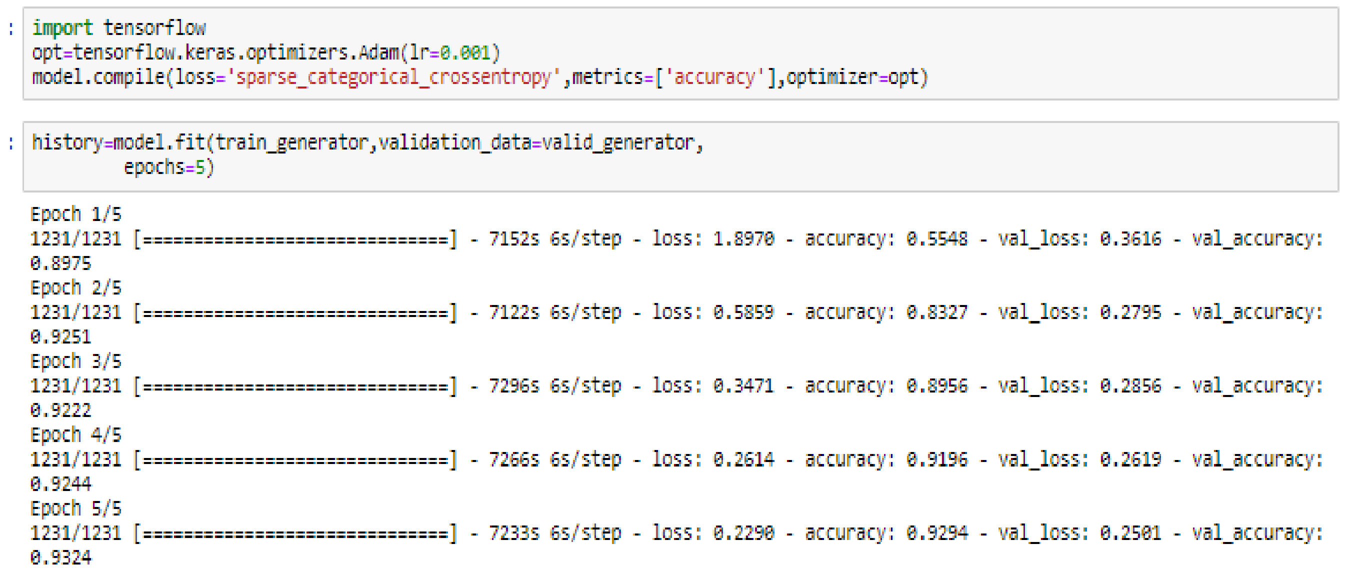
- **Dropout Layer:**
  Dropout refers to dropping or removing of neurons at random from a hidden layer in neural network to avoid overfitting. Dropout is an approach to regularization in neural network which helps in independent learning of the neurons. It is highly effective in avoid overfitting. Regularization reduces overfitting by adding penalty to all features, which are useless.
  Dropout Forces the Neural Network to learn more Robust Features that are Useful for Predictive analytics. Dropout Helps us to Reduce the Training Time required for the Neural Network. Choosing the Right value for Dropout is crucial to get Good Results.

- **BatchNormalization:**
  Batch normalization is a technique for training very deep neural network that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of epochs required to train deep networks. Batch normalization also adds two parameters to each layer, mean and standard deviation.

- **Dense Layer:**
  Dense Layer is a type of layer in which will all the input of current layer is connected to the next layer. In our model we've used dense layer at the end of model which will output 275 species of birds by calculating the probability of a certain species, the one with maximum probability is our bird corresponding to that image.

- **Feed Forward:**
  Feed Forward in a model means that we pass an input to the neural network in our case it is convolution neural network, the output is predicted based on the input and intermediate weights are initialized using random distributions. Once we predicted the output, we will calculate the error function by comparing the predicted output with the actual output. Based on this error we back propagate to reduce the error. This error can of several types like mean squared error, root mean squared error, categorical cross-entropy etc. There's a lot of calculation involved in this feed forward and back propagation.

- **Backpropagation:**
  Backpropagation is a technique of optimization of error function by using various optimization techniques like stochastic gradient descent, adagrad, rmsprop, Adam etc. The main purpose is to reduce by adjusting the weights of intermediate layers which requires a lot of calculations. In our case, we have used rmsprop optimizer to reduce the error function and increase the accuracy of our model.



```
import tensorflow
opt=tensorflow.keras.optimizers.Adam(lr=0.001)
model.compile(loss='sparse_categorical_crossentropy',metrics=['accuracy'],optimizer=opt)
```

```
history=model.fit(train_generator,validation_data=valid_generator,
        epochs=5)
```

```
Epoch 1/5
1231/1231 [==============================] - 7152s 6s/step - loss: 1.8970 - accuracy: 0.5548 - val_loss: 0.3616 - val_accuracy:
0.8975
Epoch 2/5
1231/1231 [==============================] - 7122s 6s/step - loss: 0.5859 - accuracy: 0.8327 - val_loss: 0.2795 - val_accuracy:
0.9251
Epoch 3/5
1231/1231 [==============================] - 7296s 6s/step - loss: 0.3471 - accuracy: 0.8956 - val_loss: 0.2856 - val_accuracy:
0.9222
Epoch 4/5
1231/1231 [==============================] - 7266s 6s/step - loss: 0.2614 - accuracy: 0.9196 - val_loss: 0.2619 - val_accuracy:
0.9244
Epoch 5/5
1231/1231 [==============================] - 7233s 6s/step - loss: 0.2290 - accuracy: 0.9294 - val_loss: 0.2501 - val_accuracy:
0.9324
```

Figure 10: Model fitting

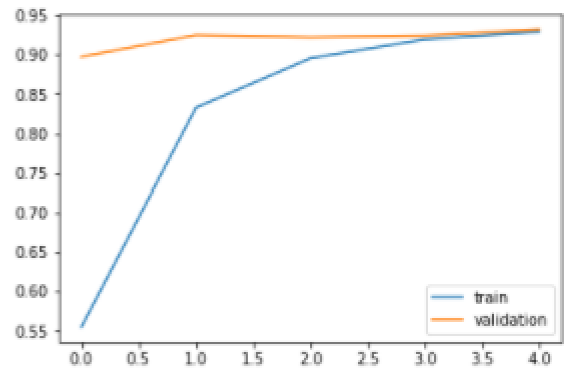
Description:

Here used Adam optimizer. Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm. It uses the squared gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using moving average of the gradient instead of gradient itself like SGD with momentum. we set a learning rate as 0.001 for better understanding of model. If learning rate is less then model will perform good and give better results.

Accuracy Plot

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['train', 'validation'], loc = 'lower right')
```

```
<matplotlib.legend.Legend at 0x2593b402a90>
```



```
model.save("BirdClassification.h5")
```

```
model.evaluate(test_generator)
```

```
43/43 [==============================] - 175s 4s/step - loss: 0.2593 - accuracy: 0.9389

[0.25929951667785645, 0.938909113407135]
```

Figure 11: Accuracy Plot

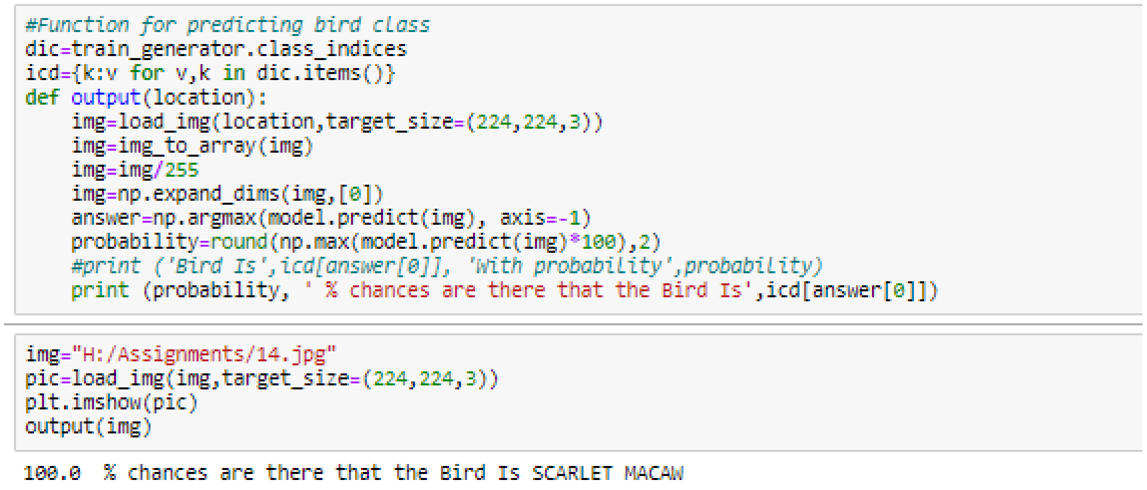
Description:

Here we plot graph of accuracy of train and validation data. By observing graph, we can say that accuracy of model is very good. Here we get 93.89% accuracy on the test data. so, we say that our model of CNN is good fit for the data.

TEST PLAN:

In this we are going to test our model so we are providing an image to predict its output.

1) Firstly, we are going give the following image of Scarlet Macaw to our model.



```
#Function for predicting bird class
dic=train_generator.class_indices
icd={k:v for v,k in dic.items()}
def output(location):
    img=load_img(location,target_size=(224,224,3))
    img=img_to_array(img)
    img=img/255
    img=np.expand_dims(img,[0])
    answer=np.argmax(model.predict(img), axis=-1)
    probability=round(np.max(model.predict(img)*100),2)
    #print ('Bird Is',icd[answer[0]], 'With probability',probability)
    print (probability, ' % chances are there that the Bird Is',icd[answer[0]])
```

```
img="H:/Assignments/14.jpg"
pic=load_img(img,target_size=(224,224,3))
plt.imshow(pic)
output(img)
```

```
100.0  % chances are there that the Bird Is SCARLET MACAW
```

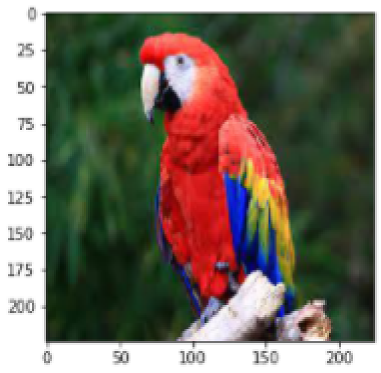
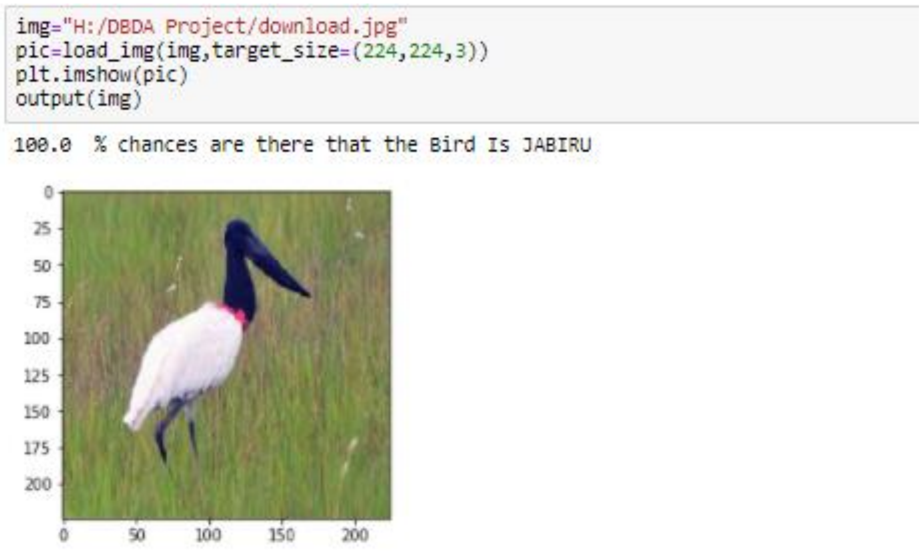


Figure 12: Model Prediction

Here we successfully predict the type of bird. Our model gives very accurate result of identifying bird’s species.

2) Now we are going to give image of 2nd bird.

This is image of JABIRU let's see how our model predict bird correctly or not



```
img="H:/DBDA Project/download.jpg"
pic=load_img(img,target_size=(224,224,3))
plt.imshow(pic)
output(img)
```

```
100.0  % chances are there that the Bird Is JABIRU
```

Here we successfully predict the type of bird. Our model gives very accurate result of identifying bird's species.

FUTURE SCOPE

In industry and even academia, developing a CNN is challenging because there is very little established methodology on determining the architecture, and conventional hyper-parameters but again not particularly well understood mechanisms for selecting them. System can be implemented using cloud which can store large amount of data for comparison and provide high computing power for processing (in case of Neural Networks).

In future we believe that with the help of object detection we count no of bird from each species are available in nature and according to that we spread awareness for saving different birds species. So, we can balance our environment.

Now a days we can see many birds but it’s difficult to identify their names, needs lots of information and knowledge about bird’s species but it’s necessary to identify their names so we could easily differentiate between them so with the help of our project people can easily identify birds.

CONCLUSION

In the end, we concluded that deep learning models have a very great capability to surpass the human potential if the data provided is sufficient. Engineers and scientists are still working on the deep learning field because till now the exploration of deep learning is limited. The proposed system works on the principle based on detection of a part and extracting CNN features from multiple convolutional layers. These features are aggregated and then given to the classifier for classification purpose. Deep learning has a lot of scope in medical sciences by analyzing the images by deep convolution neural network. Deep learning may be one of the possible reasons for the destruction of humankind. Transfer learning has a great scope in the future by combining a prebuilt model with the model we constructed. the system has provided the 93.89% accuracy in prediction of finding bird species.

REFERENCES

[1] Zagoruyko, S. and Komodakis, N., 2016. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. arXiv preprintarXiv:1612.03928.

[2] Fagerlund, S., 2007. Bird species recognition using support vector machines. EURASIP Journal on Applied Signal Processing, 2007(1),pp.64-64.

[3] Pradelle, B., Meister, B., Baskaran, M., Springer, J. and Lethin, R., 2017, November. Polyhedral Optimization of TensorFlow Computation Graphs. In 6th Workshop on Extreme-scale Programming Tools (ESPT-2017) at The International Conference for High Performance Computing, Networking,
Storage and Analysis (SC17).

[4]. Atanbori, J., Duan, W., Murray, J., Appiah, K., & Dickinson, P. (2016). Automatic classification of flying bird species using computer vision techniques. Pattern Recognition Letters, 81, 53–62." (2016)