DIGITAL NATURALIST – AI ENABLED TOOL FOR BIODIVERSITY RESEARCHERS

A NAALAIYA THIRAN PROJECT REPORT

TEAM ID: **PNT2022TMID00699**

Submitted by

B. RAKSHANA	- 211419104212
K. SWETHA	- 211419104284
S. SWETHA	- 211419104286
K. VIJAYALAKSHMI	- 211419104303

in partialfulfillment for the award

of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

from

PANIMALAR ENGINEERING COLLEGE

(AnAutonomous Institution, Affiliated to Anna University, Chennai)

NOVEMBER – 2022

TABLE OF CONTENTS

Digital Naturalist - AI Enabled tool for Biodiversity Researchers

1. INTRODUCTION

1.1 Project Overview

The ever-growing number of digital sensors in the environment has led to an increase in the amount of digital data being generated. This includes data from satellites, weather stations, data from "internet of things" devices, and data collected by members of the public via smartphone applications, to name but a few. These new sources of data have contributed to the era of "Big Data" characterized by large volumes of data, of numerous types and quality, being generated at an increasing speed. This presents challenges and opportunities across a number of domains, including water management, camera trapping, and acoustic analysis. Automated identification of plants and animals have improved considerably in the last few years. In total, nine deep learning systems implemented by three different research teams were evaluated with regard to nine expert botanists of the French flora. Therefore, we created a small set of plant observations that were identified in the field and revised by experts in order to have a near-perfect golden standard. The main outcome of this work is that the performance of state-of-the-art deep learning models is now close to the most advanced human expertise. This shows that automated plant and animal identification systems are now mature enough for several routine tasks, and can offer very promising tools for autonomous ecological surveillance systems. Artificial intelligence (AI) techniques have profoundly transformed our ability to extract information from visual data. AI techniques have been applied for a long time in security and industrial domains, for example, in iris recognition or the detection of faulty objects in manufacturing. They were nevertheless only recently made more widely accessible after their use in smartphone apps for face recognition and song identification. Combined with increasing access to cloud-based computation, AI techniques can now automatically analyse hundreds of thousands of visual data every day. Deep learning models (some of the most advanced AI algorithms) are developed with training datasets that allow them to capture discriminant visual patterns. Their performances are then strongly correlated to the quality and completeness of the datasets on which they are trained. Unbalanced, biased, or otherwise poor-quality training datasets will lead to underperforming algorithms in real conditions. During the learning phases, particular attention must be given to any relevant limitations of the training data, and the gap between these and the test data on which the developed algorithms will be evaluated.

1.2 Purpose :

To better understand the complexities of natural ecosystems and better manage and protect them, it would be helpful to have detailed, large-scale knowledge about the number, location, and behaviours of animals in natural ecosystems. Having accurate, detailed, and up-to-date information about the location and behaviour of animals in the wild would improve our ability to study and conserve ecosystems. We investigate the ability to automatically, accurately, and inexpensively collect such data, which could help catalyze the transformation of many fields of ecology, wildlife biology, zoology, conservation biology, and animal behaviour into "big data" sciences. Motion- sensor "camera traps" enable collecting wildlife pictures inexpensively, unobtrusively, and frequently. However, extracting information from these pictures remains an expensive, time- consuming, manual task. We demonstrate that such information can be automatically extracted by deep learning, a cutting-edge type of artificial intelligence. We train deep convolutional neural networks to identify, count, and describe the behaviours of 48 species in the 3.2 million-image Snapshot Serengeti dataset. Our deep neural networks automatically identify animals with &gt;93.8% accuracy, and we expect that number to improve rapidly in years to come.

2. LITERATURE SURVEY

2.1 Existing problem

Problem Statement 1

Customer States a Problem that, "As a Naturalist, I'm trying to study the patterns of nature, identifies a different kind of flora and fauna in nature. But at some point traditional approaches may become inefficient or even impossible given the volume, diversity, and heterogeneity of these data. Because, Artificial Intelligence(AI) techniques have profoundly transformed our ability to extract information from visual data. Which makes me feel, AI can also be used to extract information from big data in order to address various challenges faced by society natural resource".

ProblemStatement 2

Customer States another Problem that,"As a Influencer, I'm trying to do the interpretation of automatically collected observation data is a popular and fast-growing research field at Biodiversity. But, automatically the collection of data will only reach its full potential if data analysis can be automated to a certain degree. Because framework will include AI models that have been trained by expert taxonomists, thus providing a highg level of accuracy. Which makes me feel biodiversity data from natural collections openly accessible and easier to analyse".

2.2 References

1. Plant identification: Experts vs. machines in the era of deep learning: deep learning techniques challenge floraexperts. Bonnet P., Goeau H., Hang S.T., Lasseck M., Sulc M., Malécot V., Jauzein P., Melet J.C., You C., Joly A..2018.

2. Plant Species Identification Using Computer Vision Techniques: A Systematic Literature Review JanaWäldchen, Patrick Mäder Published 7 January 2017

3. AI Naturalists Might Hold the Key to Unlocking Biodiversity Data in Social Media Imagery Tom A August, Oliver L Pescott, Alexis Joly, Pierre Bonnet Patterns 2020

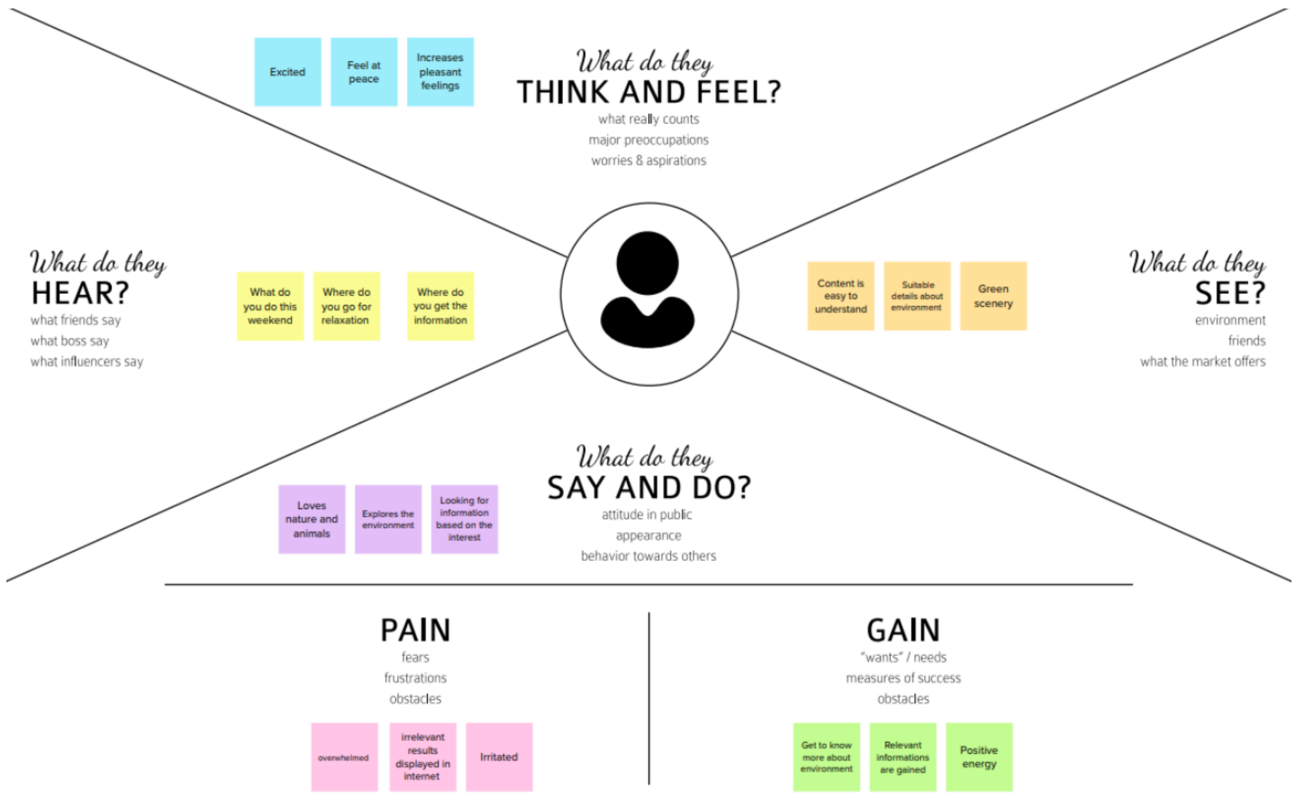
4. Automated plant identification using artificial neural networks Jonathan Y. Clark, D. Corney, H. L. Tang Computer Science 2012 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)

2.3 Problem Statement Definition

Problem to be solved is, At some point in time, traditional approaches become inefficient or even impossible given the volume, diversity, and heterogeneity of these data. Solution for the problem , this project focuses on helping researchers, naturalists, and many more people who are involved in exploring nature. It gives the species/botanical names, medicinal values, extinct/endangered species, and information about the flora and fauna to help the people who seek it. This is achieved by deep learning concepts.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

At some point traditional approaches may become inefficient, given the volume, diversity and heterogeneity of these data.

Key rules of brainstorming

To run an smooth and productive session

Stay in topic. Encourage wild ideas.

Defer judgment. Listen to others.

Go for volume. If possible, be visual.

2

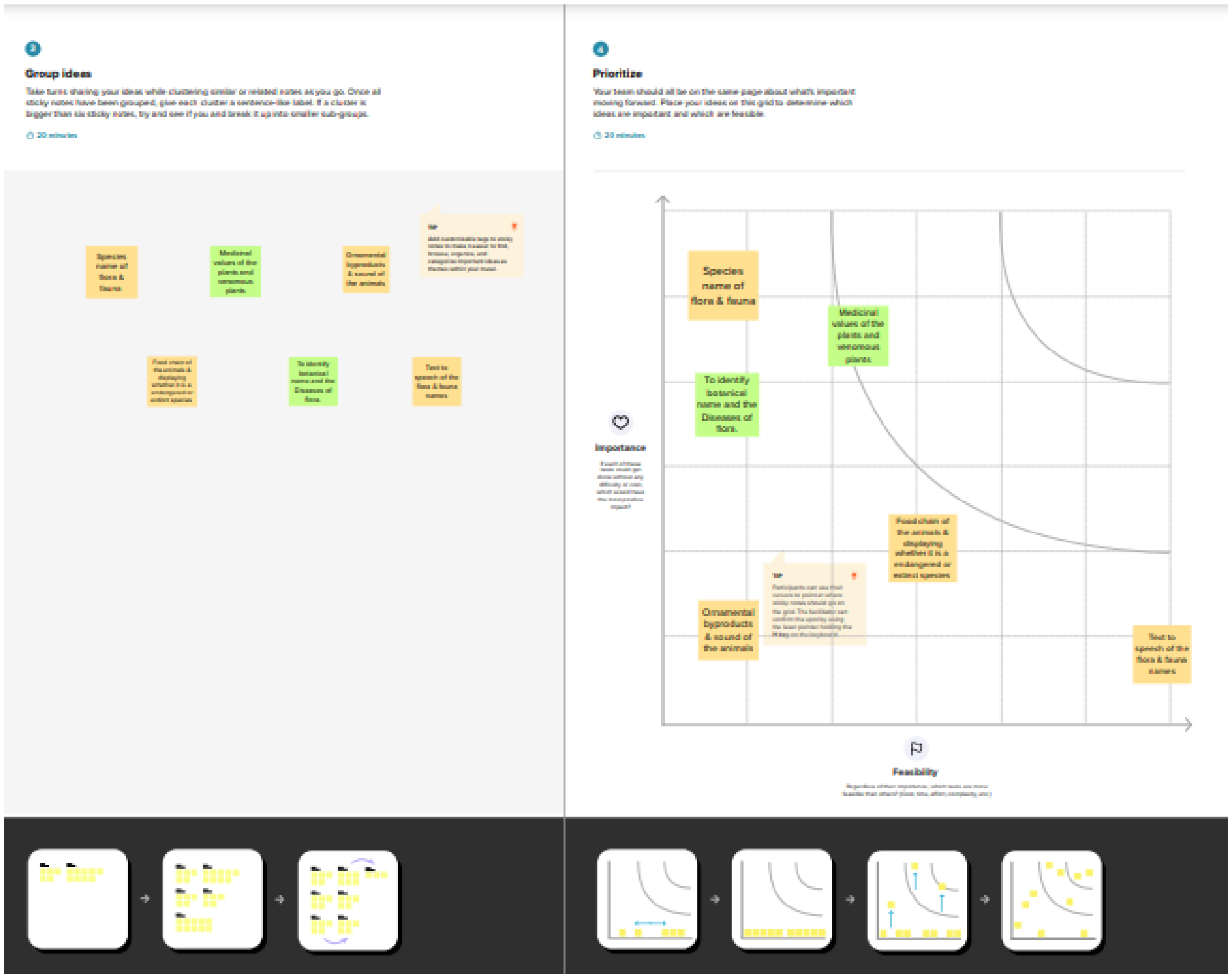
Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!



3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the H key on the keyboard.

Importance

If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

Feasibility

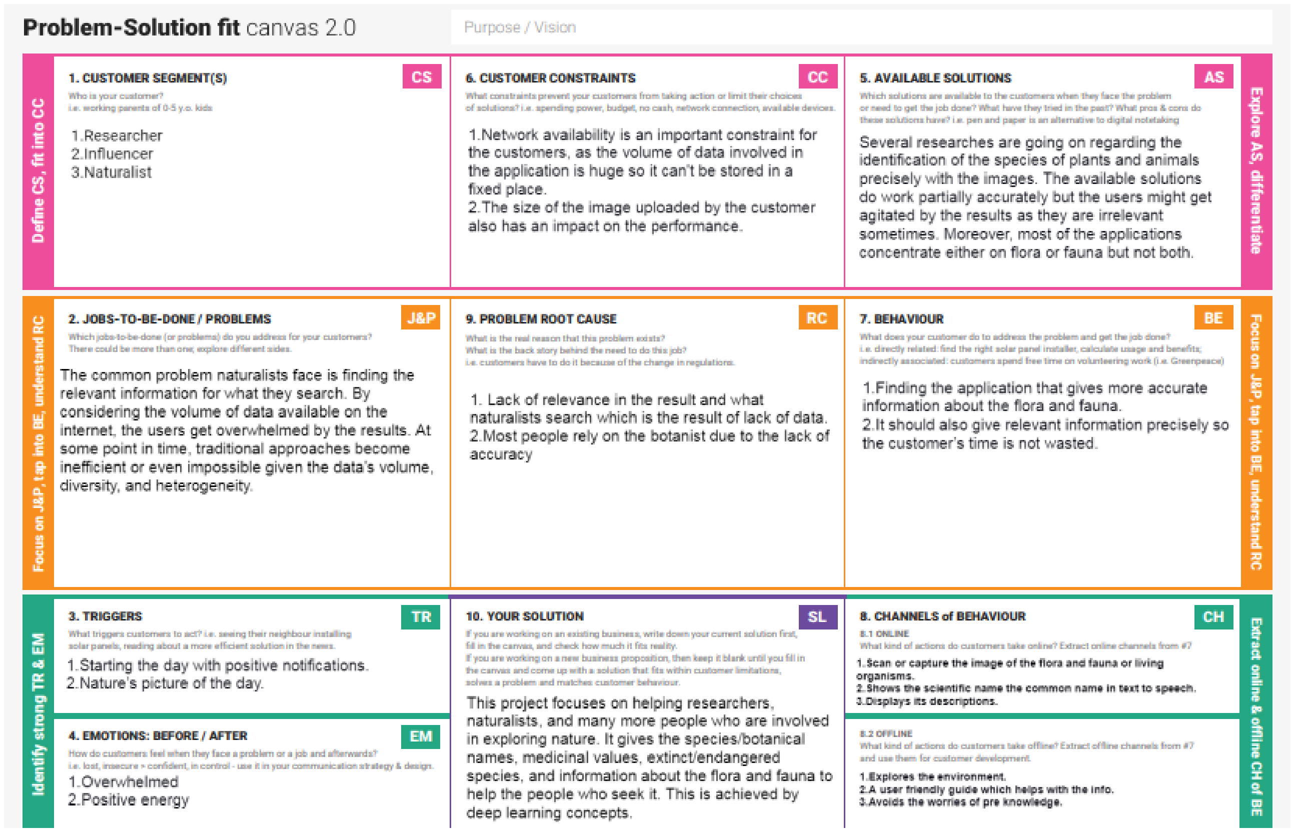
Regardless of their importance, which tasks are more feasible than others? (cost, time, effort, complexity, etc.)

Species name of flora & fauna
Medicinal values of the plants and venomous plants
Ornamental byproducts & sound of the animals
Food chain of the animals & displaying whether it is a endangered or extinct species
To identify botanical name and the Diseases of flora.
Text to speech of the flora & fauna names
Species name of flora & fauna
Medicinal values of the plants and venomous plants
To identify botanical name and the Diseases of flora.
Food chain of the animals & displaying whether it is a endangered or extinct species
Ornamental byproducts & sound of the animals
Text to speech of the flora & fauna names

3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	At some point in time, traditional approaches become inefficient or even impossible given the volume, diversity, and heterogeneity of these data.
2.	Idea / Solution description	This project focuses on helping researchers, naturalists, and many more people who are involved in exploring nature. It gives the species/botanical names, medicinal values, extinct/endangered species, and information about the flora and fauna to help the people who seek it. This is achieved by deep learning concepts.
3.	Novelty / Uniqueness	This provides information for both the flora and fauna.
4.	Social Impact / Customer Satisfaction	Being able to identify the flora and fauna around us often leads to an interest in protecting wild spaces.
5.	Business Model (Revenue Model)	It can make money through subscription-based. Partnership with many laboratories and scientists around the world.
6.	Scalability of the Solution	This application can be accessed anywhere and anytime by users with the help of the internet.

3.4 Problem Solution fit



Problem-Solution fit canvas 2.0

Purpose / Vision

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

1.Researcher
2.Influencer
3.Naturalist

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

1.Network availability is an important constraint for the customers, as the volume of data involved in the application is huge so it can't be stored in a fixed place.
2.The size of the image uploaded by the customer also has an impact on the performance.

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

Several researches are going on regarding the identification of the species of plants and animals precisely with the images. The available solutions do work partially accurately but the users might get agitated by the results as they are irrelevant sometimes. Moreover, most of the applications concentrate either on flora or fauna but not both.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

The common problem naturalists face is finding the relevant information for what they search. By considering the volume of data available on the internet, the users get overwhelmed by the results. At some point in time, traditional approaches become inefficient or even impossible given the data's volume, diversity, and heterogeneity.

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

1. Lack of relevance in the result and what naturalists search which is the result of lack of data.
2.Most people rely on the botanist due to the lack of accuracy

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

1.Finding the application that gives more accurate information about the flora and fauna.
2.It should also give relevant information precisely so the customer's time is not wasted.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

1.Starting the day with positive notifications.
2.Nature's picture of the day.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

1.Overwhelmed
2.Positive energy

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

This project focuses on helping researchers, naturalists, and many more people who are involved in exploring nature. It gives the species/botanical names, medicinal values, extinct/endangered species, and information about the flora and fauna to help the people who seek it. This is achieved by deep learning concepts.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

1.Scan or capture the image of the flora and fauna or living organisms.
2.Shows the scientific name the common name in text to speech.
3.Displays its descriptions.

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

1.Explores the environment.
2.A user friendly guide which helps with the info.
3.Avoids the worries of pre knowledge.

Extract online & offline CH of BE

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	■ Registration through Gmail ■ Registration through Mobile Number
FR-2	User Confirmation	● Confirmation via Email ● Confirmation via OTP
FR-3	Authentication	By entering the OTP sent to the Gmail or Mobile.
FR-4	Subscriptions	Transactions via ● Net Banking or ● UPI or ● Credit card.
FR-5	Administrative functions	■ Maintaining description of flora and fauna. ■ Adding the species.
FR-6	User interfaces	➤ Easy to understand. ➤ Sharing the experience with friends.

4.2 Non-functional Requirements:

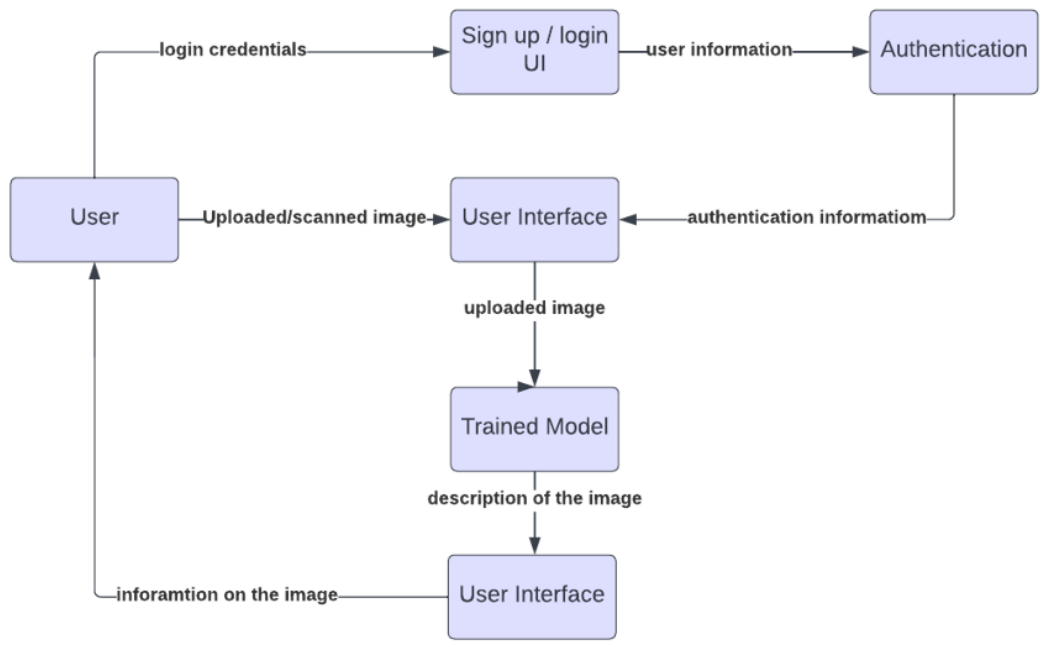
NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	This project act as a user-friendly guide to Researchers, Naturalist &amp; Tourist by predicting the image and provides useful information of flora and fauna precisely.
NFR-2	Security	● Authentication helps in maintaining the data secured. ● Subscription transactions details should be encrypted.
NFR-3	Reliability	This project provides reliability by covering the various species among different habitats.
NFR-4	Performance	■ To provide increasing in accuracy with low loss. ■ To be more efficient in prediction of flora and fauna. ■ Increased Data Augmentation
NFR-5	Availability	➤ Dataset is constantly updated. ➤ Network required for cent percent prediction.
NFR-6	Scalability	It supports many users without any issues which scaled through the cloud resources.

5. PROJECT DESIGN

5.1 Data Flow Diagrams

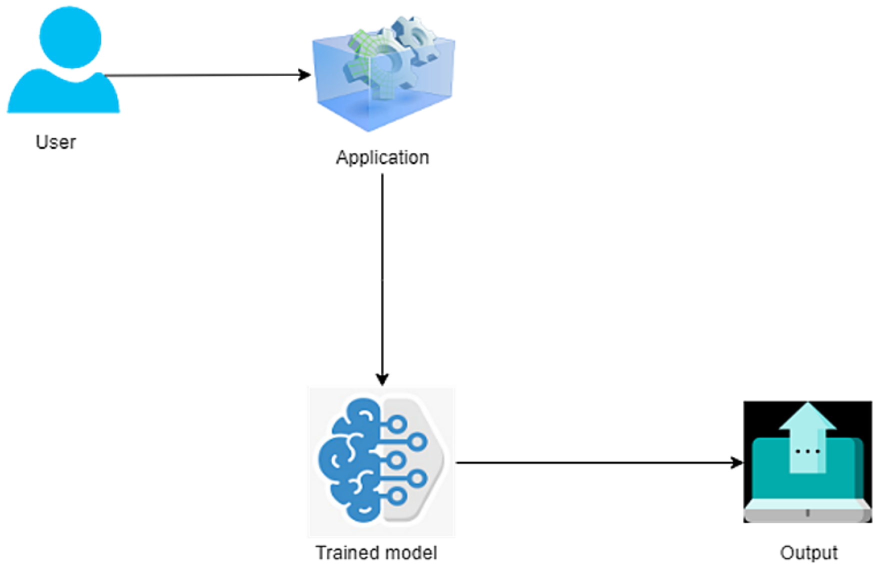
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can graphically depict the right amount of the system requirement. It shows how data enters and leaves the system, what changes the information, and where data is stored.

Data Flow Diagram:



1. The user uploads the image onto the application.
2. Application gives the user image as input to the trained model.
3. Model gives the description/information of the uploaded image.

Simplified Data Flow Diagram:



5.2 Solution & Technical Architecture

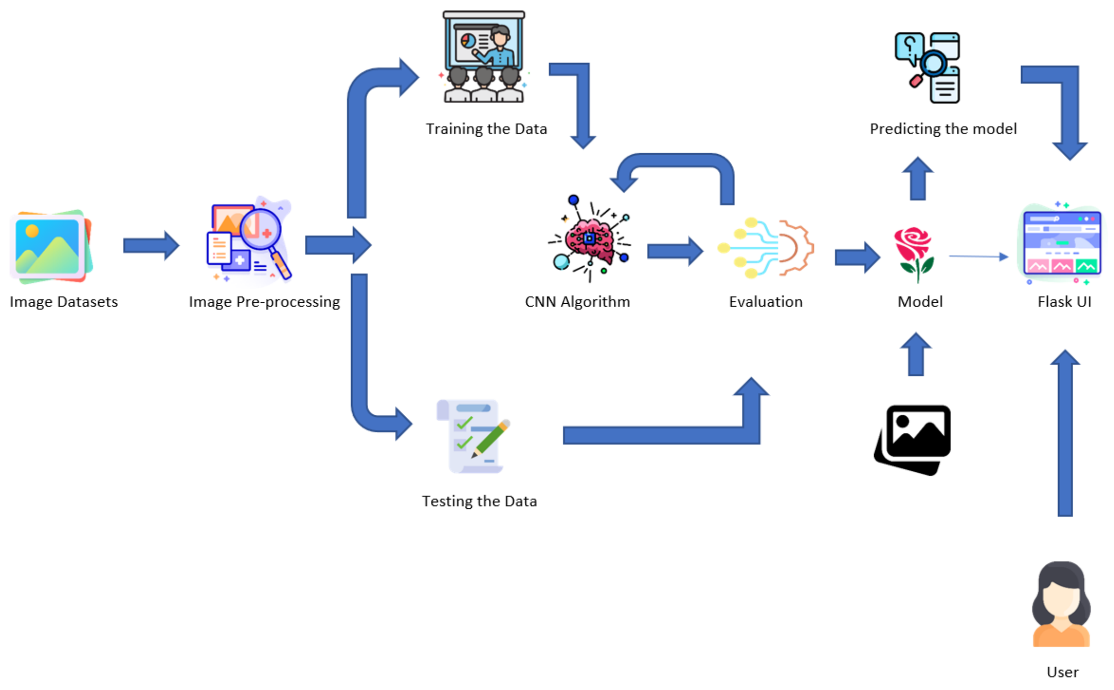


Table-1: Components & Technologies:

S. No	Component	Description	Technology
1.	User Interface	Web UI or Website	HTML, CSS, JavaScript / React JS
2.	Application Logic-1	Model building and then training the model	Python
3.	Application Logic-2	User uploads the image for the prediction	IBM Watson STT service
4.	Application Logic-3	Getting the relevant data from the database and providing to the user	IBM Watson Assistant
5.	Database	Image of all the variety species along with detailed information of each species	MySQL / NoSQL
6.	Cloud Database	Gets the data from database and feed them to model for prediction and also used to retrieve the data required for user.	IBM Cloudant, IBM DB2
7.	File Storage	User Login credentials, Images and their data, code and API keys	IBM Block Storage
8.	External API-1	To get data from the database when user gives the image as the input	IBM Storage API
9.	External API-2	To collect the username and password of the specific user	Secure Authentication API
10.	Machine Learning Model	To predict the both flora & fauna through the image which is given as input and also it gives detailed information of the particular species	Image Recognition Model(Detecting the species and identifying the model)
11.	Infrastructure (Server / Cloud)	To deploy the Application in Cloud Server	Cloud Foundry

Table-2: Application Characteristics:

S. No	**Characteristics**	**Description**	**Technology**
1.	Open-Source Frameworks	Application is built by using flask	WSGI framework (Web Server Gateway Interface)
2.	Security Implementations	To Authenticate the species data in database as well as User credentials.	SHA-256, Encryptions
3.	Scalable Architecture	To scale our application in server side by supporting clients including desktop browsers, mobile browsers etc..	IBM Auto Scaling
4.	Availability	To make application available both online and offline and also 24/7 service.	IBM Cloud load balancer
5.	Performance	Designing an application which can handle wide range of requests at a time to provide accuracy in prediction as well as without any delay in time.	IBM instance

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer or User	Sign up/Login	USN-1	As a user, I can log in/signup for the application.	I can access the application through this process.	High	Sprint-1
	Upload or scan the image	USN-2	As a user, I can upload or scan the image about which the information is needed.	I can upload the image.	High	Sprint-2
	Get information on the image	USN-3	As a user, I can get information about the image.	I can get information about the image.	High	Sprint-2

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Modelling Phase	USN-1	Data Collection and digitalizing for analyzing	3	Medium	SWETHA.K
Sprint-1		USN-2	Adding more data to avoid overfitting	3	Medium	RAKSHANA.B
Sprint-1		USN-3	Building a CNN modelusing the collected data	5	High	VIJAYALAKSHMI.K
Sprint-1		USN-4	Evaluating the model to check the accuracy and precision	4	High	SWETHA.S
Sprint-2	Development Phase	USN-5	Home page Creation – Shows the features of our application	2	Low	SWETHA.S
Sprint-2		USN-6	Setting up facilities for user to feed the image	3	Medium	RAKSHANA.B
Sprint-2		USN-7	Prediction page creation – shows prediction for the user given image	5	High	SWETHA.K
Sprint-2		USN-8	Model loading – APIcreation using flask	5	High	VIJAYALAKSHMI.K

Sprint-3	Deployment Phase	USN-9	Integrating UI & backend – Connecting the front end and backend using API calls	4	Medium	SWETHA.K
Sprint-3		USN-10	Cloud deployment – Deployment of application using IBM Cloud	5	High	VIJAYALAKSHMI.K
Sprint-4	Testing Phase	USN-11	Functional testing – Checking the scalability and robustness of the application	5	High	RAKSHANA .B
Sprint-4		USN-12	Non-Functional testing – Checking foruser acceptance and integration	5	High	SWETHA.S

Velocity:

Average Velocity = Sprint duration/velocity

= 15/6

= 2.5

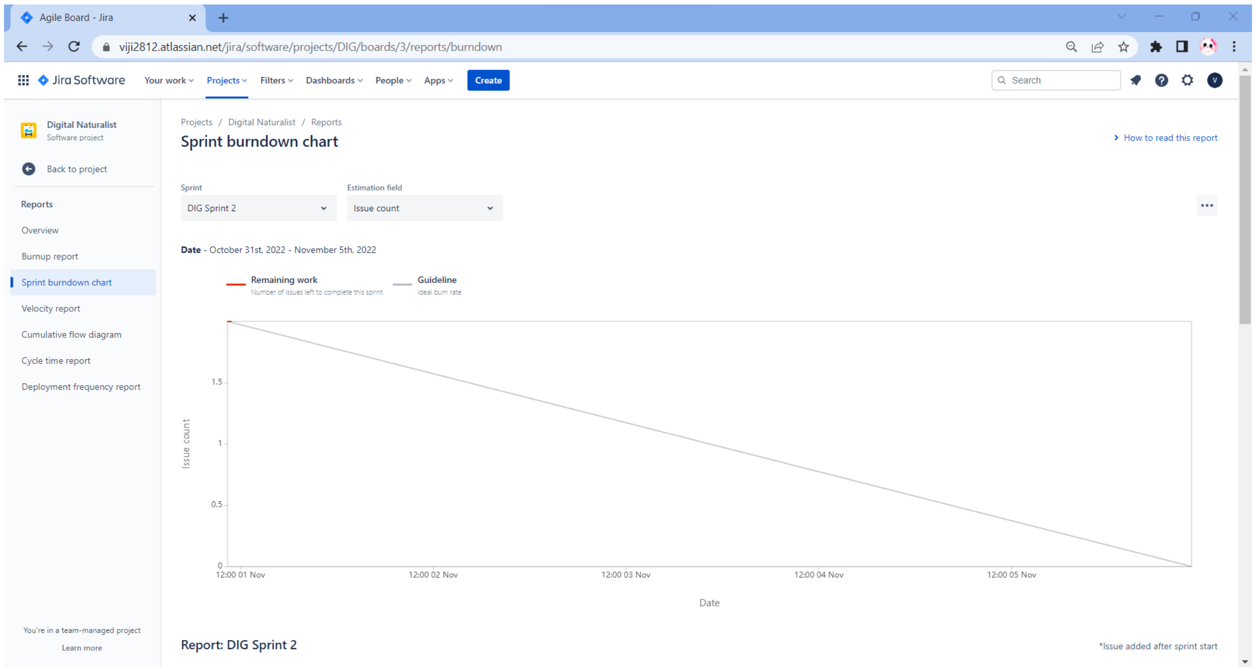
6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Acutal Release Date (Planned)
Sprint-1	15	6 Days	24 Oct 2022	29 Oct 2022	15	29 Oct 2022
Sprint-2	15	6 Days	31 Oct 2022	05 Nov 2022	15	05 Nov 2022
Sprint-3	09	6 Days	07 Nov 2022	12 Nov 2022	09	12 Nov 2022
Sprint-4	10	6 Days	14 Nov 2022	19 Nov 2022	10	19 Nov 2022

6.3 Reports from JIRA

Burndown Chart:

A Burndown Chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



Sprint burndown chart

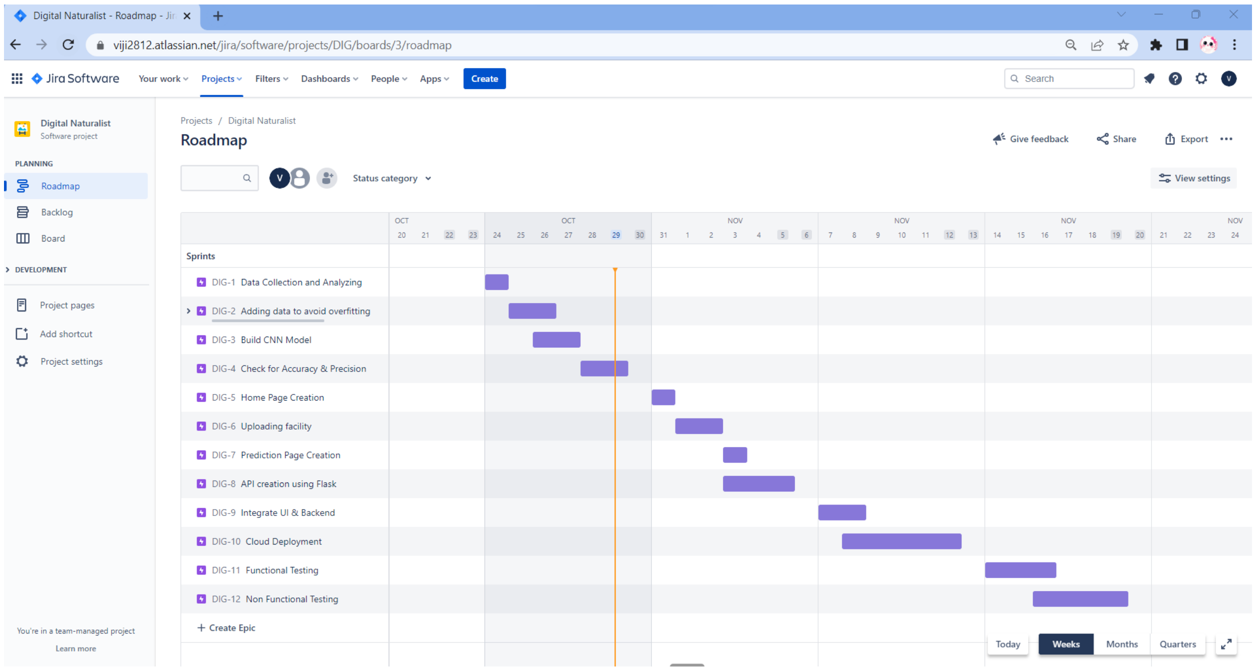
Sprint: DIG Sprint 2

Estimation field: Issue count

Date - October 31st, 2022 - November 5th, 2022

Report: DIG Sprint 2

Roadmap:



Roadmap

Sprints

- DIG-1 Data Collection and Analyzing
- DIG-2 Adding data to avoid overfitting
- DIG-3 Build CNN Model
- DIG-4 Check for Accuracy & Precision
- DIG-5 Home Page Creation
- DIG-6 Uploading facility
- DIG-7 Prediction Page Creation
- DIG-8 API creation using Flask
- DIG-9 Integrate UI & Backend
- DIG-10 Cloud Deployment
- DIG-11 Functional Testing
- DIG-12 Non Functional Testing

+ Create Epic

7. CODING & SOLUTIONING

7.1 Code for web page creation:

home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <link rel="spreadsheet" type="text/css" href="styles.css">
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<link href='https://fonts.googleapis.com/css?family=Josefin Sans' rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Merriweather' rel='stylesheet'>
        <script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>

  <title>Digital Naturalist</title>
  <style>
    @import
url('https://fonts.googleapis.com/css2?family=Josefin+Sans:ital,wght@1,300&family=Montserr
at&family=Raleway:wght@100&family=Roboto:wght@300&family=Sacramento&family=Source+
Sans+3:wght@300;400&display=swap');
    *{
      padding:0;
      margin: 0;
      box-sizing: border-box;
      list-style: none;
      text-decoration: none;

    }
    body{
      font-family: 'Montserrat', sans-serif;
      background-image: linear-gradient( 89.8deg, rgb(13, 95, 33) 4.7%, rgba(30, 29, 29, 1) 120.3%);
    }
    .sec{
      background-image: url('../static/images/tree.jpg');
      background-size: cover;


      height:100vh;

    }
    .sec::before{
    content:' ';
    display: block;
    position: absolute;
    background-color: #000;
    opacity: 0.4;
    width: 100%;
    height: 100vh;
}
    .topic{
      font-size: 80px;
      color:white;
      filter: brightness(100%);
      text-align: center;
      font-weight: 500;
      padding:160px 60px 0px 60px;
      font-family: 'Raleway', sans-serif;
    }
    .desc{
      font-size: 40px;
      color:white;
      filter: brightness(100%);
      text-align: center;
      font-weight: 500;
      padding:12px 60px 0px 60px;
      font-family: 'Raleway', sans-serif;
    }
    nav{
      background-color: black;
      height:45px;
      width: 100%;
      padding: 0;
```

```
  display:inline-block;
  margin-bottom: 0!important;
}
.logo{
  height: 45px; width:45px; line-height: 45px;

}
.lab{
  color:  linear-gradient( 89.8deg, rgb(13, 95, 33) 4.7%, rgba(30, 29, 29, 1) 120.3%);;
  font-size: 20px;
  line-height: 45px;
  margin-left: 0;
  margin-top: 0px;
  margin-bottom: 20px;
  padding:5 5 25 25;
  text-transform: uppercase;
  position: absolute;

  background: #15BD0F;
  background: linear-gradient(to right, #15BD0F 0%, #063804 100%);
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;

}

nav ul{
  float:right;
  color:white;
  margin-right: 15px;
  padding-left: 50px;
  line-height: 45px;

}
nav ul li{
  display:inline-block;
  padding-left: 25px;
}
nav ul li a{
  color:white;
  text-transform: uppercase;
  font-size: 15px;
  text-decoration: none;
}
nav ul li a:hover{
  color:rgb(37, 145, 64);
  transition:.5s;
  text-decoration: none;
}
button{
  position: relative;
  border:none;
  height: 50px;
  width:150px;
  border-radius: 4px;
  transition: .4s ease-in;
  font-family: 'Raleway';
  font-weight: 600;
  font-size: 15px;
  margin: auto;
  display: block;
  background-color: #000;
  color: white;
  z-index:1;
}
button:hover{
  border:2px solid #063804;
  color:rgb(163, 65, 0) ;
  background-color: black;
  text-decoration: none;
}

.center
{ position: block;
  width: 100%;
}

.center a{
  text-decoration: none;
}
#sidebar{
	float:right;
	width:50%;
```

```
            background-color: transparent;
            color:#000;
            font-family:Georgia, serif;
            padding-left:0px;
            padding-right:0px;
            padding-top:1px;
            box-sizing: border-box;
}
.img-preview {
    width: 300px;
    height: 300px;
    position: relative;
    border: 5px solid #F8F8F8;
    box-shadow: 0px 2px 4px 0px rgba(0, 0, 0, 0.1);
    margin-top: 1em;
    margin-bottom: 1em;

            text-align: center;
}

.img-preview>div {
            padding-left: 45%;
    width: 100%;
    height: 100%;
    background-size: 300px 300px;
    background-repeat: no-repeat;
    background-position: center;
            text-align: center;
}

input[type="file"] {
    display: none;
}

.upload-label{
    display: inline-block;
    padding: 12px 30px;
    background: #161616;
    color: #fff;
    font-size: 1em;
    transition: all .4s;
    cursor: pointer;
            font-weight:bold;
}

.upload-label:hover{
    background: #3A3A3A;
    color: white;
            font-weight:bold;
}

.loader {
    border: 8px solid #f3f3f3; /* Light grey */
    border-top: 8px solid #161616;
    border-radius: 50%;
    width: 50px;
    height: 50px;
    animation: spin 1s linear infinite;
}

@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}


  </style>


</head>
<body>
  <nav>
    <img class="logo" src="../static/images/greentree-removebg-preview.png" >
    <label class="lab">Digital Naturalist</label>
    <ul>
      <li><a href="{{url_for('home_func')}}">Home</a></li>
      <li><a href="{{url_for('aboutus_func')}}">About us</a></li>
    </ul>
  </nav>
  <div class="sec">
    <div class="content">
      <p class="topic">ARE YOU A NATURALIST ?</p><br>
      <p class="desc">
        Wanna know more about the flora and fauna around you?<br>

        Drop their picture now! </p>
    </div>
    <br>

        <div class="center">
          <a href="#section2">
            <button type="button" class="myButton"  >Click here!</button>

          </a>
    </div>
  </div>



  <div class="predicting" id="section2" >
    <section id="main">
    <div style="text-align:center;width:100%;">
    <p><h3 style="color:white;" >Click on choose and upload the image<br><br></h3></p>

      </div>
      </section>
</div>
  <div style="margin-top:0%;padding-top:0%;">
      <div>
        <h4 style="font-size:19px;text-align:center; color:white; padding-bottom:30px ; ">Upload
your image</h4>
    <center><form style="border-radius: 4px;"action = "http://localhost:5000/" id="upload-file"
method="post" enctype="multipart/form-data">
      <label style="text-align: center;"for="imageUpload" class="upload-label">
        Choose
      </label>
      <input type="file" name="image" id="imageUpload" accept=".png, .jpg, .jpeg">
    </form></center>



    <div class="image-section" style="display: none; text-align:center;padding-left: 40%;
width:500px; height:300px;">
      <div class="img-preview">
        <div id="imagePreview">
        </div>
      </div>
      </div>
      <br>
    <div class="image-section" style="display: none;">
      <div style="padding-left: 45%;">
        <button type="button" class="btn btn-lg upload-label" id="btn-predict">Predict!</button>
      </div>
    </div>

    <div class="loader" style="display:none;"></div>
    <div style="width:70%;text-align:justify;margin-left:20%;">
    <h4 style="color:white">
      <span id="result"> </span>
    </h4></div>

</div>

</div></div>
<script>
window.onscroll = function() {myFunction()};

$(document).ready(function () {
  // Init
  $('.image-section').hide();
  $('.loader').hide();
  $('#result').hide();

  // Upload Preview
  function readURL(input) {
      if (input.files && input.files[0]) {
          var reader = new FileReader();
          reader.onload = function (e) {
              $('#imagePreview').css('background-image', 'url(' + e.target.result + ')');
              $('#imagePreview').hide();
              $('#imagePreview').fadeIn(650);
          }
          reader.readAsDataURL(input.files[0]);
      }
  }
  $("#imageUpload").change(function () {
      $('.image-section').show();
      $('#btn-predict').show();
      $('#result').text('');
      $('#result').hide();
      readURL(this);
  });

  // Predict
  $('#btn-predict').click(function () {
      var form_data = new FormData($('#upload-file')[0]);

      // Show loading animation
      $(this).hide();
      $('.loader').show();

      // Make prediction by calling api /predict
      $.ajax({
          type: 'POST',
          url: '/predict',
          data: form_data,
          contentType: false,
          cache: false,
          processData: false,
          async: true,
          success: function (data) {
              // Get and display the result
              $('.loader').hide();
              $('#result').fadeIn(600);
              $('#result').text('Prediction: '+data);
              console.log('Success!');
          },
      });
  });

});
</script>




</body>
</html>
```

aboutus.html

```
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Digital Naturalist</title>

   <style>

@import
url('https://fonts.googleapis.com/css2?family=Josefin+Sans:ital,wght@1,300&family=Montserr
at&family=Raleway:wght@100&family=Roboto:wght@300&family=Sacramento&family=Source+
Sans+3:wght@300;400&display=swap');
      body{
     font-family: 'Montserrat', sans-serif;
     background-image: linear-gradient( 89.8deg, rgb(13, 95, 33) 4.7%, rgba(30, 29, 29, 1) 120.3%);
            margin: 0;
            text-decoration: none;
   }




          nav{
     background-color: black;
     height:45px;
     width: 100%;
     padding: 0;
     display:inline-block;
     margin-bottom: 0!important;
   }
   .logo{
     height: 45px; width:45px; line-height: 45px;

   }
   .lab{
     color:  linear-gradient( 89.8deg, rgb(13, 95, 33) 4.7%, rgba(30, 29, 29, 1) 120.3%);;
```

```
  font-size: 20px;
  line-height: 45px;
  margin-left: 0;
  margin-top: 0px;
  margin-bottom: 20px;
  padding: 4px 5px 25px 2px;
  text-transform: uppercase;
  position: absolute;

  background: #15BD0F;
  background: linear-gradient(to right, #15BD0F 0%, #085a06 100%);
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;

}

nav ul{
  float:right;
  color:white;
  margin-right: 15px;
  padding-left: 50px;
  margin-top: 0px;
  line-height: 45px;

}
nav ul li{
  display:inline-block;
  padding-left: 25px;
  padding-top: 0px;
  line-height: 45px;
}
nav ul li a{
  color:white;
  text-transform: uppercase;
  font-size: 15px;
  text-decoration: none;
}
nav ul li a:hover{
  color:rgb(37, 145, 64);
  transition:.5s;
  text-decoration: none;
```

```
    }
    .container{
      width:75vw;
      height: 65vh;
      padding:5% 13% 8% 13%;
      align-content: center;
    }

    .content{
      width:100%;
      height:100%;
      background-color: black;
      box-shadow: 10px 10px 28px 20px rgb(8, 31, 11);
      border-radius: 18px;

    }

    .team{
      color: white;

    }
    .team li{
      list-style: none;
      font-size: large;
      padding: 8px 0px 0px 16px;
    }


    </style>




</head>
<body>
    <nav>
        <img class="logo" src="../static/images/greentree-removebg-preview.png" >
        <label class="lab">Digital Naturalist</label>
        <ul>
          <li><a href="{{url_for('home_func')}}">Home</a></li>
          <li><a href="{{url_for('aboutus_func')}}">About us</a></li>
    </ul>
   </nav>
   <div class="container">
    <div class="content">
      <h2 style="padding: 25px 0px 0px 25px;color: #085a06; "> Digital Naturalist – AI Enabled
tool for
       Biodiversity Researchers </h2>
      <p style="padding: 8px 35px 0px 25px; color:white; font-size: large; text-align: justify;">
       <span style="display:inline-block; ">    </span>
       <span style="display:inline-block; ">    </span>
       <span style="display:inline-block; ">    </span>
       This project is a tool that predicts the uploaded flora and fauna image of the user and
displays a description about it.
        It benefits the inquistive naturalists and influencers to know more about the
environment.</p>
      <h2 style="padding: 15px 0px 0px 25px;color: #085a06; ">Team Members</h2>
      <ul class="team">
       <li >K Vijayalakshmi - 211419104303</li>
       <li >B Rakshana - 211419104212</li>
       <li >K Swetha -   211419104284</li>
       <li >S Swetha -  211419104286</li>


      </ul>
    </div>


   </div>

</body>
</html>
```

8. TESTING

8.1 TEST CASES

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
Webpage_TC_001	UI	Home Page	Verify whether user can view the homepage	1. Latest web browser 2. Proper Internet Connection	1. Enter the url of the website and click go 2. Verify the webpage is loading or not	no test data required	The webpage should be visible to the user	The webpage is visible	Pass	The test case passed without any issues	Y	1	Vijayalakshmi K
Webpage_TC_002	UI	Home Page	Verify whether user is able to upload the image	1. Latest web browser 2. Proper Internet Connection 3.Sample images for testing	1. Enter the url of the website and click go 2. After tha page loaded , Click on the choose button to uploadthe image	Image	The image should be uploaded successfully	The image was uploaded successfully	Pass	The test case passed without any issues	Y	2	Swetha S
Webpage_TC_003	UI	Home Page	Verify the webpage accepts proper inputs from user and displays it	1. Latest web browser 2. Proper Internet Connection 3.Sample images for testing	1. Enter the url of the website and click go 2. After tha page loaded , Click on the choose button to upload the image	Sample images for testing	The webpage should accept the image and display it to the user	The webpage accepts the image and displays it to the user.	Pass	The test case passed without any issues	Y	3	Rakshana B
Webpage_TC_004	UI	Home Page	Verify whether web components work properly	1. Latest web browser 2. Proper Internet Connection	1. Enter the url of the website and click go 2. Verify the webpage is loading and working properly during the upload and reset	Sample image for testing	The webpage should be stable during the upload and prediction of the image	The webpage is responding stabley	Pass	The test case passed without any issues	Y	4	Rakshana B
Webpage_TC_005	UI	About us Page	Verify whether user can view the about us page	1. Latest web browser 2. Proper Internet Connection	1. Enter the url of the website and click go 2. Verify the webpage is loading or not	no test data required	The webpage should be visible to the user	The webpage is visible	Pass	The testcase passed without any issues	Y	5	Swetha K
Flask_TC_001	Functional	Flask app	Verify the flask app whether it uses the trained model	1. Latest web browser 2. Proper Internet Connection	1. Enter the url of the website and click go 2. Verify the webpage is accepting inputs and predecting according to the category of the animal	Sample images for testing	The webapp should predict the image properly	The webapp predicts the image accurately	Pass	The test case passed without any issues, but it requires more training set to predict the image accuretly	Y	6	Vijayalakshmi K
Flask_TC_002	Functional	Flask app	Verify whether the uploaded image is saved in the specified folder	1. Latest web browser 2. Proper Internet Connection 3. Storage as a folder for storing the uploaded image	1. Enter the url of the website and click go 2. After page loading try to upload the image and wait	Sample images for testing	The website should accept the image data and save it locally in a folder	The app stored the image successfully	Pass	The testcase passed without any issues, But it will be an issue in future when the storage	Y	7	Swetha S

										overflows			

Flask _TC_003	Functional	Flask app	Verify whether the app displays the prediction or not?	1. Latest web browser 2. Proper Internet Connection 3.Sample images for testing	1. Enter the url of the website and click go 2. Verify the webpage is accepting inputs and predecting according to their category.	Sample images for testing	The web app should be able to display the prediction message	The app displays the prediction message successfully	Pass	The testcase passed without any issues	Y	8	Rakshana R
Flask _TC_004	Functional	Flask app	Verify whether the app redirects the user to the about us page or not?	1. Latest web browser 2. Proper Internet Connection 3.Sample images for testing	1. Enter the url of the website and click go. 2. Verify the page is redirecting to about us page.	no test data required	The web app should redirect to the user to about us page	The app redirected successfully	Pass	The testcase passed without any issues	Y	9	Swetha S

8.2 USER ACCEPTANCE TESTING

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Digital Naturalist project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	1	7	2	1	11
Duplicate	1	0	3	0	4
External	3	2	0	0	5
Fixed	4	2	4	2	12
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	0	0	0	0
Totals	14	11	11	4	35

3. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
User Interface	4	0	0	4
Flask Application	5	0	0	5
Exception Reporting	3	0	0	3
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9. RESULTS

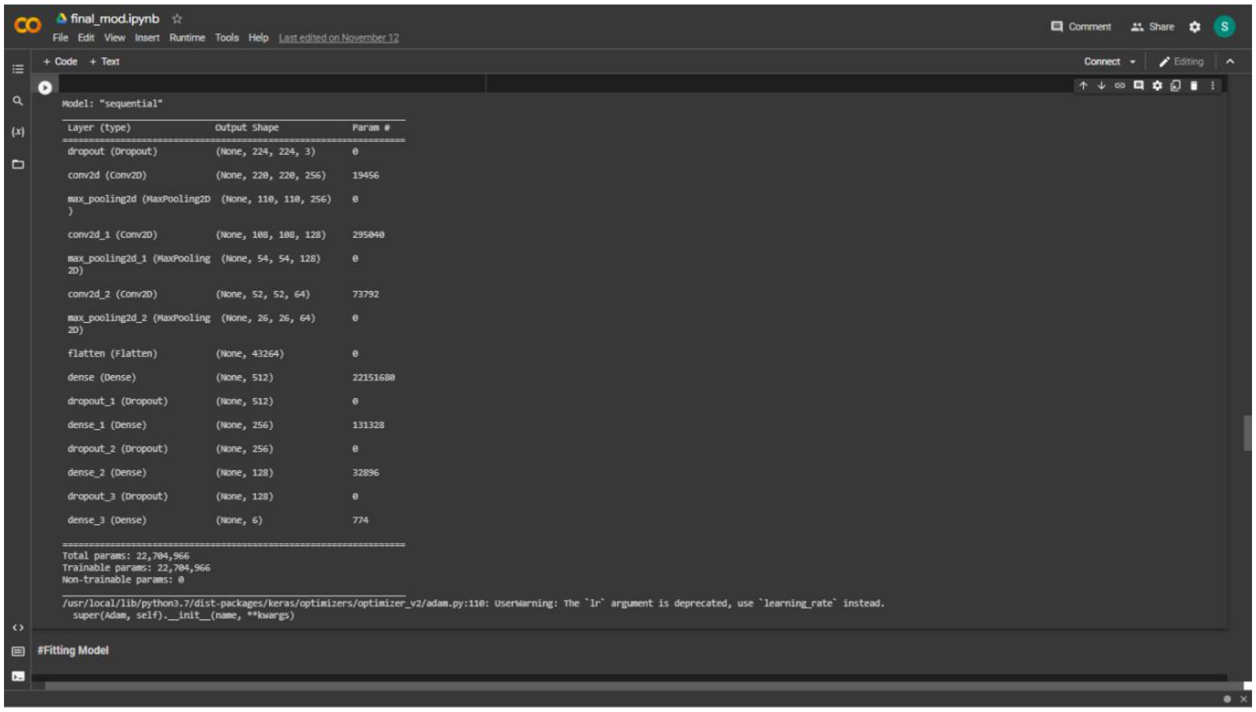
9.1 PERFORMANCE METRICS

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S. No	Parameter	Values	Screenshot
1.	Model Summary	**Total params: 22,704,966** **Trainable params: 22,704,966** **Non-trainable params: 0**	Screenshot 1
2.	Accuracy	Training Accuracy - 92.73% Validation Accuracy – 80.73%	Screenshot 2

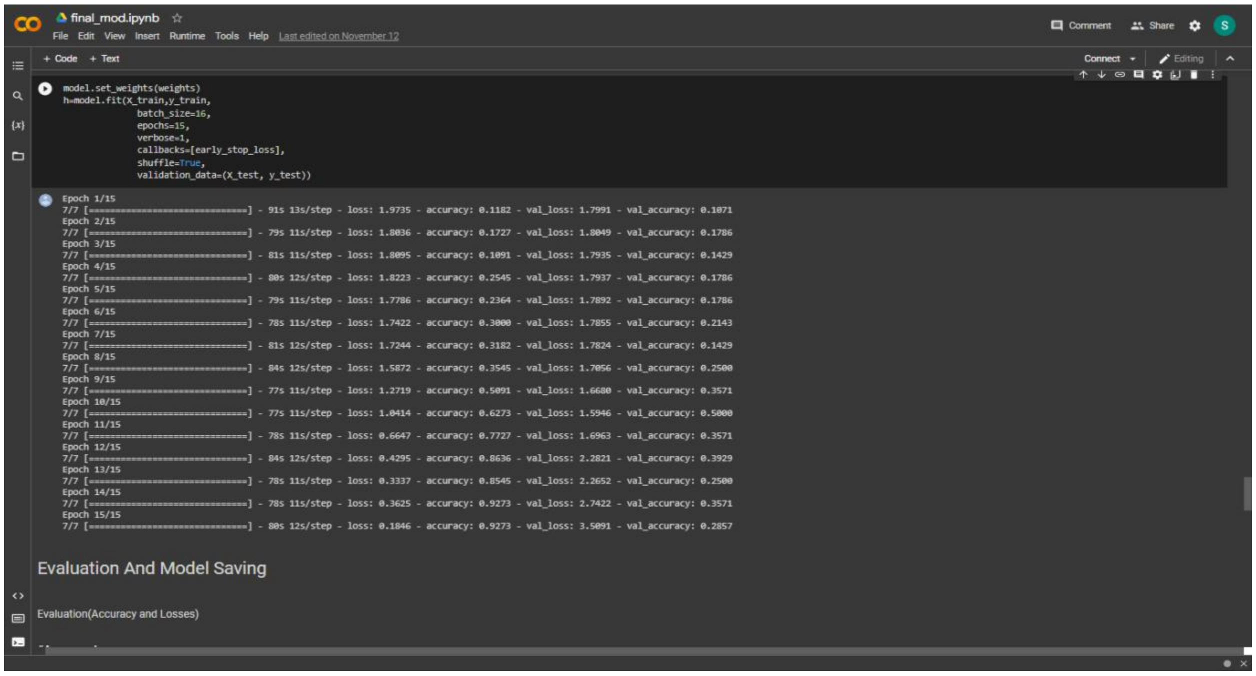
SCREENSHOT 1:


final_mod.ipynb
File Edit View Insert Runtime Tools Help Last edited on November 12
Comment Share
+ Code + Text
Connect Editing

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dropout (Dropout)           (None, 224, 224, 3)       0

 conv2d (Conv2D)             (None, 220, 220, 256)     19456

 max_pooling2d (MaxPooling2D  (None, 110, 110, 256)    0
 )

 conv2d_1 (Conv2D)           (None, 108, 108, 128)     295040

 max_pooling2d_1 (MaxPooling  (None, 54, 54, 128)      0
 2D)

 conv2d_2 (Conv2D)           (None, 52, 52, 64)        73792

 max_pooling2d_2 (MaxPooling  (None, 26, 26, 64)       0
 2D)

 flatten (Flatten)           (None, 43264)             0

 dense (Dense)               (None, 512)               22151680

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 256)               131328

 dropout_2 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 128)               32896

 dropout_3 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 6)                 774

=================================================================
Total params: 22,704,966
Trainable params: 22,704,966
Non-trainable params: 0
_________________________________________________________________
/usr/local/lib/python3.7/dist-packages/keras/optimizers/optimizer_v2/adam.py:110: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
```

#Fitting Model

SCREENSHOT 2:


final_mod.ipynb
File Edit View Insert Runtime Tools Help Last edited on November 12
Comment Share
+ Code + Text
Connect Editing

```
model.set_weights(weights)
h=model.fit(X_train,y_train,
              batch_size=16,
              epochs=15,
              verbose=1,
              callbacks=[early_stop_loss],
              shuffle=True,
              validation_data=(X_test, y_test))
```

```
Epoch 1/15
7/7 [==============================] - 91s 13s/step - loss: 1.9735 - accuracy: 0.1182 - val_loss: 1.7991 - val_accuracy: 0.1071
Epoch 2/15
7/7 [==============================] - 79s 11s/step - loss: 1.8036 - accuracy: 0.1727 - val_loss: 1.8049 - val_accuracy: 0.1786
Epoch 3/15
7/7 [==============================] - 81s 11s/step - loss: 1.8095 - accuracy: 0.1091 - val_loss: 1.7935 - val_accuracy: 0.1429
Epoch 4/15
7/7 [==============================] - 80s 12s/step - loss: 1.8223 - accuracy: 0.2545 - val_loss: 1.7937 - val_accuracy: 0.1786
Epoch 5/15
7/7 [==============================] - 79s 11s/step - loss: 1.7786 - accuracy: 0.2364 - val_loss: 1.7892 - val_accuracy: 0.1786
Epoch 6/15
7/7 [==============================] - 78s 11s/step - loss: 1.7422 - accuracy: 0.3000 - val_loss: 1.7855 - val_accuracy: 0.2143
Epoch 7/15
7/7 [==============================] - 81s 12s/step - loss: 1.7244 - accuracy: 0.3182 - val_loss: 1.7824 - val_accuracy: 0.1429
Epoch 8/15
7/7 [==============================] - 84s 12s/step - loss: 1.5872 - accuracy: 0.3545 - val_loss: 1.7056 - val_accuracy: 0.2500
Epoch 9/15
7/7 [==============================] - 77s 11s/step - loss: 1.2719 - accuracy: 0.5091 - val_loss: 1.6680 - val_accuracy: 0.3571
Epoch 10/15
7/7 [==============================] - 77s 11s/step - loss: 1.0414 - accuracy: 0.6273 - val_loss: 1.5946 - val_accuracy: 0.5000
Epoch 11/15
7/7 [==============================] - 78s 11s/step - loss: 0.6647 - accuracy: 0.7727 - val_loss: 1.6963 - val_accuracy: 0.3571
Epoch 12/15
7/7 [==============================] - 84s 12s/step - loss: 0.4295 - accuracy: 0.8636 - val_loss: 2.2821 - val_accuracy: 0.3929
Epoch 13/15
7/7 [==============================] - 78s 11s/step - loss: 0.3337 - accuracy: 0.8545 - val_loss: 2.2652 - val_accuracy: 0.2500
Epoch 14/15
7/7 [==============================] - 78s 11s/step - loss: 0.3625 - accuracy: 0.9273 - val_loss: 2.7422 - val_accuracy: 0.3571
Epoch 15/15
7/7 [==============================] - 80s 12s/step - loss: 0.1846 - accuracy: 0.9273 - val_loss: 3.5091 - val_accuracy: 0.2857
```

Evaluation And Model Saving

Evaluation(Accuracy and Losses)

10. ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

An advantage of naturalistic observation is that it allows the investigators to directly observe the subject in a natural setting. The method gives scientists a first-hand look at social behavior and can help them notice things that they might never have encountered in a lab setting.

The observations can also serve as inspiration for further investigations. The information gleaned from naturalistic observation can lead to insights that can be used to help people overcome problems and lead to healthier, happier lives.

- **Allows researchers to study behaviors or situations that cannot be manipulated in a lab due to ethical concerns**. For example, it would be unethical to study the effects of imprisonment by actually confining subjects. But researchers can gather information by using naturalistic observation in actual prison settings.
- **Can support the external validity of research**. Researchers might believe that the findings of a lab study can be generalized to a larger population, but that does not mean they would actually *observe* those findings in a natural setting.

DISADVANTAGES

Naturalistic observation can be useful in many cases, but the method also has some downsides. Some of these include:

- **Inability to draw cause-and-effect conclusions**: The biggest disadvantage of naturalistic observation is that determining the exact cause of a subject's behavior can be difficult.
- **Lack of control**: Another downside is that the experimenter cannot control for outside variables.
- **Lack of validity**: While the goal of naturalistic observation is to get a better idea of how it occurs in the real world, experimental effects can still influence how people respond. The Hawthorne effect and other demand characteristics can play a role in people altering their behavior simply because they know they are being observed.

It is also important to note that naturalistic observation is a type of correlational research (others include surveys and archival research). A correlational study is a non-experimental approach that seeks to find statistical relationships between variables. Naturalistic observation is one method that can be used to collect data for correlational studies.

While such methods can look at the direction or strength of a relationship between two variables, they cannot determine if one causes the other. As the saying goes, correlation does not imply causation.

11. CONCLUSION

Naturalistic observation can play an important role in the research process. It offers a number of advantages, including often being more affordable and less intrusive than other types of research.

In some cases, researchers may utilize naturalistic observation as a way to learn more about something that is happening in a certain population. Using this information, they can then formulate a hypothesis that can be tested further.

In conclusion, AI applications in Biodiversity help conserve various species of mammals, insects, birds, etc. Moreover, it helps experts understand the behaviors of multiple species in their ecology. Hence, it helps replicate certain necessary elements for their survival from their environments.

12. FUTURE SCOPE

One very time-consuming task in biodiversity research is data collection. Traditionally, a scientist might have spent hours waiting for one single observation, chasing away most timid animals and therefore distorting the data. Machine observations free researchers from tedious tasks and even make rare observations possible at all. Researchers have been using camera traps in order to monitor bigger animals like lions or antelopes. But after collecting huge amounts of images, the problem remains that the amount of information exceeds the capacity of human interpreters, with only a small percentage of the collected material being relevant at all. That’s why automating the collection of data will only reach its full potential if data analysis can be automated.

13. APPENDIX

SOURCE CODE

Code for building flask app:

app.py

```
from __future__ import division, print_function

import os

import numpy as np

from keras.models import load_model

import tensorflow as tf

from tensorflow.keras.preprocessing import image

from flask import Flask, request, render_template

from werkzeug.utils import secure_filename

from keras.models import model_from_json

global graph

graph=tf.compat.v1.get_default_graph()

# Define a flask app

app = Flask(__name__)




# Load your trained model

json_file = open('final_model.json', 'r')

loaded_model_json = json_file.read()
```

```
json_file.close()

loaded_model = model_from_json(loaded_model_json)

loaded_model.load_weights("final_model.h5")

print('Model loaded. Check http://127.0.0.1:5000/')

#Configure Home page.

@app.route('/')

def index():

  # Main page

  return render_template('home.html')

@app.route('/home')

def home():

  # Main page

  return render_template('home.html')

@app.route("/home", methods=['GET','POST'])

def home_func():

  # Main page

  if request.method=='POST':

      return redirect(url_for('home'))

  return render_template('home.html')


@app.route('/aboutus')

def aboutus():

  # Main page

  return render_template('aboutus.html')
```

```
@app.route("/aboutus", methods=['GET','POST'])
def aboutus_func():
    # Main page
    if request.method=='POST':
        return redirect(url_for('aboutus'))
    return render_template('aboutus.html')
#Pre-process the frame and run
@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['image']

        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        img = image.load_img(file_path, target_size=(224, 224))
x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)
with graph.as_default():
model = load_model('./model/final_model.h5')
```

```
        preds = np.argmax(model.predict(x),axis=1)
    found = ["   The great Indian bustard is a bustard found on the Indian subcontinent. A large bird with a horizontal body and long bare legs, giving it an ostrich like appearance, this bird is among the heaviest of the flying birds. It belongs to Otididae family and is listed among critically endangered species.",
            "   The spoon-billed sandpiper is a small wader which breeds in northeastern Russia and winters in Southeast Asia. It belongs to Scolopacidae family and is listed among critically endangered species.",
            "   Amorphophallus Titanum is endemic to sumantra. Due to its odor, like that of a rotting corpse, the titan arum is characterized as a Carrion Flower or Corpse Flower. It belongs to Araceae family.",
            "   Lady's slipper, (subfamily Cypripedioideae), also called lady slipper or slipper orchid, subfamily of five genera of orchids (family Orchidaceae), in which the lip of the flower is slipper-shaped.",
            "   Pangolins, sometimes known as scaly anteaters, are of the order Pholidota. Often thought of as a reptile, but pangolins are actually mammals. They are the most trafficked mammals.",
            "   The white deer found at Seneca Army Depot are a natural variation of the white-tailed deer (Odocoileus virginianus), which usually have brown coloring. The Seneca White Deer are leucistic, meaning they lack all pigmentation in the hair, but have the normal brown-colored eyes."]
    text = found[preds[0]]
    return text
if __name__ == '__main__':
app.run(threaded = False)
```

Code for augmenting the data:

Aug data.ipynb

```
#import libraries

from keras.preprocessing.image import ImageDataGenerator

import cv2

from os import listdir
import time

def hms_string(sec_elapsed):
    h = int(sec_elapsed / (60 * 60))
    m = int((sec_elapsed % (60 * 60)) / 60)
    s = sec_elapsed % 60
    return f"{h}:{m}:{round(s,1)}"

def augment_data(file_dir, n_generated_samples, save_to_dir):

    data_gen = ImageDataGenerator(rotation_range=30,
                                  width_shift_range=0.1,
                                  height_shift_range=0.15,
                                  shear_range=0.25,
                                  zoom_range = 0.2,
                                  horizontal_flip=True,
                                  vertical_flip=False,
                                  fill_mode='nearest',
                                  brightness_range=(0.5,1.2)
                                 )
    for filename in listdir(file_dir):
        # load the image
        image = cv2.imread(file_dir + '/' + filename)

        # reshape the image
        image = image.reshape((1,)+image.shape)
        # prefix of the names for the generated sampels.
        save_prefix = 'aug_' + filename[:-4]
        # generate 'n_generated_samples' sample images
        i=0
        for batch in data_gen.flow(x=image, batch_size=1, save_to_dir=save_to_dir,
save_prefix=save_prefix, save_format='jpg'):
            i += 1
            if i > n_generated_samples:
                break
    """
    Arguments: file_dir: A string representing the directory where images that we want to augment
are found.
          n_generated_samples: A string representing the number of generated samples using the
given image.
          save_to_dir: A string representing the directory in which the generated images will be
saved."""
```

Code for training the data:

<u>model_train.ipynb</u>

```
from google.colab import drive
drive.mount("/content/drive")

!unzip "/content/drive/MyDrive/Colab Notebooks/Digital Naturalist Dataset.zip"

#import libraries
from keras.preprocessing.image import ImageDataGenerator
import cv2
from os import listdir
import time

def hms_string(sec_elapsed):
    h = int(sec_elapsed / (60 * 60))
    m = int((sec_elapsed % (60 * 60)) / 60)
    s = sec_elapsed % 60
    return f"{h}:{m}:{round(s,1)}"

def augment_data(file_dir, n_generated_samples, save_to_dir):
    data_gen = ImageDataGenerator(rotation_range=30,
                                  width_shift_range=0.1,
                                  height_shift_range=0.15,
                                  shear_range=0.25,
                                  zoom_range = 0.2,
                                  horizontal_flip=True,
                                  vertical_flip=False,
                                  fill_mode='nearest',
                                  brightness_range=(0.5,1.2)
                                 )
    for filename in listdir(file_dir):
        # load the image
        image = cv2.imread(file_dir + '/' + filename)

        # reshape the image
        image = image.reshape((1,)+image.shape)

        # prefix of the names for the generated sampels.
        save_prefix = 'aug_' + filename[:-4]
        # generate 'n_generated_samples' sample images
        i=0
        for batch in data_gen.flow(x=image, batch_size=1, save_to_dir=save_to_dir,
save_prefix=save_prefix, save_format='jpg'):
            i += 1
            if i > n_generated_samples:
                break
    """
    Arguments: file_dir: A string representing the directory where images that we want to augment
are found.
          n_generated_samples: A string representing the number of generated samples using the
given image.
          save_to_dir: A string representing the directory in which the generated images will be
saved."""

file_dir=r"/content/drive/MyDrive/Colab Notebooks/Digital Naturalist/Digital Naturalist Dataset"

start_time = time.time()

#3. Augmentation Structure Creation
augmented_data_path = r"/content/drive/MyDrive/Colab Notebooks/Digital
```

```
Naturalist/augmented data"

#For Birds
# augment data for the examples with label equal to GIB in Birds
augment_data(file_dir=r'/content/drive/MyDrive/Colab Notebooks/Digital Naturalist/Digital
Naturalist Dataset/Bird/Great Indian Bustard Bird', n_generated_samples=8,
save_to_dir=augmented_data_path+'/Bird/GIB_AUG')
# augment data for the examples with label equal to GIB in Birds
augment_data(file_dir=r'/content/drive/MyDrive/Colab Notebooks/Digital Naturalist/Digital
Naturalist Dataset/Bird/Spoon Billed Sandpiper Bird', n_generated_samples=8,
save_to_dir=augmented_data_path+'/Bird/SPS_AUG')

#For MAMMALS
# augment data for the examples with label equal to GIB in Flower
augment_data(file_dir=r'/content/drive/MyDrive/Colab Notebooks/Digital Naturalist/Digital
Naturalist Dataset/Flower/Corpse Flower', n_generated_samples=8,
save_to_dir=augmented_data_path+'/Flower/Corpse_AUG')
# augment data for the examples with label equal to GIB in Flower
augment_data(file_dir=r'/content/drive/MyDrive/Colab Notebooks/Digital Naturalist/Digital
Naturalist Dataset/Flower/Lady Slipper Orchid Flower', n_generated_samples=8,
save_to_dir=augmented_data_path+'/Flower/LS_Orchid_AUG')

#For Flowers
# augment data for the examples with label equal to GIB in Mammals
augment_data(file_dir=r'/content/drive/MyDrive/Colab Notebooks/Digital Naturalist/Digital
Naturalist Dataset/Mammal/Pangolin Mammal', n_generated_samples=8,
save_to_dir=augmented_data_path+'/Mammal/LS_Pangolin_AUG')
# augment data for the examples with label equal to GIB in Mammals
augment_data(file_dir=r'/content/drive/MyDrive/Colab Notebooks/Digital Naturalist/Digital
Naturalist Dataset/Mammal/Senenca White Deer Mammal', n_generated_samples=8,
save_to_dir=augmented_data_path+'/Mammal/SW_Deer_AUG')


end_time = time.time()
execution_time = (end_time - start_time)
print(f"Elapsed time: {hms_string(execution_time)}")
```

# Loading Data and Preprocessing

```
#Importing the libraries
```

```
#For matrix calculations and data Managememnt
import pandas as pd
import numpy as np

#Importing libraries required for the model
import tensorflow as tf
import keras
import keras.backend as K

from keras.optimizers import SGD, Adam, Adagrad, RMSprop
from keras.applications import *
from keras.preprocessing import *
from keras_preprocessing.image import ImageDataGenerator, img_to_array, array_to_img,
load_img
from keras.callbacks import EarlyStopping, ModelCheckpoint
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Activation,
BatchNormalization,Dropout
from keras.models import Model
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split

#For plotting charts used for data visualizations
import matplotlib.pyplot as plt

#Libraries for Locating and loading data
import pathlib
from pathlib import Path
import os, gc, glob, random
from PIL import Image

#Make a list of paths to all folders where you have data
#Setting path to our dataset folder
#dirName = r'C:/Users/vijay/OneDrive/Desktop/Digital Naturalist'
dirName="/content/drive/MyDrive/Colab Notebooks/Digital Naturalist/Digital Naturalist
Dataset"

folders = listdir(dirName)

#Getting the names for all the folders containing data
def getListOfFiles(dirName):
```

```
# create a list of sub directories and files(if any)
# names in the given directory
    listOfFile = os.listdir(dirName)
    allFiles = list()
    for fol_name in listOfFile:
        fullPath = os.path.join(dirName, fol_name)
        allFiles.append(fullPath)

    return allFiles

Folders = getListOfFiles(dirName)
len(Folders)
subfolders = []
for num in range(len(Folders)):
    sub_fols = getListOfFiles(Folders[num])
    subfolders+=sub_fols
#Now, the subfolders contains the address to all our data folders for each class
subfolders
```

#Loading Images into machine understandable Data

```
#X data will includes the data generated for each image
#Y data will include a id no:, for every different boat type in out boats folder
#a different number is being assigned. That will be tha label we're classifying.
X_data = []
Y_data = []


id_no=0
found = []
#itering in all folders under Boats folder
for paths in subfolders:
    #setting folder path for each boat type
    files = glob.glob (paths + "/*.jpg")
    found.append((paths.split('\\')[-1],paths.split('\\')[-1]))

    #itering all files under the folder one by one
    for myFile in files:
        img = Image.open(myFile)
        #img.thumbnail((width, height), Image.ANTIALIAS) # resizes image in-place keeps ratio
        img = img.resize((224,224), Image.ANTIALIAS) # resizes image without ratio
        #convert the images to numpy arrays
        img = np.array(img)
        if img.shape == ( 224, 224, 3):
            # Add the numpy image to matrix with all data
            X_data.append (img)
            Y_data.append (id_no)
    id_no+=1


from keras.preprocessing.image import ImageDataGenerator
#Define arguments for ImageDataGenerator Class
train_datagen = ImageDataGenerator(rescale = 1./255,shear_range = 0.2,zoom_range =
0.2,horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)

#Applying ImageDataGenerator functionality to trainset and testset

path="/content/drive/MyDrive/Colab Notebooks/Digital Naturalist/Digital Naturalist Dataset"

x_train=train_datagen.flow_from_directory(path,target_size = (64,64),batch_size =
32,class_mode = "categorical")
x_test=test_datagen.flow_from_directory(path,target_size = (64,64),batch_size = 32,class_mode
= "categorical")

#Data splitting into Train And Test

#to see our data
print(X_data)
print(Y_data)

#converting lists to np arrays again
X = np.array(X_data)
Y = np.array(Y_data)

# Print shapes to see if they are correct
print("x-shape",X.shape,"y shape", Y.shape)

#The Keras library offers a function called to_categorical() that you can use to one hot enode
#integer data. The sequence has an example of all known values
#so we can use the to_categorical() function directly
X = X.astype('float32')/255.0
```

```
y_cat = to_categorical(Y_data, len(subfolders))

print("X shape",X,"y_cat shape", y_cat)
print("X shape",X.shape,"y_cat shape", y_cat.shape)

#Splitting the data to Test and Train

X_train, X_test, y_train, y_test = train_test_split(X, y_cat, test_size=0.2)
print("The model has " + str(len(X_train)) + " inputs")

#Getting Started with Convolutional Neural Networks (CNN)


#MODEL BUILDING

early_stop_loss = EarlyStopping(monitor='loss', patience=3, verbose=1)
early_stop_val_acc = EarlyStopping(monitor='val_accuracy', patience=3, verbose=1)
model_callbacks=[early_stop_loss, early_stop_val_acc]

#Add Layers(Conv, Maxpool, Flatten, Dense, Dropout)

#defining our model, All the layers and configurations
def load_CNN(output_size):
  K.clear_session()
  model = Sequential()
  model.add(Dropout(0.4,input_shape=(224, 224, 3)))

  model.add(Conv2D(256, (5, 5),input_shape=(224, 224, 3),activation='relu'))
  model.add(MaxPool2D(pool_size=(2, 2)))
  #model.add(BatchNormalization())

  model.add(Conv2D(128, (3, 3), activation='relu'))
  model.add(MaxPool2D(pool_size=(2, 2)))
  #model.add(BatchNormalization())

  model.add(Conv2D(64, (3, 3), activation='relu'))
  model.add(MaxPool2D(pool_size=(2, 2)))
  #model.add(BatchNormalization())

  model.add(Flatten())
  model.add(Dense(512, activation='relu'))
  model.add(Dropout(0.3))
  model.add(Dense(256, activation='relu'))
  model.add(Dropout(0.3))
  model.add(Dense(128, activation='relu'))
  model.add(Dropout(0.3))

  model.add(Dense(output_size, activation='softmax'))

  return model
```

#Building Model(Summary, Compile, Fit, Predict)

```
**#Model Summary**
#Building a model based on the above defined function
model = load_CNN(6) #Number of Columns / Outputs
model.compile(loss='categorical_crossentropy',optimizer=Adam(lr=0.001),metrics=['accuracy'])
model.summary() #to print model summary
weights = model.get_weights() #to get the weights from our model


**#Fitting Model**
#some arrays to store the result of each model (model trained on each bath size)
histories_acc = []
histories_val_acc = []
histories_loss = []
histories_val_loss = []


model.set_weights(weights)
h=model.fit(X_train,y_train,
            batch_size=16,
            epochs=7,
            verbose=1,
            callbacks=[early_stop_loss],
            shuffle=True,
            validation_data=(X_test, y_test))

model.summary() #to print model summary

model.set_weights(weights)
```

```
h=model.fit(X_train,y_train,
            batch_size=16,
            epochs=15,
            verbose=1,
            callbacks=[early_stop_loss],
            shuffle=True,
            validation_data=(X_test, y_test))
```

#Evaluation And Model Saving

```
**#Accuracy, Loss**
#printing the keys we have for the stores values
print(h.history.keys())

histories_acc = []
histories_val_acc = []
histories_loss = []
histories_val_loss = []

#appending the data for each epoch in a arr, and for each batch size
histories_acc.append(h.history['accuracy'])
histories_val_acc.append(h.history['val_accuracy'])
histories_loss.append(h.history['loss'])
histories_val_loss.append(h.history['val_loss'])

#converting into numpy arrays
histories_acc = np.array(histories_acc)
histories_val_acc = np.array(histories_val_acc)
histories_loss = np.array(histories_loss)
histories_val_loss = np.array(histories_val_loss)

#here we have 3 columns and 6 rows each,ever row representss differnet bath size,
#every column represent different epoch scores.
print('histories_acc',histories_acc ,
     'histories_loss', histories_loss,
     'histories_val_acc', histories_val_acc,
     'histories_val_loss', histories_val_loss)
```

*****Loading a Test Image & Making a Test Prediction*****

```
#Predicting the image's classes
#individual scores for each class as well as class with the highest score is printed

#making predictions ,storing result as array of probabilities of each class predicted
predictions = model.predict([X_test[8].reshape(1, 224,224,3)])
predictions


for idx, result, x in zip(range(0,6), found, predictions[0]):
  print("Label: {}, Type : {}, Species : {} , Score : {}%".format(idx, result[0],result[1], round(x*100,3)))

#predicting the class with max probability
ClassIndex=np.argmax(model.predict([X_test[image_number-1].reshape(1, 224,224,3)]),axis=1)
#getting the index of the class which we can pass
#to the boat_types list to get the boat type name
ClassIndex

print(found[ClassIndex[0]])

#loading Test Data
image_number = random.randint(0,len(X_test))
print(image_number)
#plotting the test image
plt.figure(figsize=(8, 8))
plt.imshow(X_test[image_number])

#loading Test Data
image_number = random.randint(0,len(X_test))
print(image_number)

#plotting the test image
plt.figure(figsize=(8, 8))
plt.imshow(X_test[image_number])
```

#Model Saving and Loading

```
h5_path=r'/content/drive/MyDrive/Colab Notebooks/Digital Naturalist/final_model.h5'
model.save(h5_path)

#saving necessary model files
model_json = model.to_json() #indent=2
```

```
with open("final_model.json", "w") as json_file:
    json_file.write(model_json)

# serialize weights to H5
model.save_weights("final_model.h5")
print("Saved model to disk")
```

GITHUB AND PROJECT LINK

GITHUB LINK:

https://github.com/IBM-EPBL/IBM-Project-3985-1658678251

PROJECT DEMO LINK:

https://drive.google.com/file/d/1nof5vhtf6lu9zM3PcccFRVtbjmxj90ho/view?usp=share_link