BIRD CLASSIFICATION USING DEEP LEARNING TECHNIQUES

CONTENTS

INTRODUCTION

Birds are the warm-blooded vertebrates constituting class Aves, there are nearly 10 thousand living species of birds in the world with multifarious characteristics and appearances. Many people visit bird sanctuaries and look around their surrounding nature to enjoy the beautiful variants of colours and characteristics of the birds. But People barely have knowledge about the various species and hence cannot easily distinguish the characteristics and the species name without expertise in the field of ornithology. So in this project, I have tried to use modern artificial intelligence techniques to make automatic identification and classification of birds. Deep Learning is a subset of machine learning which gives us various algorithms inspired by human neural networks. The algorithm imitates the working of human brains in the processing the data and produces a pattern of data for decision making. This work presents an approach of convolutional neural network models for identifying various species of birds.

AIM OF THIS PROJECT

The main objective of this project is to use deep learning techniques to classify 10 different bird species. Along with training the convolution neural networks for image classification in the transfer learning method, the aim of this project was to create a web app using ‘streamlit’ open source python library so that anyone without coding knowledge can use this model for their purpose.

THEORY

In this section we will discuss all the concepts important for continuing this work.

DATA ANALYSIS: This work has been done with images of 10 different species of birds, the bird species are Black footed Albatross, Laysan Albatross, Sooty Albatross, Groove billed Ani, Least Auklet, Parakeet Auklet, Rhinoceros Auklet, Brewer Black bird, Red winged Black bird and Crested Auklet. So this is an image classification problem and convolutional neural networks can be used to solve this problem. As we are using deep learning algorithms, so no need to do feature extraction manually.

DEEP LEARNING TECHNIQUES: All the deep learning techniques used in this project are discussed below.

CONVOLUTIONAL NEURAL NETWORK: A Convolutional Neural Network, also known as CNN or ConvNet, is a class of neural networks that specializes in processing data that has a grid-like topology, such as an image. A digital image is a binary representation of visual data. It contains a series of pixels arranged in a grid-like fashion that contains pixel values to denote how bright and what color each pixel should be. Just as each neuron responds to stimuli only in the restricted region of the visual field called the receptive field in the biological vision system, each neuron in a CNN processes data only in its receptive field as well. The layers are arranged in such a way so that they detect simpler patterns first (lines, curves,

etc.) and more complex patterns (faces, objects, etc.) further along. By using a CNN, one can enable sight to computers.

Architecture of CNN:

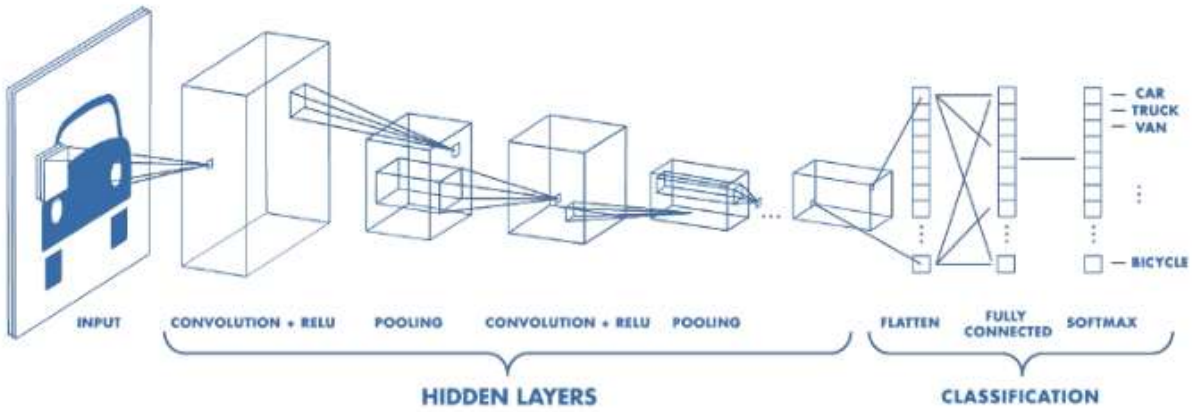


FIG: Architecture of a CNN

i) CONVOLUTION LAYER: The convolution layer is the core building block of the CNN. This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field. The kernel is spatially smaller than an image but is more in-depth.

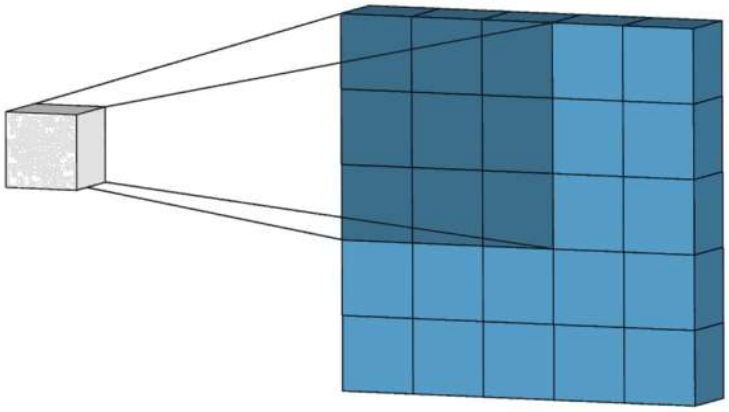


FIG: Illustration of Convolution Operation

During the forward pass, the kernel slides across the height and width of the image-producing image representation of that receptive region. This produces a two-dimensional representation of the image known as an activation map that gives the response of the kernel at each spatial position of the image. The sliding size of the kernel is called a stride. If we have an input of size W x W x D and Dout number of kernels with a spatial size of F with stride S and amount of padding P, then the size of output volume can be determined by the following formula:

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

.

ii) POOLING LAYERS: The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights. There are several pooling functions such as the average of the rectangular neighborhood, L2 norm of the rectangular neighborhood, and a weighted average based on the distance from the central pixel but the most popular process is max pooling, which reports the maximum output from the neighborhood.

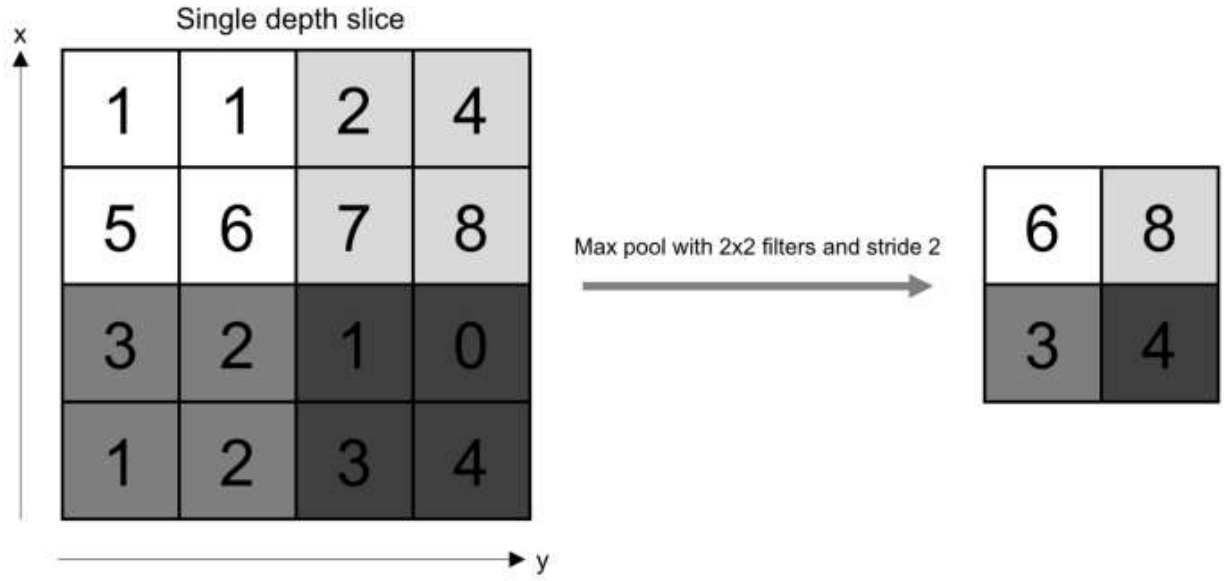


FIG: Pooling Operations

If we have an activation map of size W x W x D, a pooling kernel of spatial size F, and stride S, then the size of output volume can be determined by the following formula:

$$W_{out} = \frac{W - F}{S} + 1$$

iii) FULLY CONNECTED LAYER: Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer as seen in regular FCNN. This is why it can be computed as usual by a matrix multiplication followed by a bias effect. The FC layer helps to map the representation between the input and the output.

iv) NON-LINEARITY LAYERS: Since convolution is a linear operation and images are far from linear, non-linearity layers are often placed directly after the convolutional layer to introduce non-linearity to the activation map. There are several types of non-linear operations, the popular ones are sigmoid, RELU and tanh.

TRANSFER LEARNING METHOD: 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.

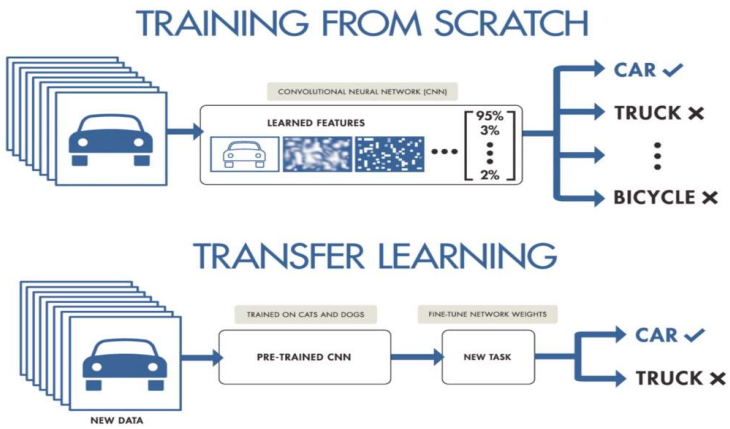


FIG: Illustration for demonstrating difference between Traditional Training and Transfer learning.

For image classifications there are many top-performing models; some famous models are VGG (e.g. VGG16 or VGG19), GoogLeNet (e.g. InceptionV3) and Residual Network (e.g. ResNet50). We can use all these pre-trained models in many different ways; some of these usage patterns as follows:

i) CLASSIFIER: Those pre-trained models are used directly to classify new images.

ii) STANDALONE FEATURE EXTRACTOR: The pre-trained model, or some portion of the model, is used to pre-process images and extract relevant features.

iii) INTEGRATED FEATURE EXTRACTOR: The pre-trained model, or some portion of the model, is integrated into a new model, but layers of the pre-trained model are frozen during training.

iv) WEIGHT INITIALIZATION: The pre-trained model, or some portion of the model, is integrated into a new model, and the layers

of the pre-trained model are trained in concert with the new model.

Each approach can be effective and save significant time in developing and training a deep convolutional neural network model.

EVOLUTION OF MODEL : The last step of this work is evaluating all the applied models to find out the best model for our given case. The evaluation metrics we are going to use are the accuracy score metric, f1 score metric, and finally the confusion matrix.

i) ACCURACY SCORE MATRIC: Accuracy score is one of the most basic evaluation metrics which is widely used to evaluate classification models. It can generally be expressed as:

Accuracy score = No.of correct predictions / Total no.of predictions.

ii) CONFUSION MATRIX: Typically, a confusion matrix is a visualization of a classification model that shows how well the model has predicted the outcomes when compared to the original ones. Usually, the predicted outcomes are stored in a variable that is then converted into a correlation table.

WORK-FLOWS

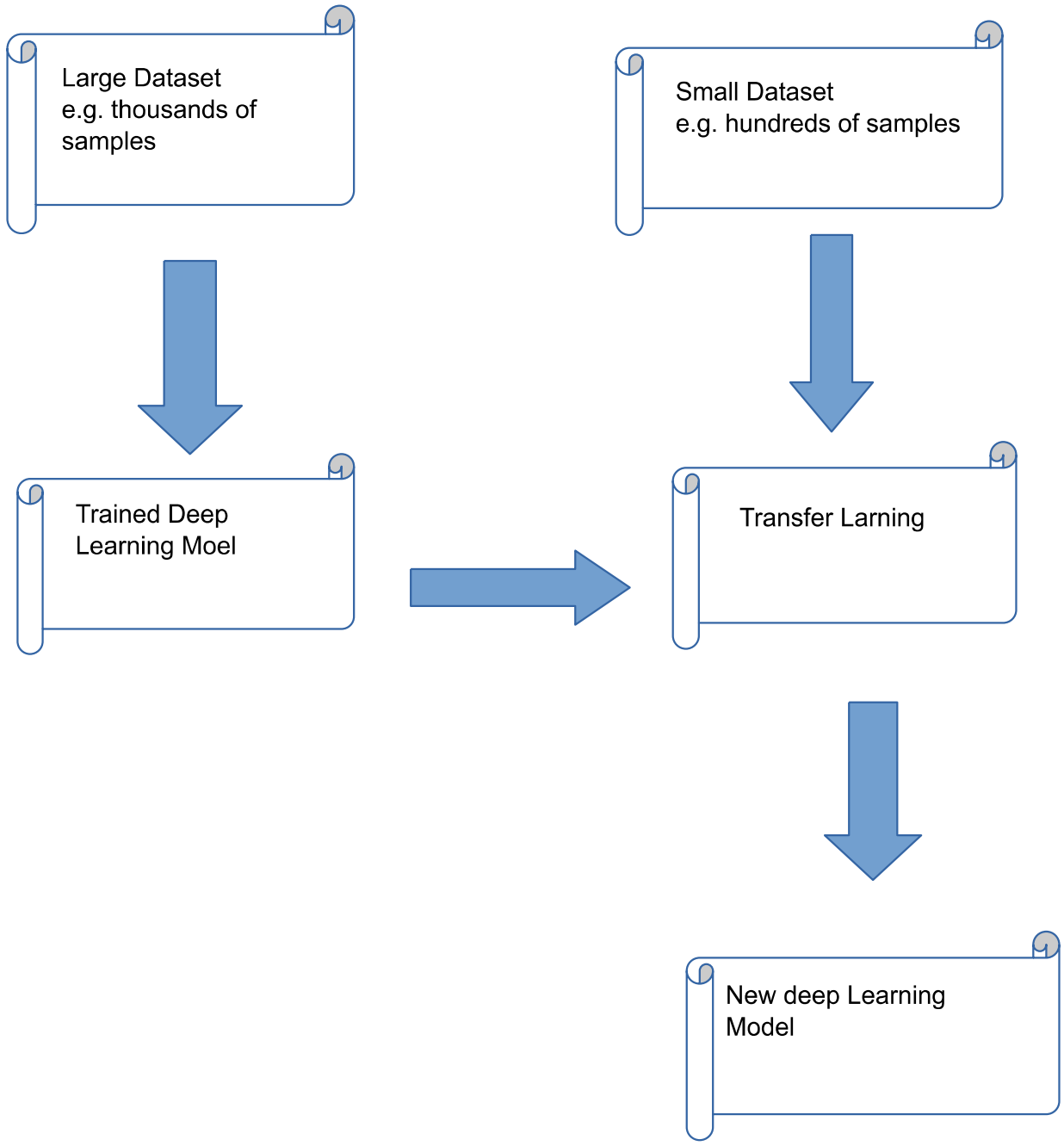


FIG: work-flow for transfer learning

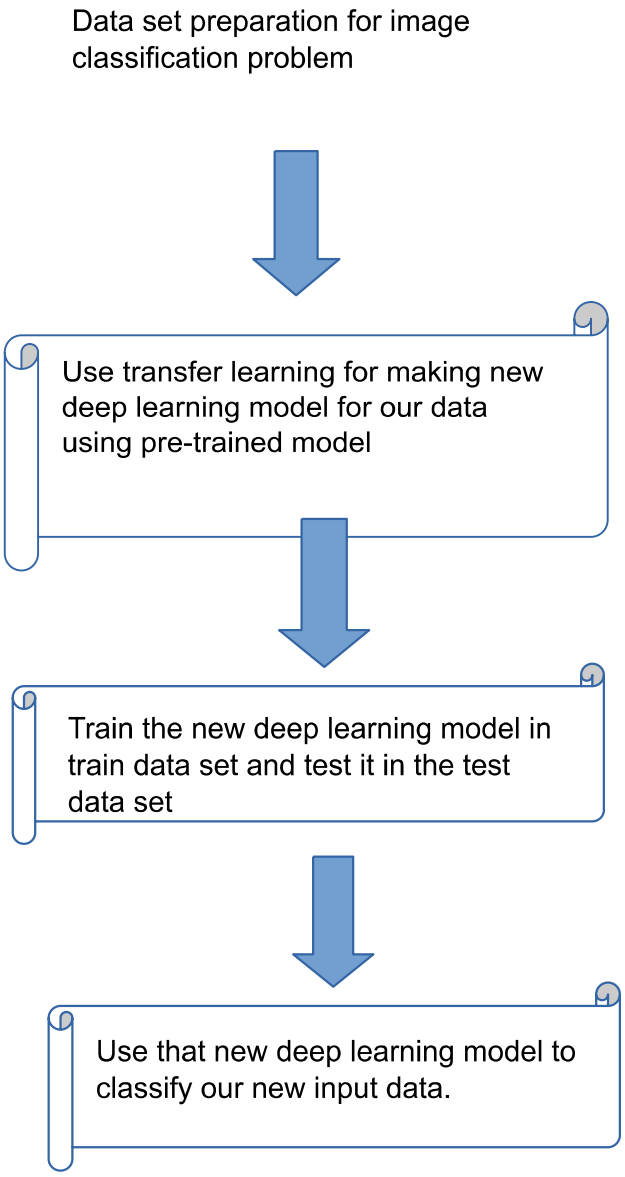


FIG: work-flow for the whole work.

METHODOLOGY

DATA SET PREPARATION:

Numerous images from the internet for each bird species are collected to prepare the data set. Those images of different bird species are labelled by integers from 0 to 9 for classification purposes. 85% of those collected images were used to train the model and 15% of those photos were used to test the model.

FIG: Labeled bird species

USING PRE-TRAINED MODEL TO CLASSIFY NEW DATA:

For this project the coding work has been done on python, so first it is necessary to import all the libraries needed for the work. Here I have used tensorflow.keras to work with deep learning models, PIL to handle image files and numpy for working with arrays.

```
import tensorflow.keras
from PIL import Image, ImageOps
import numpy as np
import tensorflow as tf

# Disable scientific notation for clarity
np.set_printoptions(suppress=True)

# Load the model
model = tensorflow.keras.models.load_model('/content/drive/MyDrive/my_keras_model.h5')
model.summary()
model.input_shape
```

By using the above code, we can also load the deep learning model which is saved in .h5 format and get useful information about the model.

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential_1 (Sequential)    (None, 1280)              410208
_________________________________________________________________
sequential_3 (Sequential)    (None, 10)                129100
=================================================================
Total params: 539,308
Trainable params: 525,228
Non-trainable params: 14,080
```

So from the above result we got that the loaded model (my_keras_model.h5) is ‘sequential_4’ model and built by transfer learning methods. It contains two layers; the first layer ‘sequential_1’ is from a standard pre-trained model containing 1280 nodes and on top that the first layer one new layer ‘sequential_2’ is added which contains 10 nodes for our new data of classification for 10 different bird species. This ‘sequential_4’ model is trained and tested on the prepared data-set.

Now, the ‘sequential_4’ model can be used directly to classify new

images.

```
# Create the array of the right shape to feed into the keras model
# The 'length' or number of images you can put into the array is
# determined by the first position in the shape tuple, in this case 1.
# Create the array of the right shape to feed into the keras model
data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

# Replace this with the path to your image
image = Image.open('/content/drive/MyDrive/test1.jpeg')
```

```
#display the image
image
```

By using above codes a new input image can be taken and we are going to use our model 'sequential_4' to classify this image.
The input image is from the bird species 'Red winged Black bird'.

```
image.split()
```

```
(<PIL.Image.Image image mode=L size=259x194 at 0x7FD4C405EDD0>,
 <PIL.Image.Image image mode=L size=259x194 at 0x7FD4C405EE90>,
 <PIL.Image.Image image mode=L size=259x194 at 0x7FD4C405EED0>)
```

```
#resize the image to a 224x224 with the same strategy as in TM2:
#resizing the image to be at least 224x224 and then cropping from the center
size = (224, 224)
image = ImageOps.fit(image, size, Image.ANTIALIAS)

#turn the image into a numpy array
image_array = np.asarray(image)
```

```
# Normalize the image
normalized_image_array = (image_array.astype(np.float32) / 127.0) - 1

# Load the image into the array
data[0] = normalized_image_array
```

```
# run the inference
prediction = model.predict(data)
print(prediction)
```

By using the above lines of code we can get more information about our new input image and resize it accordingly so that the previously built model can be used on it to predict its class.

PREPARATION OF WEB APP INTERFACE: If people without coding knowledge can use our model then it can serve a great purpose. That's why one simple web app is prepared using an open source python library called 'streamlit'. Below here its implementations are discussed.

```
!pip install streamlit
```

```
%%writefile app1.py
import tensorflow.keras
from PIL import Image, ImageOps
import numpy as np
import tensorflow as tf
import streamlit as st

model = tensorflow.keras.models.load_model('/content/drive/MyDrive/my_keras_model.h5')


st.write("# Bird species classification")
file= st.file_uploader('please upload an image file' ,type=['png','jpeg','jpg'])


def import_and_predict(image_data, model):
    data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)
    size = (224,224)
    image = ImageOps.fit(image_data, size, Image.ANTIALIAS)
    image = np.asarray(image)
    # Normalize the image
    normalized_image_array = (image.astype(np.float32) / 127.0) - 1
    # Load the image into the array
    data[0] = normalized_image_array
    prediction = model.predict(data)
```

Using the above lines of codes, necessary packages are downloaded and a pre-trained model is uploaded in .py file and also the image taken from the web server is resized.

```
if file is None:
  st.text("Please upload an image file")
else:
  image = Image.open(file)
  st.image(image, use_column_width=True)
  prediction = import_and_predict(image, model)

if np.argmax(prediction) == 0:
  st.write("Image is from Black footed Albatross species")
elif np.argmax(prediction) == 1:
  st.write("Image is from Laysan Albatross species")
elif np.argmax(prediction) == 2:
  st.write("Image is from Sooty Albatross species")
elif np.argmax(prediction) == 3:
  st.write(" Image is from Groove billed Ani species")
elif np.argmax(prediction) == 4:
  st.write("Image is from Least Auklet species")
elif np.argmax(prediction) == 5:
  st.write("Image is from Parakeet Auklet species ")
elif np.argmax(prediction) == 6:
  st.write("Image is from Rhinoceros Auklet species")
elif np.argmax(prediction) == 7:
  st.write("Image is from Brewer Blackbird   species")
elif np.argmax(prediction) == 8:
```

```
  st.write("Image is from Red winged Blackbird species")
else:
  st.write("Image is from Crested Auklet species ")

st.write(prediction)
```

```
Overwriting app1.py
```

```
!streamlit run app1.py &>/dev/null&
```

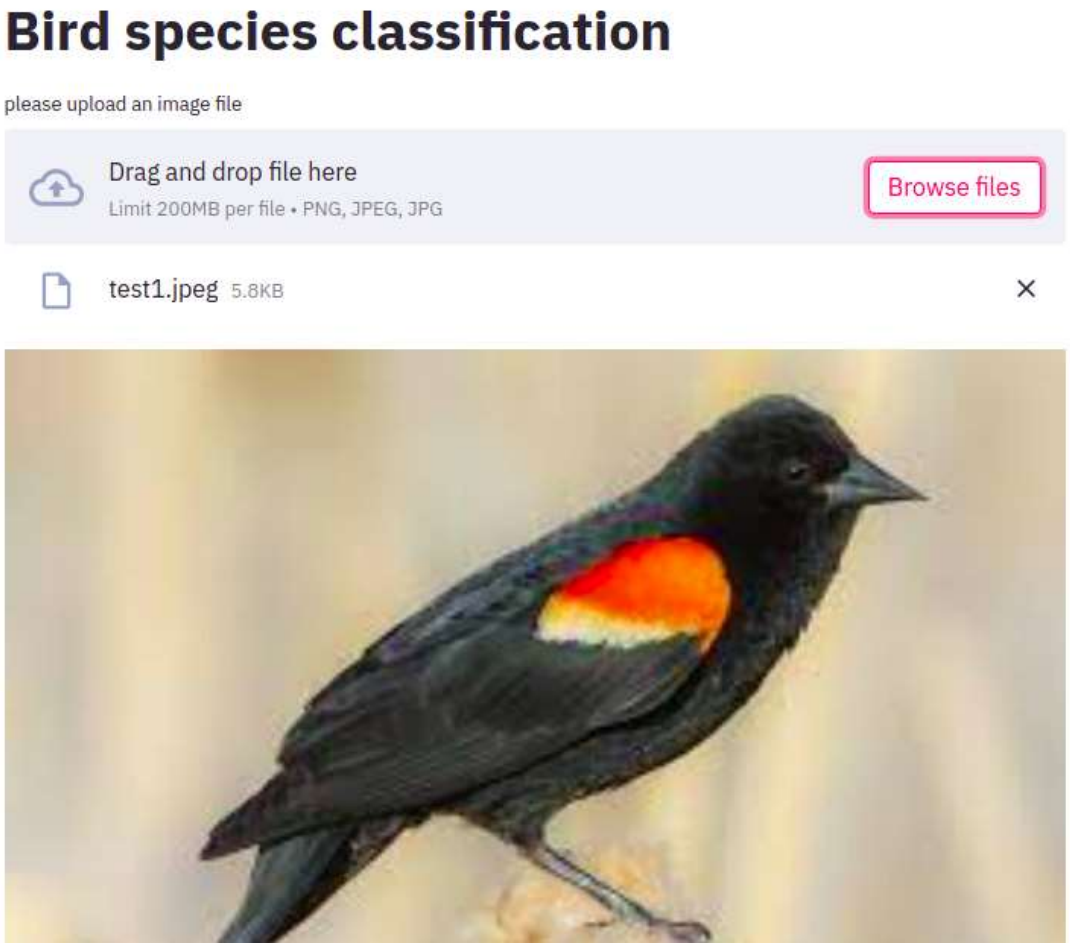
```
!npm install localtunnel
```

```
!npm install -g nmp
```

```
+ nmp@1.0.3
added 1 package from 1 contributor in 1.476s
```

```
!npx localtunnel --port 8501
```

Using above lines of codes, first work related classification task has been done and then after running the streamlit app, it has been tunneled into a local server from where we can give input and get our output.



The above screenshot shows how the web app looks in local-host server after taking input.

RESULTS AND DISCUSSIONS

```
[[0.00000016 0.00000051 0.00000794 0.00163547 0.00000014 0.00000089
  0.0000317  0.00129586 0.99700207 0.00002516]]
```

So the model predicted that the new input image belongs to class 9 with probability 0.997 which means that according to the labelling the bird image is from 'Red winged Black bird' species with probability 0.997. This result is impressively correct as we know that the image is actually from 'Red winged Black bird' species only.
In the above section the input was taken using simple python code but next section we will present the result where we have taken the input using a web server.

Image is from Red winged Blackbird species

	1	2	3	4	5	6	7	8	9
0	0.0000	0.0002	0.0004	0.0000	0.0000	0.0000	0.0004	0.9989	0.0000

Here also we get the same classification result but this one is easy to use for everyone.

So for the above mentioned 'Red winged Black bird' image our model performed well. In addition to that accuracy and confusion matrices are also calculated to get more information about the performance of the model.

Accuracy per class

CLASS	ACCURACY
Black footed...	1.00
Laysan Albat...	0.75
Sooty Albatr...	0.75
Groove bille...	1.00
Least Auklet	0.80
Parakeet Auk...	0.60
Rhinoceros A...	0.60
Brewer Black...	0.50
Red winged B...	1.00
Crested Aukl...	0.60

FIG: Accuracy per class table

From the above accuracy per class table, we can see that some classes are having more accuracy than other classes. The reason is that those bird species are having some inter species similarities as there present some genetic inter-link between some species and the model failed to classify them properly. To solve this problem, we need to make a larger dataset containing more images of those bird species and use more filters so that model can learn properly about each species.

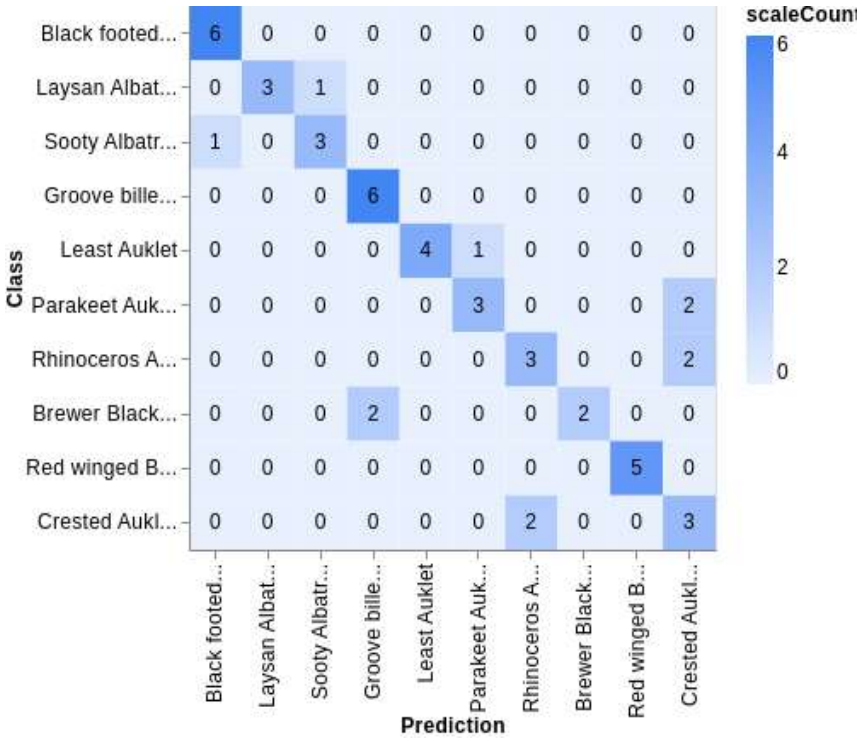


FIG: Confusion matrix

From the confusion matrix also it is clear that for some species the model mis-classified the data. So some more improvement is needed; for that a larger set of test and train data-sets are required and may be some more layers can be added on top of 'sequential-2' layer to take care of all those species who are having inter species similarities. Loss vs epochs and Accuracy vs epochs graphs are also plotted to get a better understanding of the improvement of the model with time. In blue lines the train data set accuracy and loss and in orange line test dataset accuracy and loss are plotted against epochs.

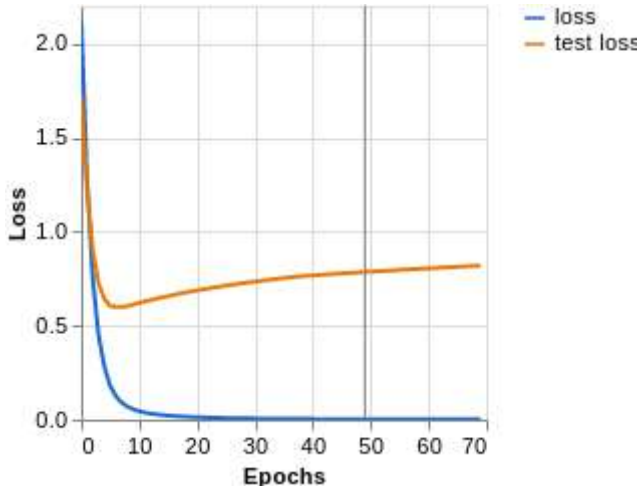


FIG: Loss vs epochs graph

Loss vs epochs graph gives us a snapshot of the training process and the direction in which the network learns. From the above graph we can see that The plot of test loss decreases to a point and begins increasing again which demonstrates a case of overfitting.

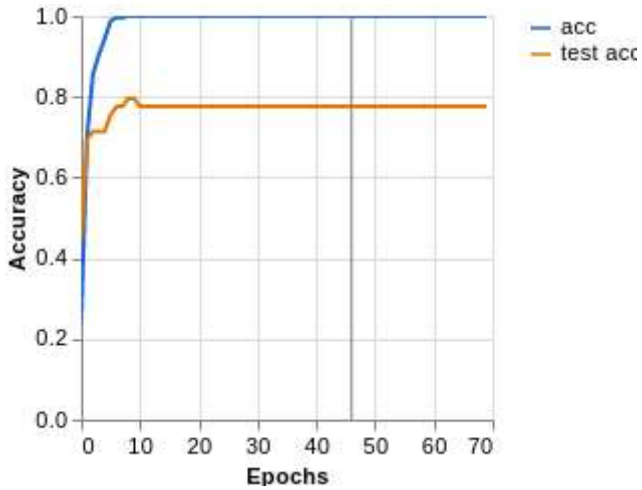


FIG: Accuracy vs epochs graph

The gap between training and test accuracy is an indication of overfitting, here the gap is present but not too much larger. So some

amount of overfitting is present which can be removed by improving the model.

CONCLUSIONS

This work presents the development of an automated model using deep learning techniques like CNN and transfer learning to classify 10 different bird species. So using this model it will be easier to classify different bird species and a web app is also created which increases its accessibility. This model works well for some species and needs some improvement for those species which are having inter species similarities. So some more improvement of this model is needed to overcome the overfitting problem and mis-classification problem. For lack of the availability of bird images the entire dataset used here consists of a small number of images which can contribute to this overfitting problem, so a larger data-set may be able to increase the efficiency of this model.

REFERENCES

1) https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/
2) https://www.ijert.org/bird-species-identification-using-deep-learning#:~:text=200%20%5BCUB%2D200%2D2011%5D)%20shows%20that%20algorithm%20achieves,using%20a%20Tensor%20flow%20library.&text=algorithms%20that%20carries%20out%20such%20a%20task%20in%20an%20automatic%20fashion.
3) https://towardsdatascience.com/adventures-in-pytorch-image-classification-with-caltech-birds-200-part-1-the-dataset-6e5433e9897c
4) https://www.geeksforgeeks.org/image-classification-using-googles-teachable-machine/
5) https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939#:~:text=A%20Convolutional%20Neural%20Network%2C%20also,topology%2C%20such%20as%20an%20image.&text=Each%20neuron%20works%20in%20its,cover%20the%20entire%20visual%20field.
6) https://tealfeed.com/deploying-image-classification-web-app-python-ae085