## 

## 18CS3074P – PROJECT BASED REPORT

**ON**

BIRD SPECIES IDENTIFICATION FROM AN IMAGE

***Submitted in partial fulfillment of the requirement for the award of the degree of***

## BACHELOR OF TECHNOLOGY

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

**Id number:**

**180031315 PAVAN KUMAR BELLAMKONDA**

**180031318 ASHRITHA G**

**180031368 M ROHAN KANTH**

**180031385 BHAVYA LAHARI**

*Under the Esteemed Guidance of*

# Dr. S SAGAR IMAMBI

***Associate Professor***



## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**K L (Deemed to be) University**

Green Fields, Vaddeswaram, Guntur District – 522 502

## (2020-2021)

## 

## K L (Deemed to be) University

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**CERTIFICATE**

This is certify that the project based report entitled **“**BIRD SPECIES IDENTIFICATION FROM AN IMAGE**”** is a bonafide work done and submitted by 180031315, 180031318, 180031368, 180031385 in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in Department of Computer Science Engineering, K L (Deemed to be University), Guntur District during the academic year **2020-2021**.

**Dr. S SAGAR IMAMBI Mr. V. HARI KIRAN**

**FACULTYINCHARGE HEAD OF THE DEPARTMENT**

DEPARTMENT OF CSE DEPARTMENT OF CSE

K L (Deemed to be University) K L (Deemed to be University)

## K L (Deemed to be) University

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**DECLARATION**

This is certify that the project based report entitled **“**BIRD SPECIES INDENTIFICATION FROM AN IMAGE**”** is a bonafide work done and submitted by 180031315- Pavan Kumar Bellamkonda,180031318-Ashritha.G, 180031368- M.Rohan Kanth, 180031385- Bhavya Lahari in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in Department of Computer Science Engineering, K L (Deemed to be University), Guntur District during the academic year **2020-2021**.

**180031315-PAVAN KUMAR BELLAMKONDA**

**180031318-ASHRITHA G**

**180031368-M. ROHAN KANTH**

**180031385 -BHAVYA LAHARI**

## 

## ACKNOWLEDGEMENT

The success in this project would not have been possible but for the timely help and guidance rendered by many people. Our sincere thanks to all those who have assisted us in one way or the other for the completion of my project.

Our greatest appreciation to our guide **Dr. S SAGAR IMAMBI**, Associate Professor, Department of Computer Science and Engineering which cannot be expressed in words for his tremendous support, encouragement and guidance for this project.

We express our gratitude to **Mr. V. HariKiran,** Head of the Department for Computer Science and Engineering for providing us with adequate facilities, ways and means by which we are able to complete this project.

We thank all the members of teaching and non-teaching staff members, and also who have assisted me directly or indirectly for successful completion of this project.

Finally, we sincerely thank our friends and classmates for their kind help and co- operation during our work.

**180031315 PAVAN KUMAR BELLAMKONDA**

**180031318 ASHRITHA G**

**180031368 M. ROHAN KANTH**

**180031385 BHAVYA LAHARI**

## CONTENTS

|  |  |  |
| --- | --- | --- |
| **S No:** | **Tittle** | **Page No** |
| 1 | ABSTRACT | 6 |
| 2 | METHODOLOGY (Modules  Description) | 7 |
| 3 | SOFTWARE  REQUIREMENTS | 13 |
| 4 | DATASETS REQUIRED | 13 |
| 5 | IMPLEMENTATION | 15 |
| 6 | OUTPUT SCREENSHOTS | 17 |
| 7 | SCOPE OF PROJECT | 23 |
| 8 | CONCLUSION | 23 |

## ABSTRACT

Now a day some bird species are being found rarely, and if found, classification of bird species prediction is difficult. Naturally, birds present in various scenarios appear in different sizes, shapes, colors, and angles from the human perspective. Besides, the images present substantial variations to identify the bird species more than audio classification. Also, the human ability to recognize the birds through the images is more understandable.

Identification of bird species is a challenging task often resulting in ambiguous labels. Even professional bird watchers sometimes disagree on the species given an image of a bird. It is a difficult problem that pushes the limits of the visual abilities for both humans and computers.

Although different bird species share the same basic set of parts, different bird species can vary dramatically in shape and appearance. Intraclass variance is high due to variation in lighting and background and extreme variation in pose (e.g., flying birds,swimming birds, and perched birds that are partially occluded by branches). An automatic classification system for bird species are needed, which will be great convenience for many practical applications. Classifying bird species is an interesting problem for Fine-grained categorization, also known as subcategory recognition, which is a subfield in object recognition.

The identification can be done through image, audio or video. An audio processing technique makes it possible to identify by capturing the audio signal of birds. But, due to the mixed sounds in environment such as insects, objects from real world, etc. processing of such information becomes more complicated. Usually, human beings find images more effective than audios or videos. So, an approach to classify bird using an image over audio or video is preferred. Bird species identification is a challenging task to humans as well as to computational algorithms that carries out such a task in an automatic fashion.

.  **METHODOLOGY**

**CONVOLUTION NEURAL NETWORK**: CNN consists of four layers: convolutional layer, activation layer, pooling layer and fully connected. Convolutional layer allows extracting visual features from an image in small amounts. Pooling is used to reduce the number of neurons from previous convolutional layer but maintaining the important information. Activation layer passes a value through a function which compresses values into range. Fully connected layer connects a neuron from one layer to every neuron in another layer. As CNN classifies each neuron in depth, so it provides more accuracy.

The CNN have two components:

1) Feature extraction part: Features are detected when network performs a series of convolutional and pooling operation.

2) Classification part: Extracted features are given to fully connected layer which acts as classifier.

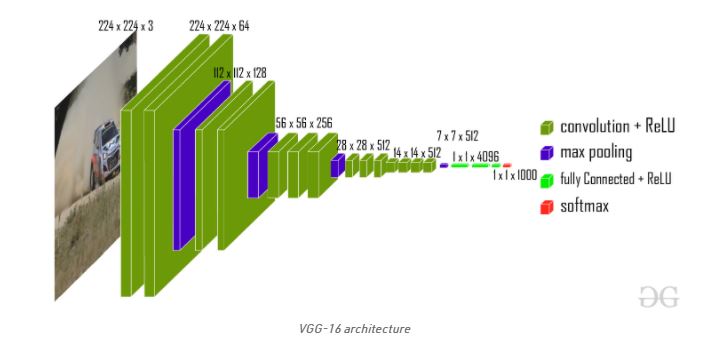
Feature Extraction Features are detected when network performs a series of convolutional and pooling operation.

Convolutional Layer: Convolutional neural network layer types mainly include three types, namely Convo- lutional layer, pooling layer and fully-connected layer Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli. Each convolutional neuron processes data only for its receptive field. Although fully connected feed forward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images.

It resolves the vanishing or exploding gradients problem in training traditional multi- layer neural networks with many layers by using back propagation. The aim of Convolutional layer is to learn feature representations of the inputs. Convolutional layer is consists of several feature maps. Each neuron of the same feature map is used to extract local characteristics of different positions in the former layer, but for single neurons, its extraction is local characteristics of same positions in former different feature map. In order to obtain a new feature, the input feature maps are first convolved with a learned kernel and then the results are passed into a nonlinear activation function. We have used pre-trained vgg16 model in our approach to solve the problem.

VGG-16 | CNN model

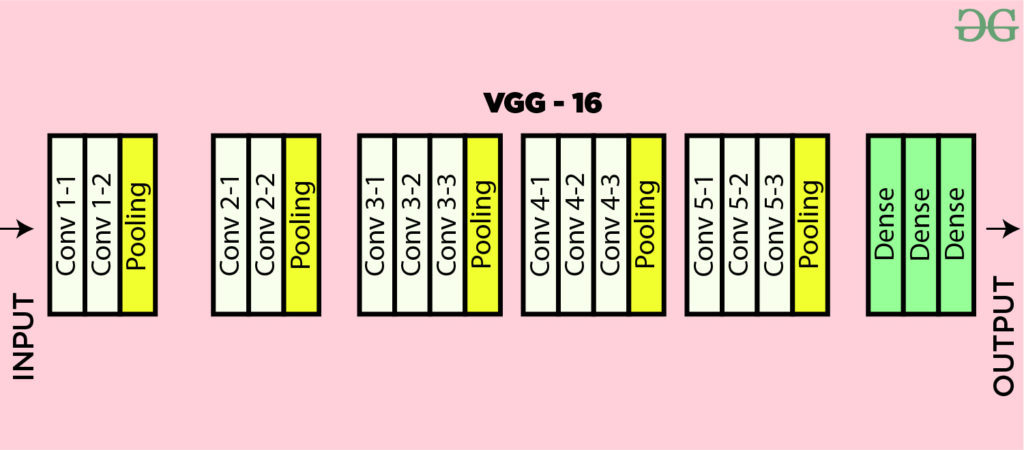
The ImageNet Large Scale Visual Recognition Challenge ([ILSVRC](http://www.image-net.org/challenges/LSVRC/)) is an annual computer vision competition. Each year, teams compete on two tasks. The first is to detect objects within an image coming from *200* classes, which is called object localization. The second is to classify images, each labelled with one of *1000* categories, which is called image classification. VGG 16 was proposed by Karen Simonyan and Andrew Zisserman of the Visual Geometry Group Lab of Oxford University in 2014 in the paper “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION”. This model won the 1st and 2nd place on the above categories in 2014 ILSVRC challenge.



This model achieves 92.7% top-5 test accuracy on ImageNet dataset which contains 14 million images belonging to 1000 classes.

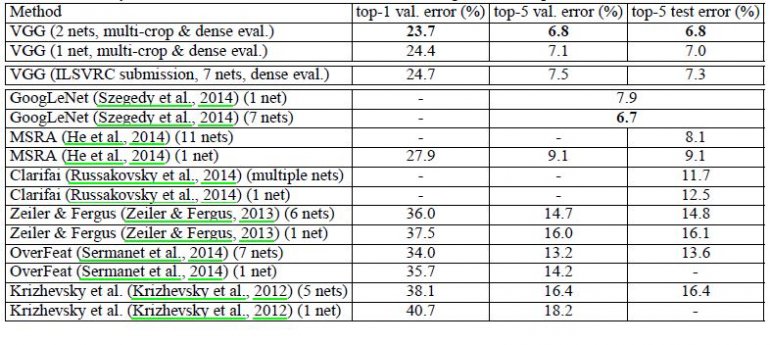
The ImageNet dataset contains images of fixed size of 224\*224 and have RGB channels. So, we have a tensor of (224, 224, 3) as our input. This model process the input image and outputs the a vector of 1000 values.

**Architecture:**  
The input to the network is image of dimensions (224, 224, 3). The first two layers have 64 channels of 3\*3 filter size and same padding. Then after a max pool layer of stride (2, 2), two layers which have convolution layers of 256 filter size and filter size (3, 3). This followed by a max pooling layer of stride (2, 2) which is same as previous layer. Then there are 2 convolution layers of filter size (3, 3) and 256 filter. After that there are 2 sets of 3 convolution layer and a max pool layer. Each have 512 filters of (3, 3) size with same padding. This image is then passed to the stack of two convolution layers. In these convolution and max pooling layers, the filters we use is of the size 3\*3 instead of 11\*11 in Alex Net and 7\*7 in ZF-Net. In some of the layers, it also uses 1\*1 pixel which is used to manipulate the number of input channels. There is a padding of 1-pixel (same padding) done after each convolution layer to prevent the spatial feature of the image.



After the stack of convolution and max-pooling layer, we got a (7, 7, 512) feature map. We flatten this output to make it a (1, 25088) feature vector.After this there are 3 fully connected layer, the first layer takes input from the last feature vector and outputs a (1, 4096) vector, second layer also outputs a vector of size (1, 4096) but the third layer output a 1000 channels for 1000 classes of ILSVRC challenge, then after the output of 3rd fully connected layer is passed to softmax layer in order to normalize the classification vector. After the output of classification vector top-5 categories for evaluation. All the hidden layers use ReLU as its activation function. ReLU is more computationally efficient because it results in faster learning and it also decreases the likelihood of vanishing gradient problem.

**Result**

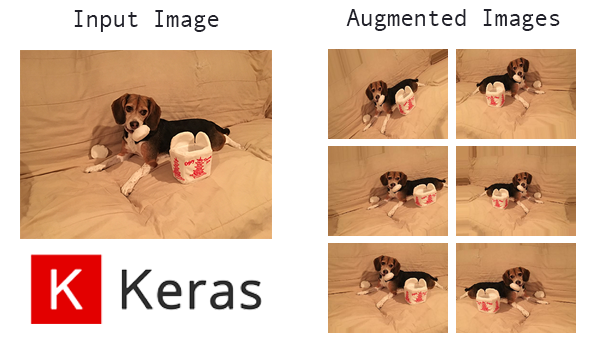
VGG16 significantly outperforms the previous generation of models in the ILSVRC-2012 and ILSVRC-2013 competitions. The VGG16 result is also competing for the classification task winner (GoogLeNet with 6.7% error) and substantially outperforms the ILSVRC-2013 winning submission Clarifai, which achieved 11.2% with external training data and 11.7% without it. Concerning the single-net performance, VGG16 architecture achieves the best result (7.0% test error), outperforming a single GoogLeNet by 0.9%.

It was demonstrated that the representation depth is beneficial for the classification accuracy, and that state-of-the-art performance on the ImageNet challenge dataset can be achieved using a conventional ConvNet architecture with substantially increased depth.

# Image data preprocessing

Pre-processing is a common name for operations with images at the lowest level of abstraction — both input and output are intensity images. These iconic images are of the same kind as the original data captured by the sensor, with an intensity image usually represented by a matrix of image function values (brightnesses). The aim of pre-processing is an improvement of the image data that suppresses unwilling distortions or enhances some image features important for further processing, although geometric transformations of images (e.g. rotation, scaling, translation) are classified among pre-processing methods here since similar techniques are used.

## Image augmentation

Image augmentation is a technique of applying different transformations to original images which results in multiple transformed copies of the same image. Each copy, however, is different from the other in certain aspects depending on the augmentation techniques you apply like shifting, rotating, flipping, etc.Applying these small amounts of variations on the original image does not change its target class but only provides a new perspective of capturing the object in real life. And so, we use it is quite often for building deep learning models. 

## Image augmentation in Keras

Keras **ImageDataGenerator**class provides a quick and easy way to augment your images. It provides a host of different augmentation techniques like standardization, rotation, shifts, flips, brightness change, and many more. However, the main benefit of using the Keras ImageDataGenerator class is that it is designed to provide real-time data augmentation. Meaning it is generating augmented images on the fly while your model is still in the training stage.

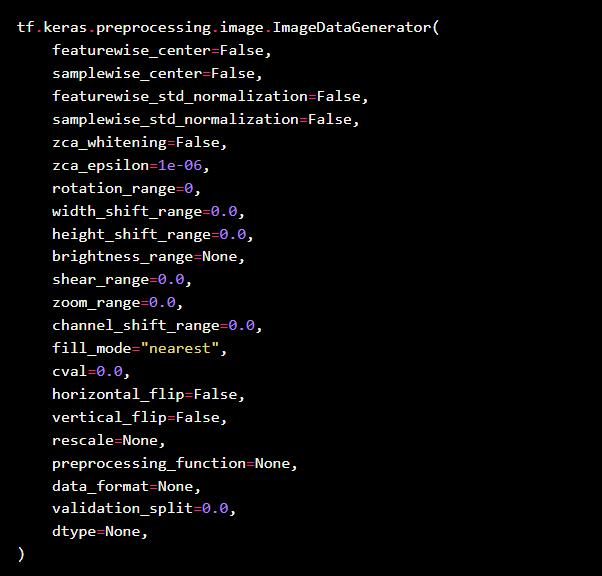
ImageDataGenerator class ensures that the model receives new variations of the images at each epoch. But it only returns the transformed images and does not add it to the original corpus of images. If it was, in fact, the case, then the model would be seeing the original images multiple times which would definitely overfit our model.Another advantage of ImageDataGenerator is that it requires lower memory usage. This is so because without using this class, we load all the images at once. But on using it, we are loading the images in batches which saves a lot of memory.

**How Keras ImageDataGenerator**

\* Take a batch of images used for training.

\* Apply random transformations to each image in the batch.

\* Replacing the original batch of images with a new randomly transformed batch.

\*Train a Deep Learning model on this transformed batch. 

**SOFTWARE AND HARDWARE REQUIREMENTS**

* Preferred OS (it can be a mac, Linux or Windows)
* Enough RAM should be present in your laptop or PC to run the computations without any interruptions.
* Jupyter Notebook (anaconda 3) must be installed.
* Google Colab. (Another environment like Jupyter Notebook)

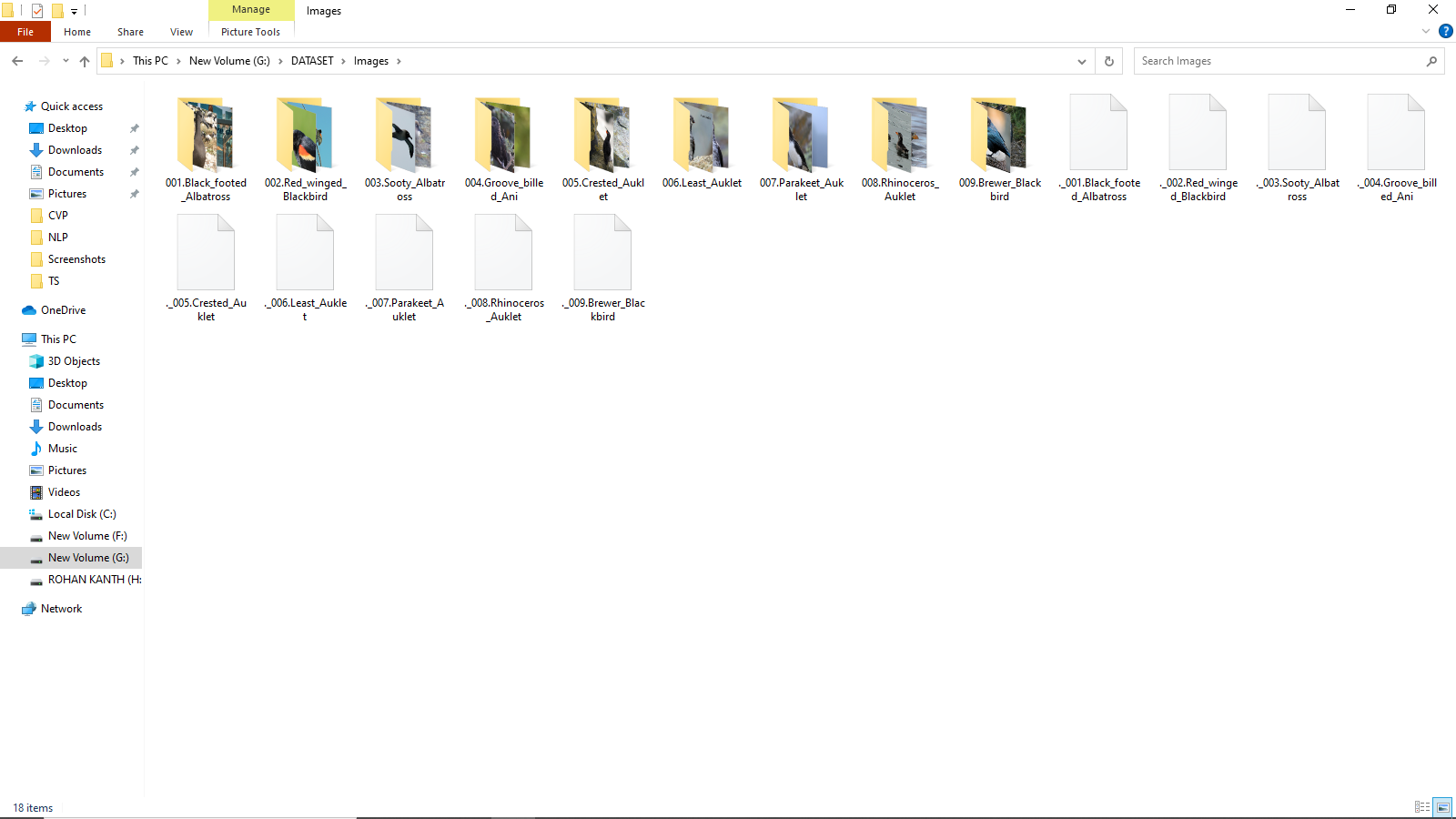
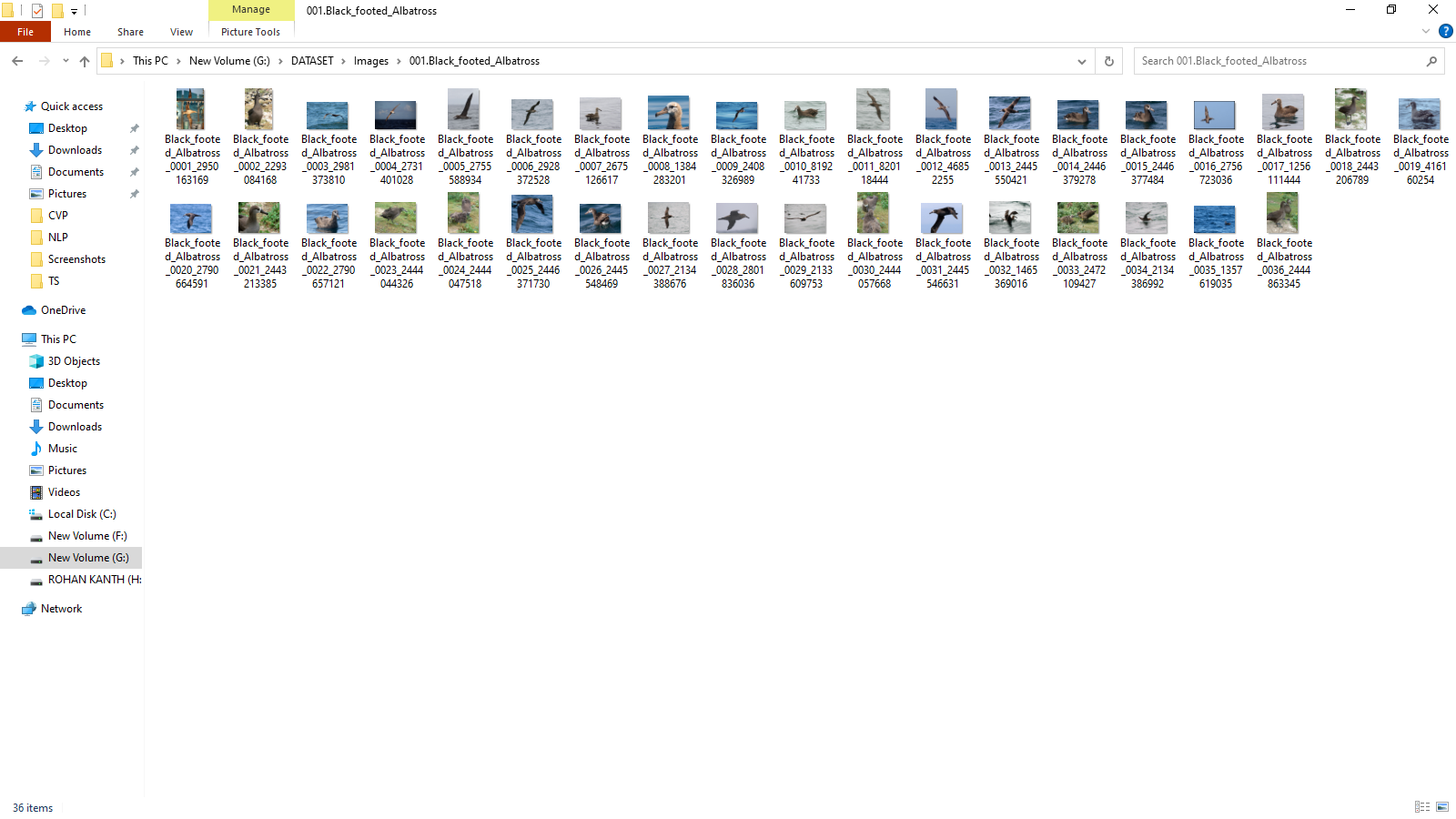
**DATASET(S) REQUIRED**

Caltech-UCSD Birds 200 [CUB-200-2011] dataset for training as well as testing purpose. This data set is widely used for projects that are related to birds and have a lot of classes and images related to a lot of species and captured those images in different angles and variations.



### CONTENT IN THE DATA SET:

A dataset is a collection of data. For performing action related to birds a dataset named CaltechUCSD Birds 200 (CUB-200-2011) is used. It is an extended version of the CUB-200 dataset, with roughly double the number of images per class and also has new part location annotations for higher accuracy [8]. The detailed information about the dataset is as follows: Number of categories: 200, Number dataset is validated with an accuracy of 75% to increase the performance of system.

## IMPLEMENTATION

## 

import cv2

from keras.layers import Input, Lambda, Dense, Flatten

from keras.models import Model

from keras.applications.vgg16 import VGG16

from keras.applications.vgg16 import preprocess\_input

from keras.preprocessing import image

from keras.preprocessing.image import ImageDataGenerator

from keras.models import Sequential

import numpy as np

from glob import glob

import matplotlib.pyplot as plt

# re-size all the images to this

IMAGE\_SIZE = [224, 224]

train\_path = 'G:\DATASET\Images'

valid\_path = 'G:\DATASET\Images'

# add preprocessing layer to the front of VGG

vgg = VGG16(input\_shape=IMAGE\_SIZE + [3], weights='imagenet', include\_top=False)

# don't train existing weights

for layer in vgg.layers:

layer.trainable = False

# useful for getting number of classes

folders = glob('G:\DATASET\Images\\*')

# our layers - you can add more if you want

x = Flatten()(vgg.output)

x = Dense(1000, activation='relu')(x)

prediction = Dense(len(folders), activation='softmax')(x)

# create a model object

model = Model(inputs=vgg.input, outputs=prediction)

# view the structure of the model

model.summary()

# tell the model what cost and optimization method to use

model.compile(

loss='categorical\_crossentropy',

optimizer='adam',

metrics=['acc']

)

from keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(rescale = 1./255,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True)

test\_datagen = ImageDataGenerator(rescale = 1./255)

training\_set = train\_datagen.flow\_from\_directory('G:\DATASET\Images',

target\_size = (224, 224),

batch\_size = 32,

class\_mode='categorical')

test\_set = test\_datagen.flow\_from\_directory('G:\DATASET\Images',

target\_size = (224, 224),

batch\_size = 32,

class\_mode = 'categorical')

'''r=model.fit\_generator(training\_set,

samples\_per\_epoch = 8000,

nb\_epoch = 5,

validation\_data = test\_set,

nb\_val\_samples = 2000)'''

print(len(training\_set))

print(len(test\_set))

# fit the model

r = model.fit\_generator(

training\_set,

validation\_data=test\_set,

epochs=5,

steps\_per\_epoch=len(training\_set),

validation\_steps=len(test\_set),

)

# loss

plt.plot(r.history['loss'], label='train loss')

plt.plot(r.history['val\_loss'], label='val loss')

plt.legend()

plt.show()

plt.savefig('LossVal\_loss')

# accuracies

plt.plot(r.history['acc'], label='train acc')

plt.plot(r.history['val\_acc'], label='val acc')

plt.legend()

plt.show()

plt.savefig('AccVal\_acc')

import tensorflow as tf

from keras.models import load\_model

model.save('facefeatures\_new\_model.h5')

test\_generator = train\_datagen.flow\_from\_dataframe(test,directory='G:\DATASET\Test',x\_col = 'Image',

y\_col = None,target\_size=(224,224),class\_mode = None,batch\_size=32)

test\_x=model.predict(test\_generator)

test\_x=np.array(test\_x)

test\_x

out=np.argmax(test\_x,axis=1)

out

import pandas as pd

import os

test=pd.DataFrame()

test\_image=os.listdir('G:\DATASET\Test')

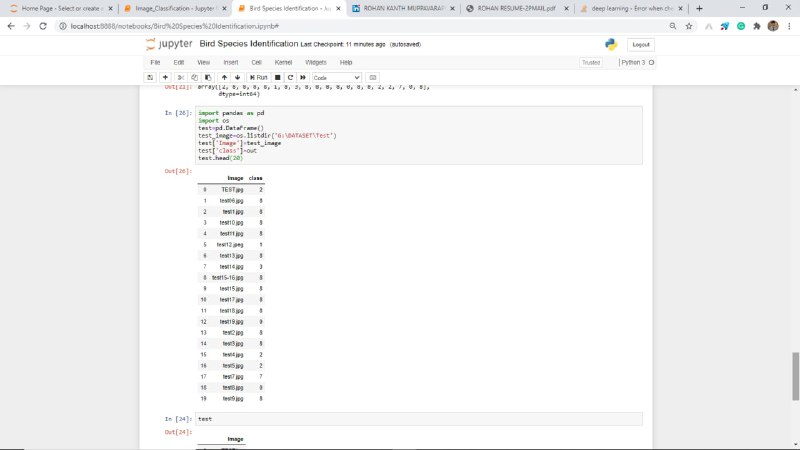
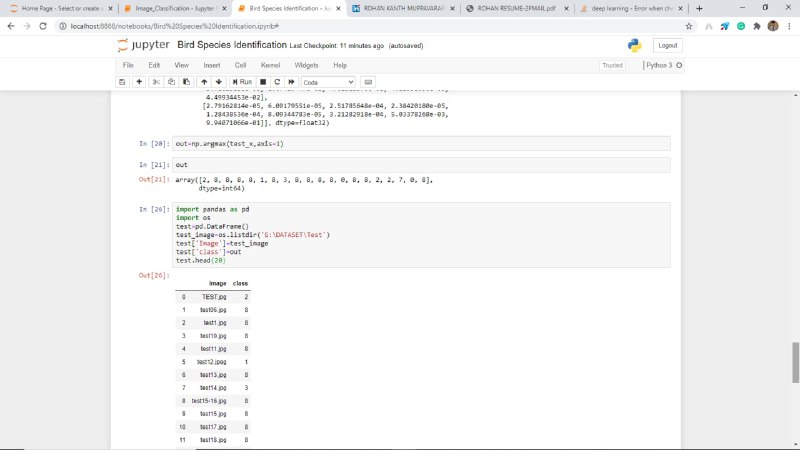
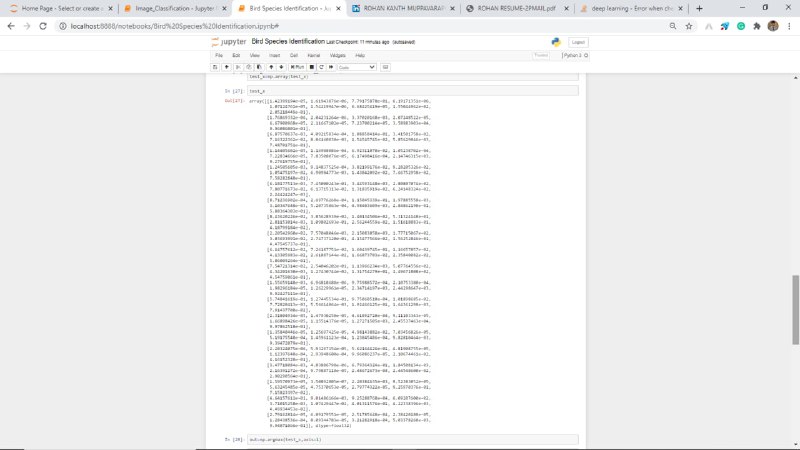
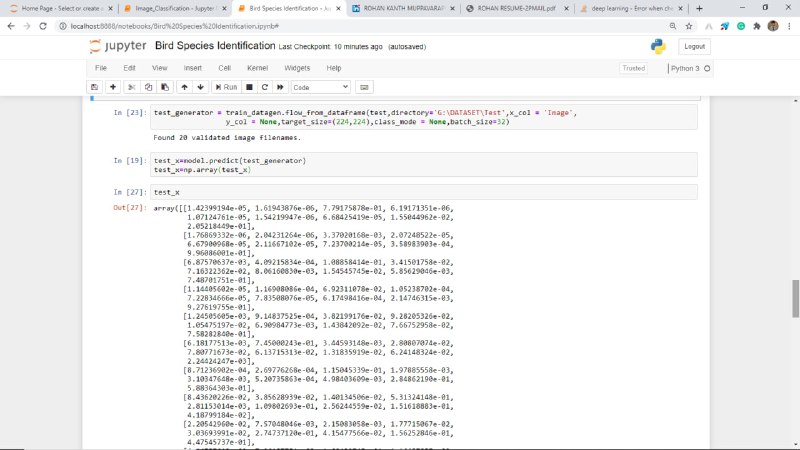
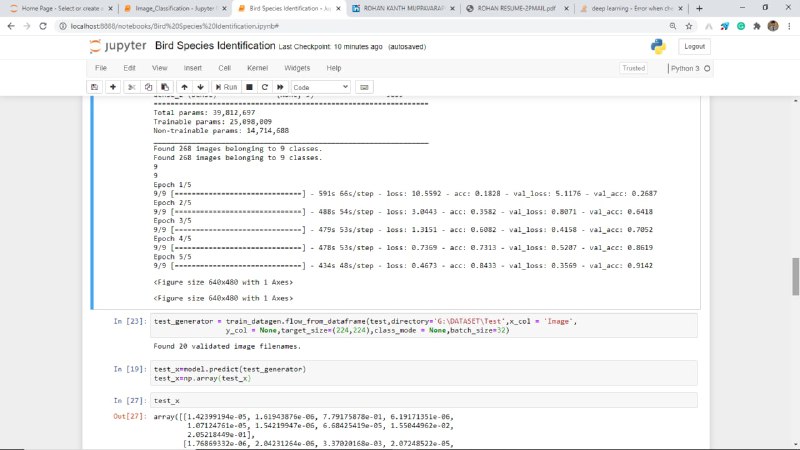
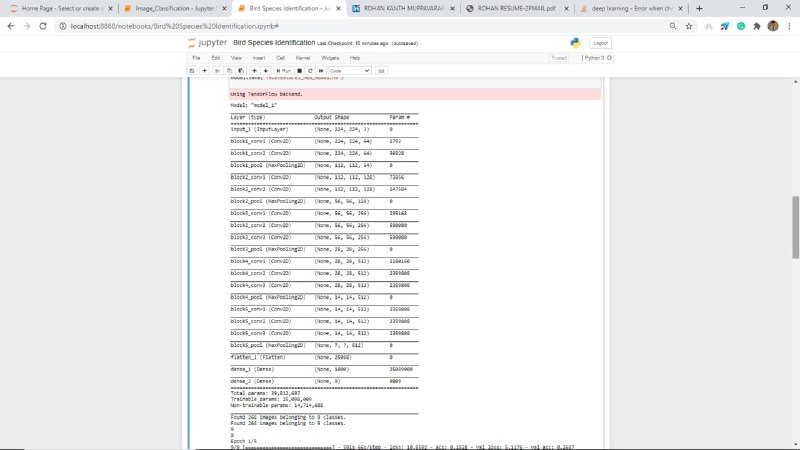
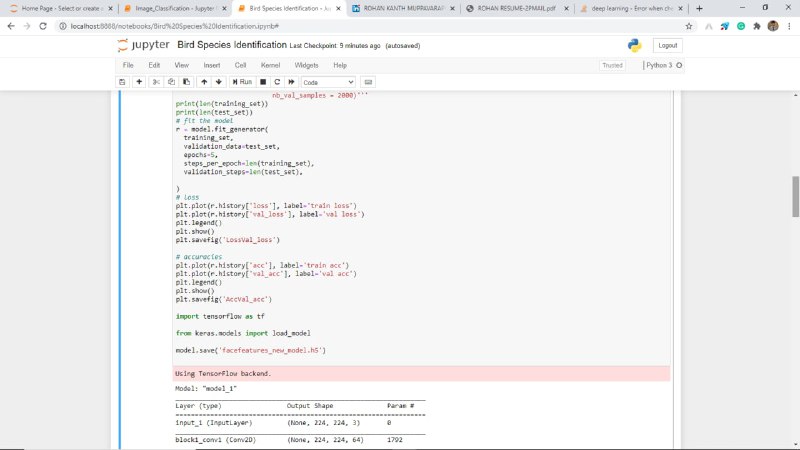
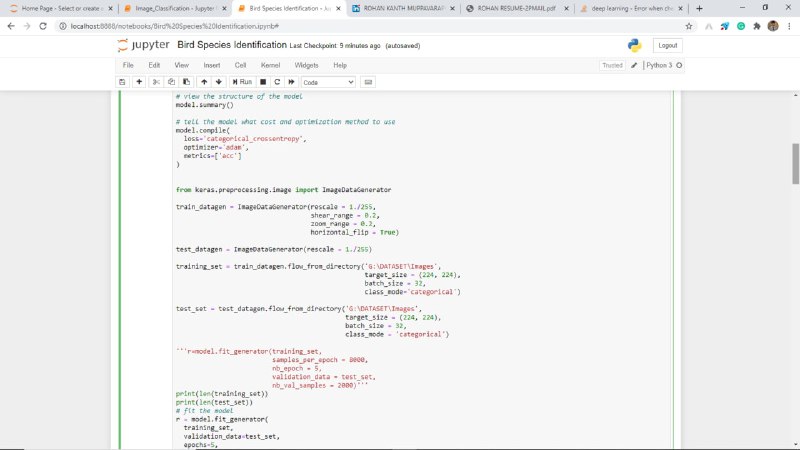
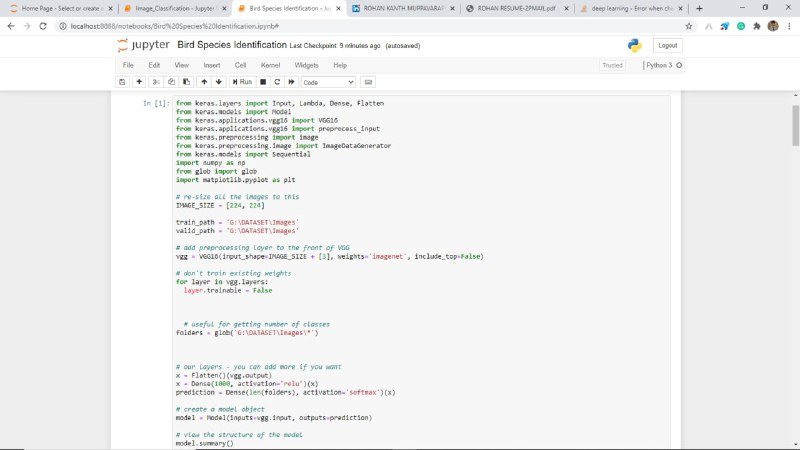
test['Image']=test\_image

test['class']=out

test.head(20)

test

**OUTPUT**



**FUTURE SCOPE**

It should be Implemented in cloud for fast computation and better results. We can also use web technologies like Flask for web deployment of the model in order to make it more user accessible and readily used.

**CONCLUSION**

In this project we fed the image to system and given to VGG16 model which is a part of CNN, is coupled with trained dataset. A CNN consists of various convolutional layers. Various alignments/features such as head, body, color, beak, shape, entire image of bird are considered for classification to yield maximum accuracy. Each alignment is given through deep convocational network to extract features out from multiple layers of network.