BIRD SPECIES IDENTIFICATION

Submitted by

SUSHANTH M (RA2011026010202)

SREYA K (RA2011026010205)

KAVETI ANJALI (RA2011026010189)

Under the Guidance of

Dr. U.SAKTHI

Assistant Professor, Department of Computational Intelligence

In partial satisfaction of the requirements for the degree of

BACHELORS OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

with specialization in Artificial Intelligence & Machine Learning

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

MAY 2023

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University *u/s 3 of UGC Act, 1956*

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this Course Project Report titled "**BIRD SPECIES IDENTIFICATION** " is the bonafide work done by **SREYA K (RA2011026010205, SUSHANTH M(RA2011026010202) AND ANJALI K(RA2011026010189)** who carried out under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

Sai 21/5/23

SIGNATURE
Faculty In-Charge
Dr.U.Sakthi
Assistant Professor
Department of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

SIGNATURE
HEAD OF THE DEPARTMENT
Dr. R Annie Uthra
Professor and Head,
Department of Computational Intelligence,
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.We would like to express my heartfelt thanks to the Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project.

We are highly thankful to our Course project Faculty **Dr.U.Sakthi , Assistant Professor, CINTEL,** for her assistance, timely suggestion and guidance throughout the duration of this course project.We extend my gratitude to our **HoD Dr. R Annie Uthra, Professor and Head, CINTEL** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project

TABLE OF CONTENTS

1. ABSTRACT

In our world, there are above 9000 bird species. Some bird species are being found rarely and if found also prediction becomes very difficult. In order to overcome this problem, we have an effective and simple way to recognize these bird species based on their features. Also, the human ability to recognize the birds through the images is more understandable than audio recognition. So, we have used Convolutional Neural Networks (CNN). CNN's are the strong assemblage of machine learning which have proven efficient in image processing. In this paper, a CNN system classifying bird species is presented and uses the Caltech-UCSD Birds 200 [CUB-200-2011] dataset for training as well as testing purposes. By establishing this dataset and using the algorithm of similarity comparison, this system is proved to achieve good results in practice. By using this method, everyone can easily identify the name of the particular bird which they want to know

Keywords: Bird species, Machine Learning, Convolutional Neural Networks

CHAPTER 2 INTRODUCTION

Bird behavior and populace patterns have become a significant issue nowadays. Birds help us to recognize different life forms on the earth effectively as they react rapidly to ecological changes. Be that as it may, assembling and gathering data about bird species requires immense human exertion just as it turns into an extremely costly technique. In such a case, a solid framework that will give enormous scale preparation of data about birds and will fill in as a significant apparatus for scientists, legislative offices, and so forth is required. In this way, bird species distinguishing proof assumes a significant job in recognizing that a specific picture of birds has a place with which categories. Bird species identification means predicting the bird species belongs to which category by using an image.

The recognition of bird species can be possible through a picture, audio or video. An audio processing method makes it conceivable to recognize by catching the sound sign of different birds. Be that as it may, because of the blended sounds in condition, for example, creepy crawlies, objects from the real world, and so forth handling of such data turns out to be progressively convoluted. Normally, people discover images more effectively than sounds or recordings. So, an approach to classify birds using an image over audio or video is preferred. Bird species identification is a challenging task to humans as well as to computational procedures that carry out such a task in an automated fashion.

As image-based classification systems are improving the task of classifying, objects are moving into datasets with far more categories such as Caltech-UCSD. Recent work has seen much success in this area. Caltech UCSD Birds 200(CUB-200-2011) is a wellknown dataset for bird images with photos of 200 categories. The dataset contains birds that are mostly found in Northern America. Caltech-UCSD Birds 200 consists of 11,788 images and annotations like 15 Part Locations, 312 Binary Attributes, 1 Bounding Box.

In this project, rather than recognizing an oversized number of disparate categories, the matter of recognizing an oversized number of classes within one category is investigated – that of birds. Classifying birds pose an additional challenge over categories, as a result of the massive similarity between classes. Additionally, birds are non-rigid objects which will deform in many ways and consequently there's also an oversized variation within classes. Previous work on bird classification has taken care of a little number of classes, or through voice.

CHAPTER 3

LITERATURE SURVEY

2.1. Introduction:

In particular, it was discovered that ecologists monitor them to determine the factors causing population fluctuation and to help in conserving and managing threatened and endangered species. The various surveys used in counting bird species including data collection techniques were succinctly reviewed. It was established that a small but growing number of researchers have studied the use of computer vision for monitoring species of birds.

This chapter evaluates reports of studies found in literature that are related to monitoring and classification of species. In particular, it focuses on reviewing bird techniques used for these species that are often similar, and motion features which this research seeks to investigate for classification of birds. First techniques which perform classification using single images were explored. This was done by reviewing them separately as those that are used for classification of bird species and those for other species, specifically bats.

2.2. Survey of different classification methods: John Martinsson et al (2017) [1], presented the CNN algorithm and deep residual neural networks to detect an image in two ways i.e., based on feature extraction and signal classification. They did an experimental analysis for datasets consisting of different images. But their work didn't consider the background species. In Order to identify the background species larger volumes of training data are required, which may not be available.

Juha Niemi, Juha T Tanttu et al (2018) [2], proposed a Convolutional neural network trained with deep learning algorithms for image classification. It also proposed a data augmentation method in which images are converted and rotated in accordance with the desired color. The final identification is based on a fusion of parameters provided by the radar and predictions of the image classifier.

Li Jian, Zhang Lei et al (2014)[3], proposed an effective automatic bird species identification based on the analysis of image features. Used the database of standard images and the algorithm of similarity comparisons.

Madhuri A. Tayal, Atharva Magrulkar et al (2018)[4], developed a software application that is used to simplify the bird identification process. This bird identification software takes an image as an input and gives the identity of the bird as an output. The technology used is transfer learning and MATLAB for the identification process.

Andreia Marini, Jacques Facon et al (2013) [5], proposed a novel approach based on color features extracted from unconstrained images, applying a color segmentation algorithm in an attempt to eliminate background elements and to delimit candidate regions where the bird may be present within the image. Aggregation processing was employed to reduce the number of intervals of the histograms to a fixed number of bins. In this paper, the authors

experimented with the CUB-200 dataset and results show that this technique is more accurate.

Marcelo T. Lopes, Lucas L. Gioppo et al (2011) [6], focused on the automatic identification of bird species from their audio recorded song. Here the authors dealt with the bird species identification problem using signal processing and machine learning techniques with the MARSYAS feature set. Presented a series of experiments conducted in a database composed of bird songs from 75 species out of which problem obtained in performance with 12 species.

Peter Jancovic and Munevver Kokuer et al (2012) [7], investigated acoustic modelling for recognition of bird species from audio field recordings. Developed a hybrid deep neural network hidden Markov model (DNN-HMM). The developed models were employed for bird species identification, detection of specific species and recognition of multiple bird species vocalising in a given recording. In this paper, the authors achieved an identification accuracy of 98.7% and recognition accuracy of 97.3%.

Mario Lasseck et al (2013) [8], presented deep convolutional neural networks and data augmentation techniques for audio-based bird species identification. In this paper, the author used the Xeno-Canto set of audio recordings of bird species.

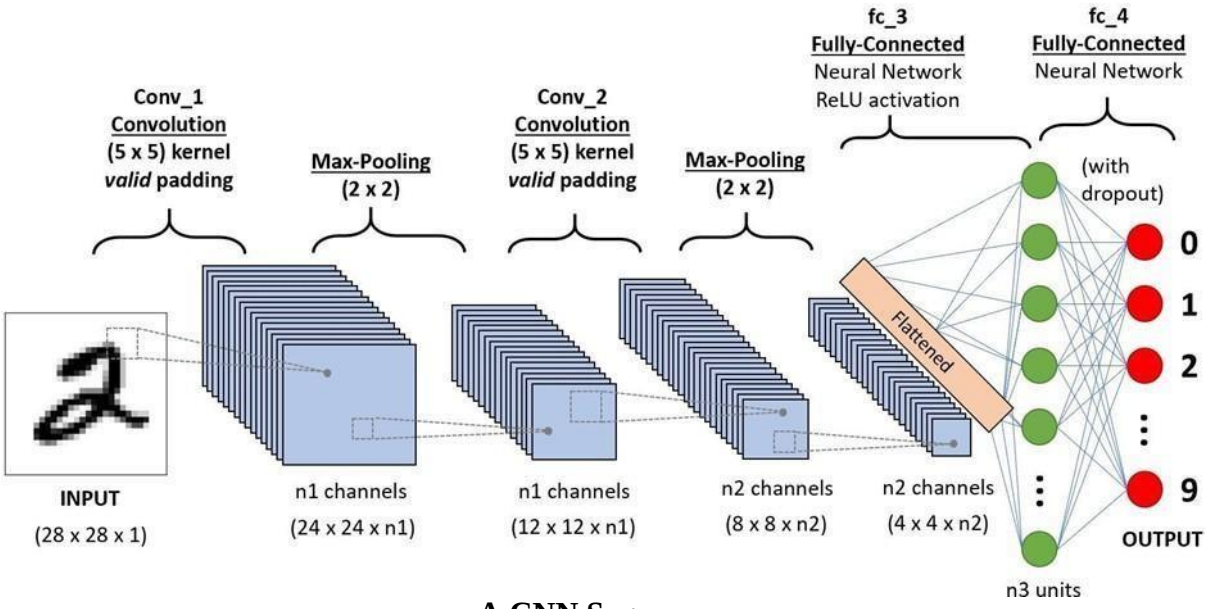
2.3. Existing System:

To identify the bird species there are many websites that produce the results using different technologies. But the results are not accurate. For suppose if we will give an input in those websites and android applications it gives us multiple results instead of a single bird name. It shows us all bird names which have similar characteristics. So, we aimed to develop a project to produce better and accurate results. In order to achieve this, we have used Convolutional Neural Networks to classify the bird species.

CHAPTER 4

SYSTEM ARCHITECTURE AND DESIGN

ARCHITECTURE:



A CNN Sequence

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 primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction**.**

CHAPTER 5

METHODOLOGY

Convolution Layer:

The convolutional layer is the core constructing block of a CNN. The convolution layer comprises a set of independent feature detectors. Each Feature map is independently convolved with the images.

Pooling Layer:

The pooling layer feature is to progressively reduce the spatial size of the illustration to reduce the wide variety of parameters and computation in the network. The pooling layer operates on each function map independently. The approaches used in pooling are:

· Max Pooling

·Mean Pooling

·Sum Pooling

Why to use pooling layers:

Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network.

The pooling layer summarises the features present in a region of the feature map generated by a convolution layer. So, further operations are performed on summarised features instead of precisely positioned features generated by the convolution layer. This makes the model more robust to variations in the position of the features in the input image.

Why to use pooling layers:

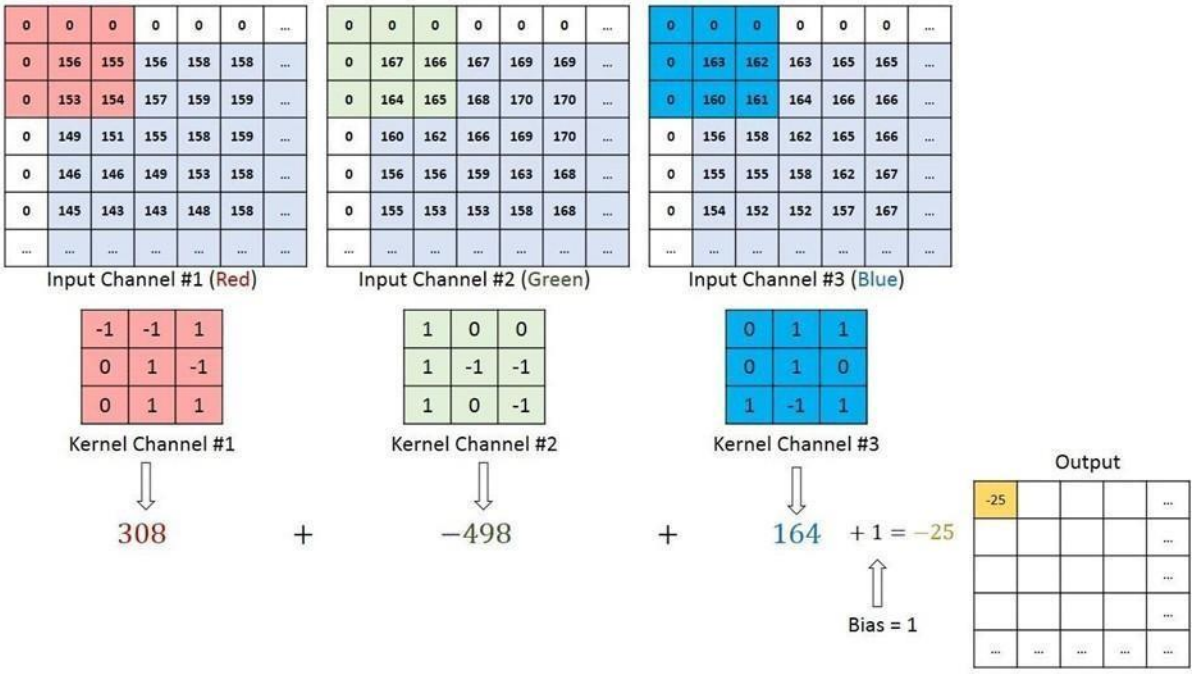
Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network. The pooling layer summarises the features present in a region of the feature map generated by a convolution layer. So, further operations are performed on summarised features instead of precisely positioned features generated by the convolution layer. This makes the model more robust to variations in the position of the features in the input image.

Convolution Layer

Filters (Convolution Kernels)

A filter (or kernel) is an integral component of the layered architecture.
Generally, it refers to an operator applied to the entirety of the image such that it transforms the information encoded in the pixels. In practice, however, a kernel is a smaller-sized matrix in comparison to the input dimensions of the image, that consists of real valued entries.

The real values of the kernel matrix change with each learning iteration over the training set, indicating that the network is learning to identify which regions are of significance for extracting features from the data.



Convolution operation with kernel

We convoluted a 5x5x1 image with a 3x3x1 kernel (which changes each iteration to extract significant features) to get a 3x3x1 convolved feature. The filter moves to the right with a certain stride value till it parses the complete width. In case of images with multiple channels (e.g. RGB) the kernel has the same depth as that of the input image. Matrix multiplication is performed between Kn and In

stack ([K1, I1]; [K2, I2]; [K3, I3]) and all results are summed with bias to give us a squashed 1 depth channel convoluted feature output.

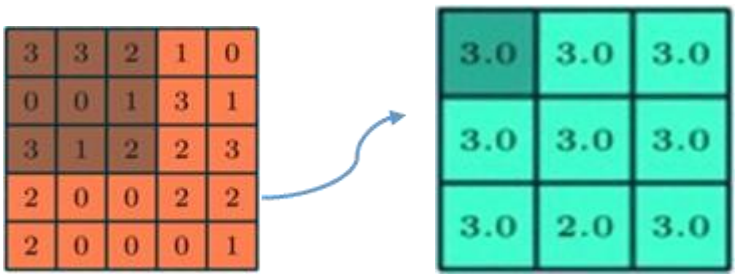
The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image, the first Convolutional Layer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc.

Kernel Size -3x3

Pooling Matrix 2x2

Image will be Resized to - (Height=250, Width=250)

Pooling Layer:



(a) Convoluted Output (b) Pooling Output

Performing Pooling operation

The Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training the model.

There are three types of Pooling: Max Pooling, Average Pooling and Global Pooling.

Max Pooling: Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max- pooling layer would be a feature map containing the most prominent features of the previous feature map. In Fig.
3 we perform maximum pooling operation by considering convoluted feature output obtained from the convolution layer.

Average Pooling: Average pooling computes the average of the elements present in the region of the feature map covered by the filter. Thus, while max pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the average of features present in a patch.

Global Pooling: Global pooling reduces each channel in the feature map to a single value. Thus, an nh x nw x nc feature map is reduced to 1 x 1 x nc feature map. This is equivalent to using a filter of dimensions nh x nw i.e. the dimensions of the feature map. Further, it can be either global max pooling or global average pooling.

Training:

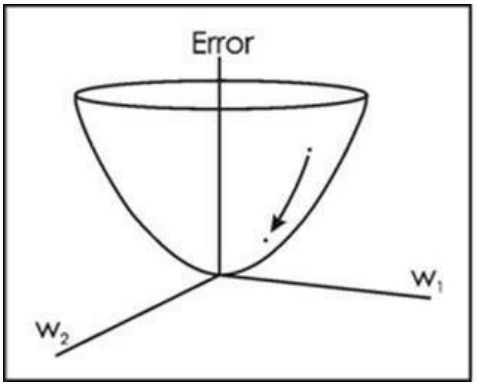
The way the computer is able to adjust its filter values (or weights) is through a training process called **backpropagation**.

Before we get into backpropagation, we must first take a step back and talk about what a neural network needs in order to work. At the moment we all were born, our minds were fresh. We didn't know what a cat or dog or bird was. In a similar sort of way, before the CNN starts, the weights or filter values are randomized. The filters don't know to look for edges and curves. The filters in the higher layers don't know to look for paws and beaks. As we grew older however, our parents and teachers showed us different pictures and images and gave us a corresponding label. This idea of being given an image and a label is the training process that CNNs go through. Before getting too into it, let's just say that we have a training set that has thousands of images of dogs, cats, and birds and each of the images has a label of what animal that picture is. Back to backprop.

So backpropagation can be separated into 4 distinct sections, the forward pass, the loss function, the backward pass, and the weight update. During the **forward pass**, you take a training image which as we remember is a 32 x 32 x 3 array of numbers and pass it through the whole network. On our first training example, since all of the weights or filter values were randomly initialized, the output will probably be something like [.1.1] particular. The network, with its current weights, isn't able to look for those low level features or thus isn't able to make any reasonable conclusion about what the classification might be. This goes to the **loss function** part of backpropagation. Remember that what we are using right now is training data. This data has both an image and a label. Let's say for example that the first training image inputted was a 3. The label for the image would be [0 0 0 1 0 0 0 0 0 0]. A loss function can be defined in many different ways but a common one is MSE (mean squared error), which is ½ times (actual-predicted) squared.

$$\mathbf{Etotal} = \sum \mathbf{1/2(target - output)^2}$$

Let's say the variable L is equal to that value. As you can imagine, the loss will be extremely high for the first couple of training images. Now, let's just think about this intuitively. We want to get to a point where the predicted label (output of the ConvNet) is the same as the training label (This means that our network got its prediction right).In order to get there, we want to minimise the amount of loss we have. Visualising this as just an optimization problem in calculus, we want to find out which inputs (weights in our case) most directly contributed to the loss (or error) of the network.



One way of visualizing this idea of minimizing the loss is to consider a 3-D graph where the weights of the neural net (there are obviously more than 2 weights, but let's go for simplicity) are the independent variables and the dependent variable is the loss. The task of minimizing the loss involves trying to adjust the weights so that the loss decreases. In visual terms, we want to get to the lowest point in our bowl shaped object. To do this, we have to take a derivative of the loss (visual terms: calculate the slope in every direction) with respect to the weights.

3-D Graph of Neural Net

This is the mathematical equivalent of a **dL/dW** where W are the weights at a particular layer. Now, what we want to do is perform a **backward pass** through the network, which is determining which weights contributed most to the loss and finding ways to adjust them so that the loss decreases.

Once we compute this derivative, we then go to the last step which is the **weight update**. This is where we take all the weights of the filters and update them so that they change in the opposite direction of the gradient.

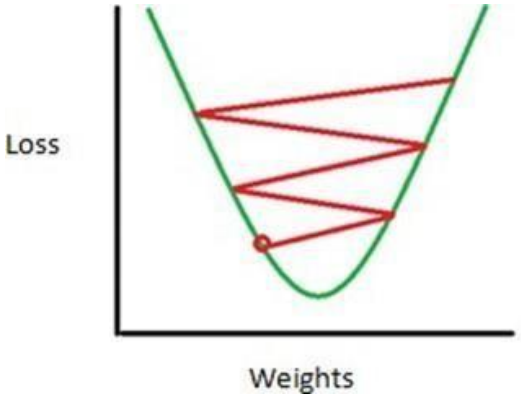
$$W = W_i - \eta \, (dL/dW)$$

where W = Weight

W_i = Initial Weight η

= Learning Rate

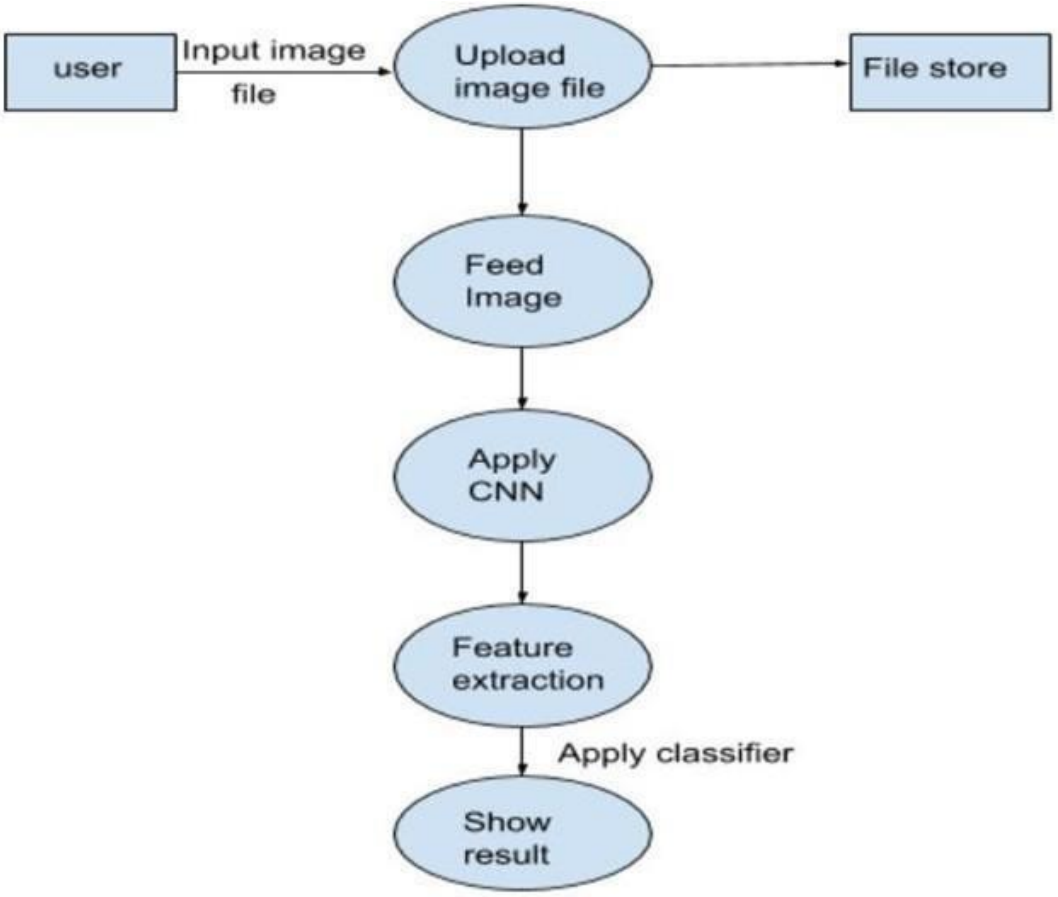
The **learning rate** is a parameter that is chosen by the programmer. A high learning rate means that bigger steps are taken in the weight updates and thus, it may take less time for the model to converge on an optimal set of weights.
However, a learning rate that is too high could result in jumps that are too large and not precise enough to reach the optimal point.



Consequence of a high learning rate where the jumps are too large and we are not able to minimize the loss.

The process of forward pass, loss function, backward pass, and parameter update is one training iteration. The program will repeat this process for a fixed number of iterations for each set of training images (commonly called a batch). Once you finish the parameter update on the last training example, hopefully the network should be trained well enough so that the weights of the layers are tuned correctly.

Flow of System:



Flow of System

CHAPTER 6

CODING AND TESTING

Sample code :-

1.System Training :

```
import numpy as
np import pandas
as pd
from keras.preprocessing.image import ImageDataGenerator, load_img from
keras.utils import to_categorical
import matplotlib.pyplot as
plt import random
FAST_RUN=False
IMAGE_WIDTH=128
IMAGE_HEIGHT=128

IMAGE_SIZE=(IMAGE_WIDTH, IMAGE_HEIGHT)
 IMAGE_CHANNELS=3 # RGB color
 from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten,
Dense, Activation, BatchNormalization
model = Sequential()

model.add(Conv2D(32, (3, 3),
activation='relu',
```

```
input_shape=(IMAGE_WIDTH,
IMAGE_HEIGHT,
IMAGE_CHANNELS)))
model.add(BatchNormalization()) model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25)) model.add(Conv2D(64, (3,
3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25)) model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
```

```
model.add(Dense(200,
activation='softmax'))
model.compile(loss='categorical_c
rossentropy', optimizer='rmsprop',
metrics=['accuracy'])

from keras.callbacks import EarlyStopping, ReduceLROnPlateau
earlystop = EarlyStopping(patience=10)
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
patience=2,
verbose=1,

factor=0.5, min_lr=0.00001)
callbacks = [earlystop, learning_rate_reduction] train_datagen =
ImageDataGenerator( rotation_range=15,
rescale=1./255,
shear_range=0.1,
zoom_range=0.2,
horizontal_flip=True,
width_shift_range=0.1,
height_shift_range=0.1)
 test_datagen = ImageDataGenerator(rescale=1./255)
```

```
x_train=train_datagen.flow_from_directory(r'C:\Users\user\Desktop\datas
et\traindata',target_size = (128, 128), batch_size = 32, class_mode = 'categorical')

x_test=test_datagen.flow_from_directory(r'C:\Users\user\Desktop\dataset\t
estdata', target_size = (128, 128),
batch_size = 32,
class_mode = 'categorical')
print(x_train.class_indices)

 model.fit_generator(x_train,

steps_per_epoch =130 ,

 epochs = 15, validation_data = x_test, validation_steps =

20,callbacks=callbacks) model.save("mybird.h5")

from keras.models import load_model
```

Matplotlib.pyplot:

Matplotlib.pyplot is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. In matplotlib.pylot various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes (please note that "axes" here and in most places in the documentation refers to the axes part of a figure and not the strict mathematical term for more than one axis).

Model = Sequential ():

The sequential model is a linear stack of layers. You can create a sequential model by passing a list of layer instances to the constructor:

```
from keras.models import Sequential

from keras.layers import Dense,
Activation model =
Sequential([Dense(32,
input_shape=(784,)),
```

```
Activation('relu')Dense(10),
Activation('softmax'),])
```

You can also simply add layers via the .add() method:

```
model = Sequential()
model.add(Dense(32,
input_dim=784))
model.add(Activation('relu'))
```

You can save a model built with the Functional API into a single file. You can later recreate the same model from this file, even if you no longer have access to the code that created the model.

This file includes:

The model's architure

The model's weight values (which were learned during training) The

model's training config (what you passed to compile), if any

The optimizer and its state, if any (this enables you to restart training where you left)

Batch Normalization ():

One possible reason for this difficulty is the distribution of the inputs to layers deep in the network may change after each mini-batch when the weights are updated. This can cause the learning algorithm to forever chase a moving target. This change in the distribution of inputs to layers in the network is referred to the technical name "*internal covariate shift*."

Batch normalization is a technique for training very deep neural networks 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 training epochs required to train deep networks.

Conv2D ():

This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If use_bias is True, a bias vector is created and added to the outputs. Finally, if activation is not None, it is applied to the outputs as well.

ReduceLROnPlateau ():

Reduce learning rate when a metric has stopped improving. Models often benefit from call num ning rate by a factor of 2-10 once learning stagnates. This quantity and if no improvement is seen for a 'patience' learning rate is reduced.

Dropout ():

Dropout is a technique where randomly selected neurons are ignored during training. They are “dropped-out” randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass.

ImageDataGenerator ():

Data augmentation is what Kera’s’ “ImageDataGenerator” class implements. Using this type of data augmentation, we want to ensure that our network, when trained, sees new variations of our data at each and every epoch.

1. An input batch of images is presented to the ImageDataGenerator.
2. The ImageDataGenerator transforms each image in the batch by a series of random translations, rotations, etc.
3. The randomly transformed batch is then returned to the calling function.

2. Extraction of Images:

```
from keras.models import load_model import numpy as np import cv2

model= load_model('mybird.h5')

model.compile(optimizer='adam',loss='categorical_crossentropy',metrics
=[ 'accuracy'])
```

```
from skimage.transform import resize def detect(frame):

try:img = resize(frame,(128,128)) img=np.expand_dims(img,axis=0) if(np.max(img)>1):

img= img/255.0
prediction

=model.predict(img)

print(prediction) prediction_class=

model.predict_classes(img)

print(prediction_class) except

AttributeError:

print("Shape not

found")

from matplotlib.pyplot
import *
frame=cv2.imread(r'C:\Users\user\Desktop\dataset\testdata\002.Painted
_B unting\Painted_Bunting_0091_15198.jpg')#path of image imshow(fram

e) data=

detect(frame)


frame=cv2.imread(r'C:\Users\user\Desktop\dataset\testdata\009.Brewer
_Bl ackbird\Brewer_Blackbird_0112_2340.jpg')#path of image
```

imshow(fram e)

data=

detect(frame)

frame=cv2.imread(r'C:\Users\user\Desktop\dataset\testdata\050.Eared_ Gre
be\Eared_Grebe_0088_34067.jpg')#path of image

imshow(fram e)

data=

detect(frame)

CV2:

OpenCV, which is an image and video processing library with bindings in C++, C, Python, and Java. OpenCV is used for all sorts of image and video analysis, like facial recognition and detection, license plate reading, photo editing, advanced robotic vision, optical character recognition, and a whole lot more.

imRead ():

cv2.imread() method loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix.

imShow ():

cv2.imshow() method is used to display an image in a window. The window automatically fits to the image size.

3. gui.py:

import numpy as np

```
from keras.preprocessing import image
from tkinter import *
from PIL import ImageTk,
Image from tkinter import
filedialog import os

from keras.models import load_model
classifier =
load_model(r'C:\Users\user\Desktop\Birds-identification- master\mybird.h5')
classifier.compile(optimizer = 'adam', loss =
'categorical_crossentropy', metrics = ['accuracy'])
root = Tk()
root.geometry("550x300+300+150 ")
root.resizable(width=True, height=True) def
openfn(): filename =
filedialog.askopenfilename(title='open') return
filename
def
open_img
(): x =
openfn()
```

test_image = image.load_img(x, target_size = (128, 128))

test_image = image.img_to_array(test_image) test_image =

np.expand_dims(test_image, axis

= 0) result = classifier.predict_classes(test_image)

print(result)

index=['002.Painted_Bunting','009.Brewer_Blackbird','050.Eared_Grebe'] label = Label(root, text="Prediction : "+index[result[0]])

label.pack()

img = Image.open(x)

img = img.resize((250, 250), Image.ANTIALIAS) img = ImageTk.PhotoImage(img)

panel = Label(root, image=img)

panel.image = img panel.pack()

btn = Button(root, text='open image', command=open_img).pack() root.mainloop()

tKinter:

The tkinter package ("Tk interface") is the standard Python interface to the Tk GUI toolkit. Both Tk and tkinter are available on most Unix platforms, as well as on Windows systems. (Tk itself is not part of Python; it is maintained at Active State.)

Running python -m tkinter from the command line should open a window demonstrating a simple Tk interface, letting you know that tkinter is properly installed on your system, and also showing what version of Tcl/Tk is installed, so you can read the Tcl/Tk documentation specific to that version.

Label ():

The tkinter label widgets can be used to show text or an image to the screen. A label can only display text in a single font. The text can span multiple lines. We can put any text in a label and we can have multiple labels in a window (just like any widget can be placed multiple times in a window).

Button ():

The Button widget is used to add buttons in a Python application. These buttons can display text or images that convey the purpose of the buttons. You can attach a function or a method to a button which is called automatically when you click the button.

mainloop ():

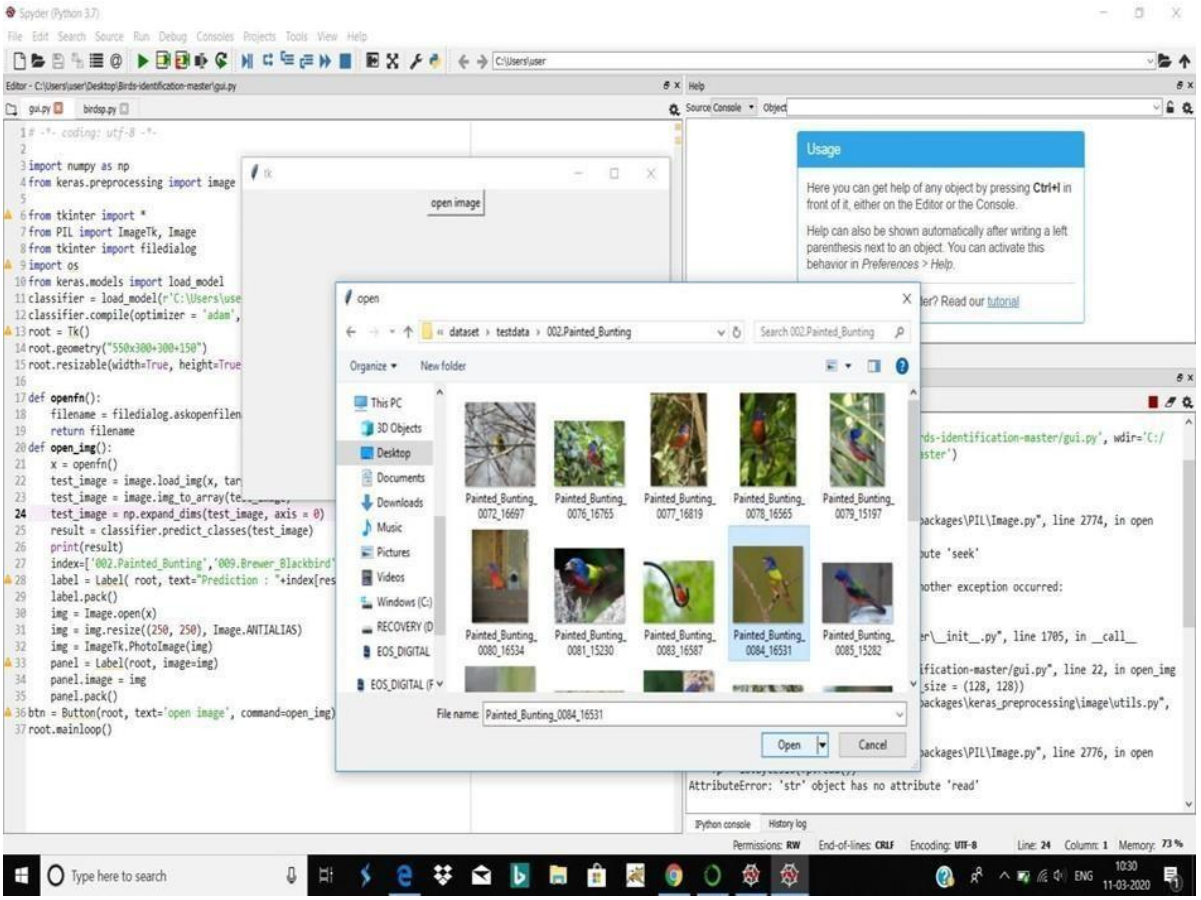
There is a method known by the name mainloop () is used when your application is ready to run. mainloop () is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.

Tested Results:

	Birdname (species)	**Tested Results**	**Accuracy (%)**
	Painted_Bunting	10/10	95%

	Brewer_Blackbird	10/10	95%
	Eared_Grebe	7/10	70%
	Total	**27/30**	**90%**

Efficiency Table



CHAPTER 7

SCREENSHOTS AND RESULTS


tk
open image
Prediction : 050.Eared_Grebe

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENTS

Conclusion:

The main idea behind developing the identification website is to build awareness regarding bird-watching, bird and their identification, especially birds found in India. It also caters to the need of simplifying the bird identification process and thus making bird-watching easier. The technology used in the experimental setup is Convolutional Neural Networks (CNN). It uses feature extraction for image recognition. The method used is good enough to extract features and classify images.

The main purpose of the project is to identify the bird species from an image given as input by the user. We used CNN because it is suitable for implementing advanced algorithms and gives good numerical precision accuracy. It is also general-purpose and scientific. We achieved an accuracy of 85%-90%. We believe this project extends a great deal of scope as the purpose meets. In wildlife research and monitoring, this concept can be implemented in-camera traps to maintain the record of wildlife movement in specific habitat and behaviour of any species.

Enhancements :

1) Create an android/iOS app instead of website which will be more convenient to user.
2) 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).

9. REFERENCES

[1] Indian Birds [Online] https://play.google.com /store/apps /details ?id = com.kokanes.birds info & hl = en

[2] Bird Watching Apps: Five Useful Apps to Get Started With Birding [Online] https://gadgets.ndtv.com/apps/features/bird-watching-apps-five-useful-apps-to-get- started-with-birding-1640679

[3] Transfer learning using Alex Net [Online] https://in.mathworks.com /help /nnet /examples / transfer-learning-using- alexnet.html

[4] Feature extraction using AlexNet [Online] https://in.mathworks.com /help /nnet /examples / feature-extraction-using-alexnet.html # d119e4167

[5] Dipta Das, Sourya & Kumar, Akash" An Exploration of Computer Vision Techniques for Bird Species Classification", December 15, 2017.

[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, CA, 2005.

[7] C. Wah et al. The Caltech-UCSD Birds-200-2011 Dataset. Tech. rep. CNS-TR-2011-001. California Institute of Technology, 2011.

[8] Thomas Berg, Jiongxin Liu et. Al " Large-scaleFine-grainedVisualCategorizationofBirds",

[9] Yuning Chai Electrical Engineering Dept. ETH Zurich,Victor Lempitsky Dept. of Engineering Science University of Oxford, Andrew Zisserman Dept. of Engineering Science University of Oxford BiCoS:A BiSegmentation Method for Image Classification.

[10] Tóth, B.P. and Czeba, B., 2016, September. Convolutional Neural Networks for Large-Scale Bird Song Classification in Noisy Environment. In CLEF (Working Notes) (pp. 560-568).

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