

Project Report

1. INTRODUCTION

- 1.1 Project Overview
- 1.2 Purpose

2. LITERATURE SURVEY

- 2.1 Existing problem
- 2.2 References
- 2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

- 3.1 Empathy Map Canvas
- 3.2 Ideation & Brainstorming
- 3.3 Proposed Solution
- 3.4 Problem Solution fit

4. REQUIREMENT ANALYSIS

- 4.1 Functional requirement
- 4.2 Non-Functional requirements

5. PROJECT DESIGN

- 5.1 Data Flow Diagrams
- 5.2 Solution & Technical Architecture

5.3 User Stories

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Feature 1

7.2 Feature 2

7.3 Database Schema (if Applicable)

8. TESTING

8.1 Test Cases

8.2 User Acceptance Testing

9. RESULTS

9.1 Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

1 PROJECT REPORT

1.1Project overview

A naturalist is someone who studies the patterns of nature, identifies a different kind of flora and fauna in nature. Being able to identify the flora and fauna around us often leads to an interest in protecting wild spaces, and collecting and sharing information about the species we see on our travels is very useful for conservation groups like NCC.

When venturing into the woods, field naturalists usually rely on common approaches like always carrying a guidebook around everywhere or seeking help from experienced ornithologists. There should be a handy tool for them to capture, identify and share the beauty to the outside world.

Field naturalists can only use this web app from anywhere to identify the birds, flowers, mammals and other species they see on their hikes, canoe trips and other excursions.

In this project, we are creating a web application which uses a deep learning model, trained on different species of birds, flowers and mammals (2 subclasses in each for a quick understanding) and get the prediction of the bird when an image is been given.

1.2 Purpose

A Naturalist is someone who studies the patterns of nature, identifies a different kind of flora and fauna in nature. Being able to identify the flora and fauna around us often leads to an interest in protecting wild spaces, and collecting and sharing information about the species we see on our travels is very useful for conservation groups like NCC. When venturing into the woods, field naturalists usually rely on common approaches like always carrying a guidebook around everywhere or seeking help from experienced ornithologists. There should be a handy tool for them to capture, identify and share the beauty to the outside world. Field naturalists can only use this web app from anywhere to identify the birds, flowers, mammals and other species they see on their hikes, canoe trips and other excursions. In this project, we are creating a web application which uses a deep learning model, trained on different species of birds, flowers and mammals (2 sub classes in each for a quick understanding) and get the prediction of the bird when an image is been given

2 . LITERATURE SURVEY

1	Paper title	“Rare Animal Image Recognition Based on Convolutional Neural Networks” .Hao, Xinyu, Guangsong Yang, Qiubo Ye, and Donghai Lin. In 2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), pp. 1-5. IEEE, 2019.
	Problem definition	<ul style="list-style-type: none">• Rare animal image recognition based on the basic model of CNNs, by which to autonomously extract the image features in the training set• Construct an image recognition system to identify rare animals
	Methodology/ Algorithm	<ul style="list-style-type: none">• Convolutional neural networks(CNN)• Matrix Multiple CNN (MMCNN)• Deep learning Convolutional neural network
	Advantages	<ul style="list-style-type: none">• Compared with ordinary neural networks, the advantages of simple operation and small computational complexity are very beneficial for the application Compared with ordinary neural networks• the advantages of simple operation and small computational complexity are very beneficial for the application and promotion of many industries.• The subsequent work of this research is to improve the network structure to improve the recognition accuracy while reducing the computational complexity.

	Disadvantages	<ul style="list-style-type: none"> • The subsequent work of this research is to improve the network structure to improve the recognition accuracy while reducing the computational complexity.
--	---------------	---

2	Paper title	“Image Classification Using Deep Neural Network”. Tiwari, Vaibhav, Chandrasen Pandey, Ankita Dwivedi, and Vrinda Yadav. In 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), pp. 730-733. IEEE, 2020.
	Problem definition	<ul style="list-style-type: none"> • Image Classification is widely used in various fields such as Plant leaf disease classification, facial expression classification. • To make bulky images handy, image classification is done using the concept of a deep neural network.
	Methodology/ Algorithm	<ul style="list-style-type: none"> • Deep Neural Network • VGG , • Image Classification • Convolutional Neural Network (CNN)
	Advantages	<ul style="list-style-type: none"> • An initial interesting point is that the common design principles of the VGG models since it performed best in the competition called ILSVRC 2014[10] • It is very simple and easy to comprehend and implement this modular construction of the architecture.
	Disadvantages	<ul style="list-style-type: none"> • It is extremely expensive to train due to complex data models.

		<ul style="list-style-type: none"> Moreover deep learning requires expensive GPUs and hundreds of machines. This increases cost to the users.
--	--	--

3	Paper title	“Convolutional Network based Animal Recognition using YOLO and Darknet”.Reddy, B. Karthikeya, Shahana Bano, G. Greeshmanth Reddy, Rakesh Kommineni, and P. Yaswanth Reddy. In 2021 6th International Conference on Inventive Computation Technologies (ICICT), pp. 1198-1203. IEEE, 2021.
	Problem definition	<ul style="list-style-type: none"> The main goal of this research work to build animal an recognition methodology using YOLOV3 model. The image of animal will be given as input, then it will display the name of the animal as output by using YOLOV3 model. The detectionis done by using a pre-trained coco dataset from darknet.
	Methodology/ Algorithm	<ul style="list-style-type: none"> YOLO V3 Darknet Convolutional network Detector Opencv
	Advantages	The image which are predicted correct type of animal name
	Disadvantages	<ul style="list-style-type: none"> Wrong output means the images which are predicted a different name rather than the correct name of the given input image. No output means it is not able to predict the given input images.

4	Paper title	“Automatic Bird-Species Recognition using the Deep Learning and Web Data Mining”.Kang, Min-Seok, and Kwang-Seok Hong. In 2018 International Conference on Information and Communication Technology Convergence (ICTC), pp. 1258-1260. IEEE, 2018.
	Problem definition	<ul style="list-style-type: none"> • First, if you enter the name of the targeted bird breed, the image will be collected from the Web using the image crawl. • To refine the collected images into the training dataset, the corrupted image is corrected and deleted, the outlier is removed, and finally the image is expanded to obtain the refined training data.
	Methodology/ Algorithm	<ul style="list-style-type: none"> • Deep Neural Network (DNN) • Convolutional Neural Network (CNN) • Tensorflow Framework • Back Propagation
	Advantages	<ul style="list-style-type: none"> • It is used in various applications like the image recognition, video analysis, natural language processing, and drug discovery • The performances are improving annually.
	Disadvantages	<ul style="list-style-type: none"> • Birdwatching is a common hobby but to identify their species requires the assistance of bird books.

5	Paper title	“Detection and classification of opened and closed flowers in grape inflorescences using Mask R-CNN”. Pahalawatta, Kapila, Jaco Fourie, Amber Parker, Peter Carey, and Armin Werner. In 2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ), pp. 1-6. IEEE, 2020.
	Problem definition	<ul style="list-style-type: none"> • This is because it involves the processing of images with varying image qualities, and also because of the close similarity in images between the two classes of interests, opened and closed flowers. • Our aim is to build a system with one of the most promising deep learning object detection networks, Mask R-CNN, to detect the individual instances of the above two classes separately using the images with no prior alterations
	Methodology/ Algorithm	<ul style="list-style-type: none"> • R- Convolutional Neural Network (R-CNN) • Convolutional Neural Network (CNN)
	Advantages	<ul style="list-style-type: none"> • The similarity of instance shapes between the two classes, opened and closed flowers, and also the similarity of pixel texture between opened and closed flowers makes the purely image processing based instance segmentation a challenging task.
	Disadvantages	<ul style="list-style-type: none"> • Model accuracy was tested by letting the model extract and segment flower instances from images that were not in the training set.

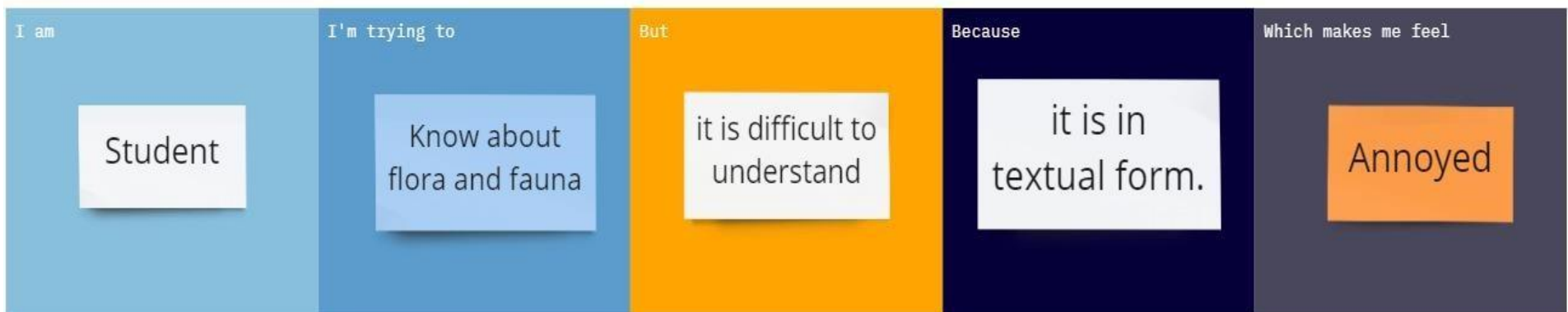
2.1 Problem Statement

Digital Naturalist - AI Enabled Tool For Biodiversity Researchers

Animal classification in wildlife in many fields has highlighted the importance of algorithms for this kind of problems. However, there have been only a few attempts to solve this problem, mainly focused at the detecting and tracking for mostly popular pet like cat, dog, cow, etc. In this scope of this project, we focus on some algorithms for animal classification, one using images and statistical methods and another using decision tree

To build a web application which uses **A DEEP LEARNING MODEL**, trained on different species of birds, flowers and mammals and get the prediction of the bird when an image is been given, by using **IBM WATSON ASSISTANT** to effectively curb out the following constraints :

- To educate hikers, travelers about the species they were looking at.
- To differentiate the variety of breed among the same species.
- To identify the birds, flowers, mammals and other species.
- To recognize each and every flora and fauna that the AI tool captures
- To provide the user friendly interface to identify the species.



2.2 REFERENCES

1. R. L. Siegel, K. D. Miller, and A. Jemal, “Cancer statistics, 2016,” *CA: A Cancer Journal for Clinicians*, vol. 66, no. 1, pp. 7–30, 2016. View at: [Publisher Site](#) | [Google Scholar](#)
2. L. Fan, K. Strasser-Weippl, J.-J. Li et al., “Breast cancer in China,” *Lancet Oncology*, vol. 15, no. 7, pp. e279–e289, 2014. View at: [Publisher Site](#) | [Google Scholar](#)
3. L. Tabár, A. Gad, L. H. Holmberg et al., “Reduction in mortality from breast cancer after mass screening with mammography,” *The Lancet*, vol. 325, no. 8433, pp. 829–832, 1985. View at: [Google Scholar](#)
4. S. Yu, S. Wu, L. Zhuang et al., “Efficient segmentation of a breast in B-mode ultrasound tomography using three-dimensional GrabCut (GC3D),” *Sensors*, vol. 17, no. 8, p. 1827, 2017. View at: [Publisher Site](#) | [Google Scholar](#)

5. R. Longo, F. Arfelli, R. Bellazzini et al., “Towards breast tomography with synchrotron radiation at Elettra: first images,” *Physics in Medicine and Biology*, vol. 61, no. 4, pp. 1634–1649, 2016. View at: [Publisher Site](#) | [Google Scholar](#)
6. M. J. Michell, A. Iqbal, R. K. Wasan et al., “A comparison of the accuracy of film-screen mammography, full-field digital mammography, and digital breast tomosynthesis,” *Clinical Radiology*, vol. 67, no. 10, pp. 976–981, 2012. View at: [Publisher Site](#) | [Google Scholar](#)
7. F. M. Hall, J. M. Storella, D. Z. Silverstone, and G. Wyshak, “Nonpalpable breast lesions: recommendations for biopsy based on suspicion of carcinoma at mammography,” *Radiology*, vol. 167, no. 2, pp. 353–358, 1988. View at: [Publisher Site](#) | [Google Scholar](#)
8. J. Tang, R. M. Rangayyan, J. Xu, I. El Naqa, and Y. Yang, “Computer-aided detection and diagnosis of breast cancer with mammography: recent advances,” *IEEE Transactions on Information Technology in Biomedicine*, vol. 13, no. 2, pp. 236–251, 2009. View at: [Publisher Site](#) | [Google Scholar](#)
9. N. I. R. Yassin, S. Omran, E. M. F. El Houby, and H. Allam, “Machine learning techniques for breast cancer computer aided diagnosis using different image modalities: a systematic review,” *Computer Methods and Programs in Biomedicine*, vol. 156, pp. 25–45, 2018. View at: [Publisher Site](#) | [Google Scholar](#)

10. Y. Jiang, R. M. Nishikawa, R. A. Schmidt, C. E. Metz, M. L. Giger, and K. Doi, “Improving breast cancer diagnosis with computer-aided diagnosis,” *Academic radiology*, vol. 6, no. 1, pp. 22–33, 1999. View at: [Publisher Site](#) | [Google Scholar](#)
11. H. D. Cheng, J. Shan, W. Ju, Y. Guo, and L. Zhang, “Automated breast cancer detection and classification using ultrasound images: a survey,” *Pattern Recognition*, vol. 43, no. 1, pp. 299–317, 2010. View review,” *Clinical Imaging*, vol. 37, no. 3, pp. 420–426, 2013. View at: [Publisher Site](#) | [Google Scholar](#)
12. S. Mambou, P. Maresova, O. Krejcar, A. Selamat, and K. Kuca, “Breast cancer detection using infrared thermal imaging and a deep learning model,” *Sensors*, vol. 18, no. 9, p. 2799, 2018. View at: [Publisher Site](#) | [Google Scholar](#)
13. H. D. Cheng, X. Cai, X. Chen, L. Hu, and X. Lou, “Computer-aided detection and classification of microcalcifications in mammograms: a survey,” *Pattern Recognition*, vol. 36, no. 12, pp. 2967–2991, 2003. View at: [Publisher Site](#) | [Google Scholar](#)
14. R. M. Rangayyan, F. J. Ayres, and J. E. Leo Desautels, “A review of computer-aided diagnosis of breast cancer: toward the detection of subtle signs,” *Journal of the Franklin Institute*, vol. 344, no. 3-4, pp. 312–348, 2007. View at: [Publisher Site](#) | [Google Scholar](#)

15. I. Christoyianni, A. Koutras, E. Dermatas, and G. Kokkinakis, “Computer aided diagnosis of breast cancer in digitized mammograms,” *Computerized Medical Imaging and Graphics*, vol. 26, no. 5, pp. 309–319, 2002. View at: [Publisher Site](#) | [Google Scholar](#)
16. L. Wei, Y. Yang, R. M. Nishikawa, and Y. Jiang, “A study on several machine-learning methods for classification of malignant and benign clustered microcalcifications,” *IEEE Transactions on Medical Imaging*, vol. 24, no. 3, pp. 371–380, 2005. View at: [Publisher Site](#) | [Google Scholar](#)
17. Y. Bengio, A. Courville, and P. Vincent, “Representation learning: a review and new perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013. View at: [Publisher Site](#) | [Google Scholar](#)
18. Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. View at: [Publisher Site](#) | [Google Scholar](#)
19. Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, “Deep learning for visual understanding: a review,” *Neurocomputing*, vol. 187, pp. 27–48, 2016. View at: [Publisher Site](#) | [Google Scholar](#)

2.3 EXISTING SYSTEM:

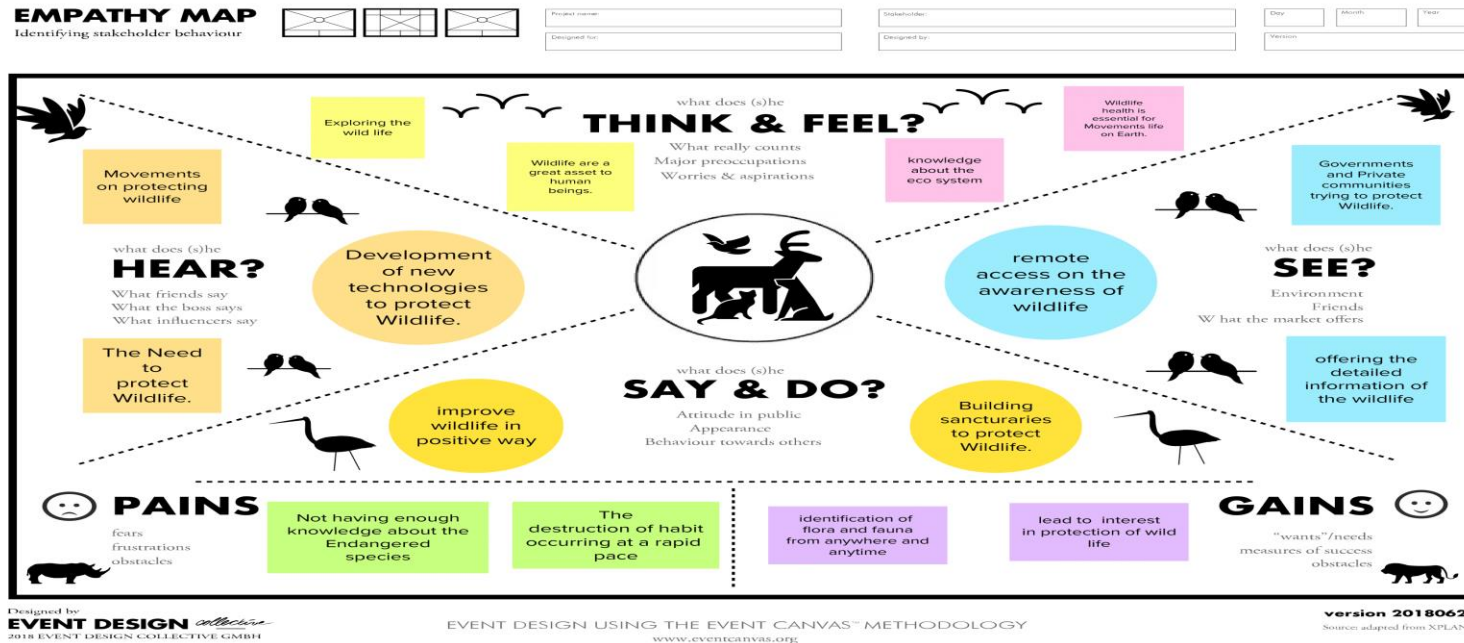
The goal is to discover ways that DIY technology, created in the wild, can let us explore nature in new ways. The key relationship in this work is between field biologists (of any level) and technologists (of any ability), thus many of our activities will involve hybrid artistic and scientific examinations of the wilderness surrounding us. For instance we may develop biological tools for studying nearby creatures, and then adapt these into artistic devices for continued exploration and sharing of this phenomena.

Open Endess encourages creating tools for general exploration increasing chances of serendipitously stumbling across interesting new phenomena. Making simple capacitive touch sensor probes that we can connect to nearby flora, for instance, lets us openly poke and probe novel questions in the environment. We might also program generalised tools, such as robotic arms, to poke and probe different ecological systems in multiple ways

3 Ideation Phase

3.1 Empathy Map


Digital Natural-AI enabled tool



3.2 Brainstorm & Idea Prioritization Template

Step-1: Team Gathering, Collaboration and Select the Problem Statement

Template




Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- 🕒 10 minutes to prepare
- 🕒 1 hour to collaborate
- 👥 2-8 people recommended

Digital Naturalist - AI Enabled tool for Biodiversity Researchers

Share template feedback



Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 10 minutes

A

Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B

Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

C

Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

Open article →

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


PROBLEM


In this project, we are creating a web application which uses a deep learning model, trained on different species of birds, flowers and mammals (2 subclasses in each for a quick understanding) and get the prediction of the bird when an image is been given





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.

Step-2: Brainstorm, Idea Listing and Grouping

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 icon to start drawing.

MUKESH KUMAR S	Interactive app with camera interface	CNN based Model	ML model for classification	Hyper-parameter Tuning
ARAVIND P	Responsive UI	Data collection	Image processing	Building HTML page
JANAKIRAMAN R	Data Augmentation	Deep learning models	Data Preprocessing	Computing model accuracy
THARUN KUMAR T	One hot encoding	Real life implementation	Accessible UI	choice of optimizer and loss funtion

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 can break it up into smaller sub-groups.

🕒 20 minutes

User Interface(UI)	Machine learning model
Interactive app with camera interface	ML model for classification
Responsive UI	choice of optimizer and loss funtion
Accessible UI	CNN based Model
Building HTML page	Computing model accuracy
Model training	Data Preprocessing
Hyper-parameter Tuning	Data Preprocessing
One hot encoding	Data collection
Real life implementation	Data Augmentation
Deep learning models	Image processing

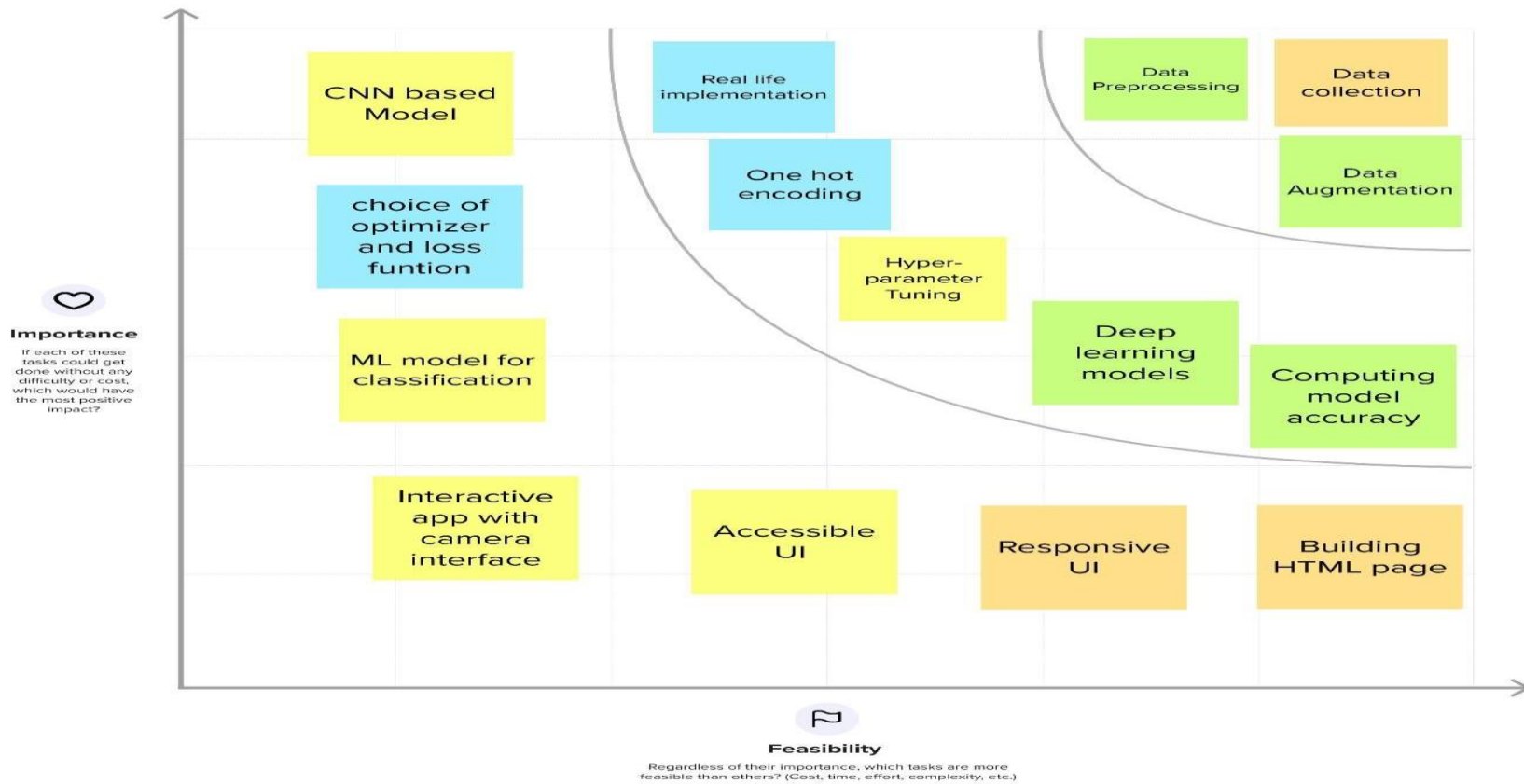
Step-3: Idea Prioritization

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

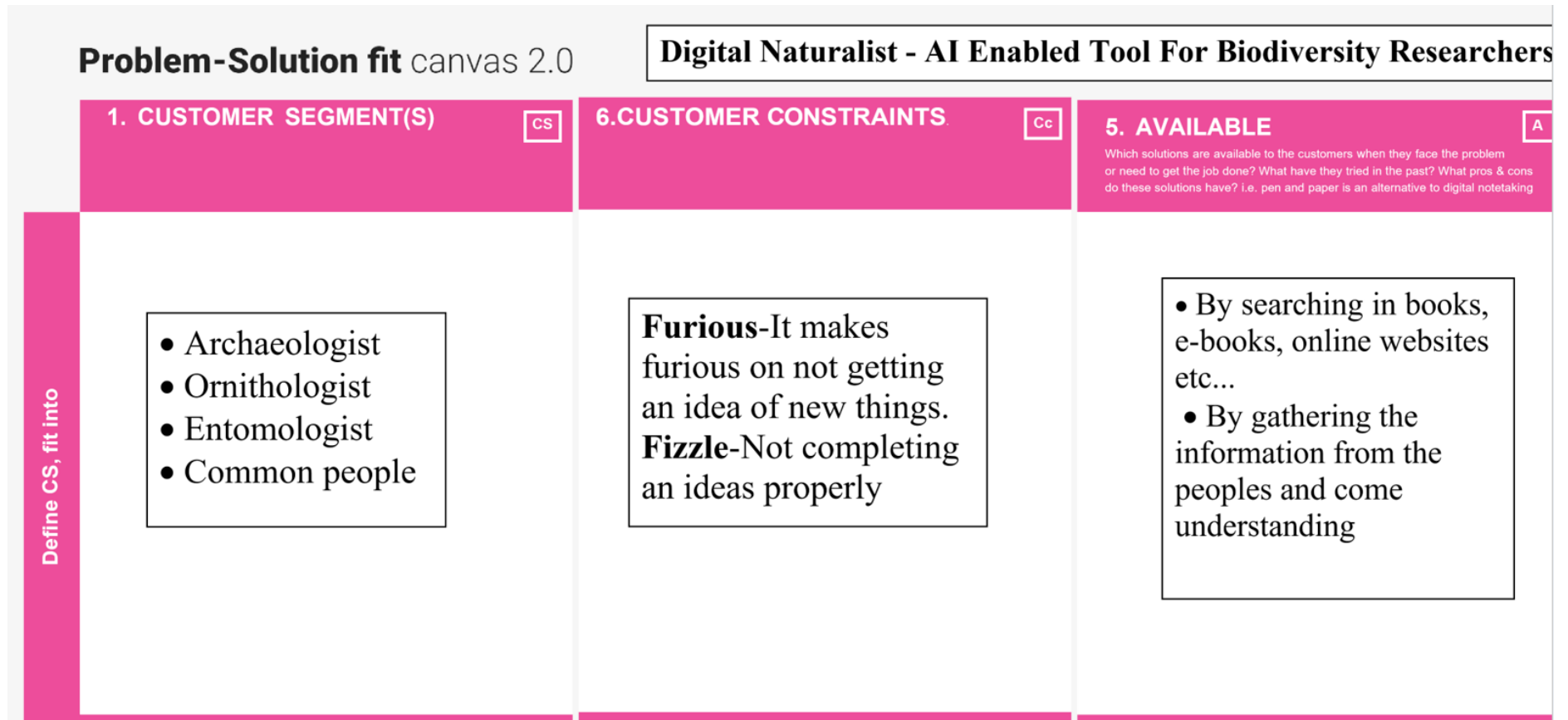


3.3 Proposed Solution Template:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	<ul style="list-style-type: none">• We are creating a web application which uses a deep learning model, trained on different species of birds, flowers and mammals (2 subclasses in each for a quick understanding)• Getting the prediction of the bird when an image is been given.
2.	Idea / Solution description	<ul style="list-style-type: none">• This system is built by using the Image/object recognition and classification using (CNN) Convolutional neural network.• By using this system, we can capture the image of any animals and plants and can obtain the information about the flora and fauna at any time
3.	Novelty / Uniqueness	<ul style="list-style-type: none">• We can get the resource of the wildlife, flora and fauna using this responsive UI design with highly automated machine learning algorithms• Anytime and anywhere we can get the resource it carries out the visualization and interpreted result

4.	Social Impact / Customer Satisfaction	<ul style="list-style-type: none"> • It satisfies the basic requirements of the customer it will be the bridge between the wildlife and researchers , customer ,wildlife photographers • image is given as input and it will predict with trained models and output will be displayed with respective to the input
5.	Business Model (Revenue Model)	<ul style="list-style-type: none"> • By using this system, the users can predict and analyse the picture of the animals or plants. • In which it results to the visualizing the description of the flora or fauna which taken as input.
6.	Scalability of the Solution	<ul style="list-style-type: none"> • By implementing in this project people can effectively gain more knowledge about the wildlife • it will be more effective to the wildlife photographers and it can be use any time at anywhere we wish . • the system can also be integrated with the future technology

3.4 Problem Solution Fit



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.

- Customer satisfaction is more important
- Providing the correct information to the researchers & customers, wildlife photographers
- easily accessible and it should be more user friendly
- negotiating the unwanted stuffs

9. PROBLEM ROOT CAUSE

R

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.

- Too much of data cannot be stored by any human or they may forget
- Need depend on experts
- Lack of study in the sequence of things
- Unaware of the object
- New to environment

7. BEHAVIOUR

B

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)

- When the user Don't have the knowledge about particular thing (flora and fauna) this kind of situation occurs.
- In important situation not gaining the proper information

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.

10. YOUR SOLUTION

SL

What kind of solution suits Customer scenario the best?
Adjust your solution to fit Customer behaviour, use Triggers, Channels & Emotions for marketing and communication.

8.1 ONLINE CHANNELS

CH

What kind of actions do customers take online? Extract online channels from box #7 Behaviour

Define CS, fit into CL

- Seeking for self-gratification by identity the thing
- Enthusiastic on learning new things
- To help peoples to get extra knowledge about the
- thing in (flora and fauna)

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.

BEFORE-Knowledge about the wildlife and flora and fauna will be quite lesser

AFTER-Gaining more information by a resource Will make a satisfaction on getting it

In this project, we are creating a web application which uses a deep learning model, trained on different species of birds, flowers and mammals (2 subclasses in each for a quick understanding) and get the prediction of the bird when an image is been given

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

8.2 OFFLINE CHANNELS

CH

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

- Using the guide books we can take the information about the species
- Collecting the information from the tourist guide will be

4 REQUIREMENT ANALYSIS

_4.1Non-Functional requirements

NFR-1	Usability	This tool verifies that usability is a special and important perspective to analyze user requirements, which can further improve the tool quality. In the model process with user experience as the core, the analysis of users' usability can indeed help designers better understand users' potential needs, behavior and experience.
NFR-2	Security	By identifying the danger and poisoning flora and fauna. which the human become more secure from the attack by animals.
NFR-3	Reliability	The conventional computer vision approach of image recognition is a sequence of image filtering, segmentation, feature extraction, and rule -based classification. The images from the created dataset are fed into a neural network algorithm. This is the deep or machine learning aspect of creating an image recognition model. The training of an image recognition algorithm makes it possible for

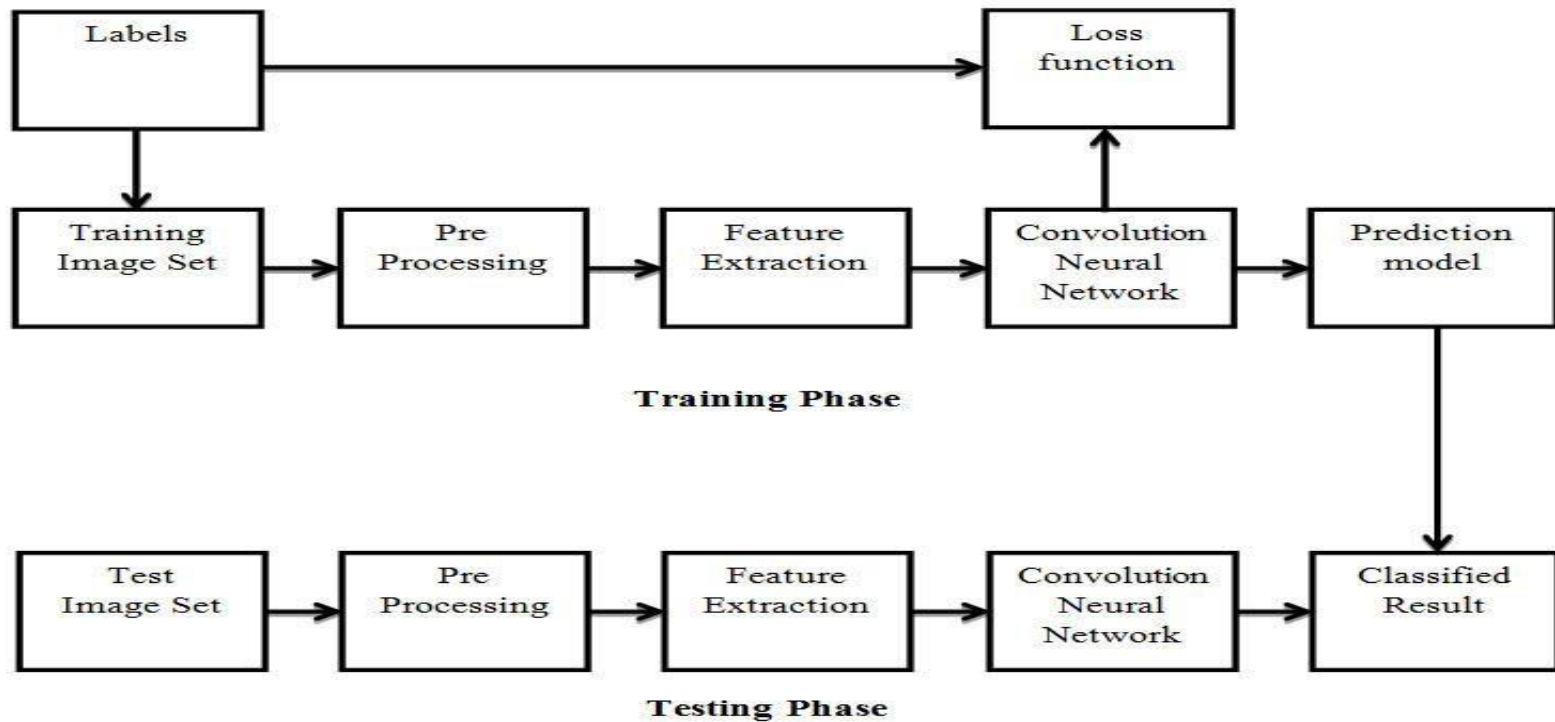
4.2Functional requirements

FR No.	Functional Requirement	Sub Requirement
FR-1	Classification:	It identifies the “class,” i.e., the category to which the image belongs. Note that an image can have only one class.
FR-2	Navigation Service	GPS
FR-3	Database	IBM Cloud
FR-4	Localization:	It helps in placing the image in the given class andcreates a bounding box around the object to show its location in the image
FR-5	Detection	It helps to categorize the multiple objects in the image and create a bounding box around it to locate each of them. It is a variation of the classification wit localization tasks for numerous objects
FR-6	Final Output	Final description of the image captured

5 Project Design

5.1 Data Flow Diagram

Data Flow Diagrams:



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Ui design	USN-1	The UI design will be created there will be box in that we can import the images	We can access the responsive UI	LOW	Sprint-1
Customer (Mobile user)	Image uploading	USN-2	The user can upload the picture of flora and fauna and get the detailed information of the species	I can access my library of choosing the data set	MEDIUM	Sprint-2
Customer (Mobile user)	Training Phase	USN-3	Training image set are pre processed and feature extraction is used to to reduce the amount of redundant data from the data set and using ccn algorithm it will predict the model	Training the given data set which we have give using CNN algorithm	HIGH	Sprint-3
Customer (Mobile user)	Testing phase	USN-4	Testing images are pre processed using feature extraction reduce the redundant data and CNN is used to classify the result	Testing the trained data set which we have give using CNN algorithm	HIGH	Sprint-3
Customer (Mobile user)	Predict the species	USN-5	The detailed information of the species is displayed in the webpage	CNN model predicts the species	LOW	Sprint-4

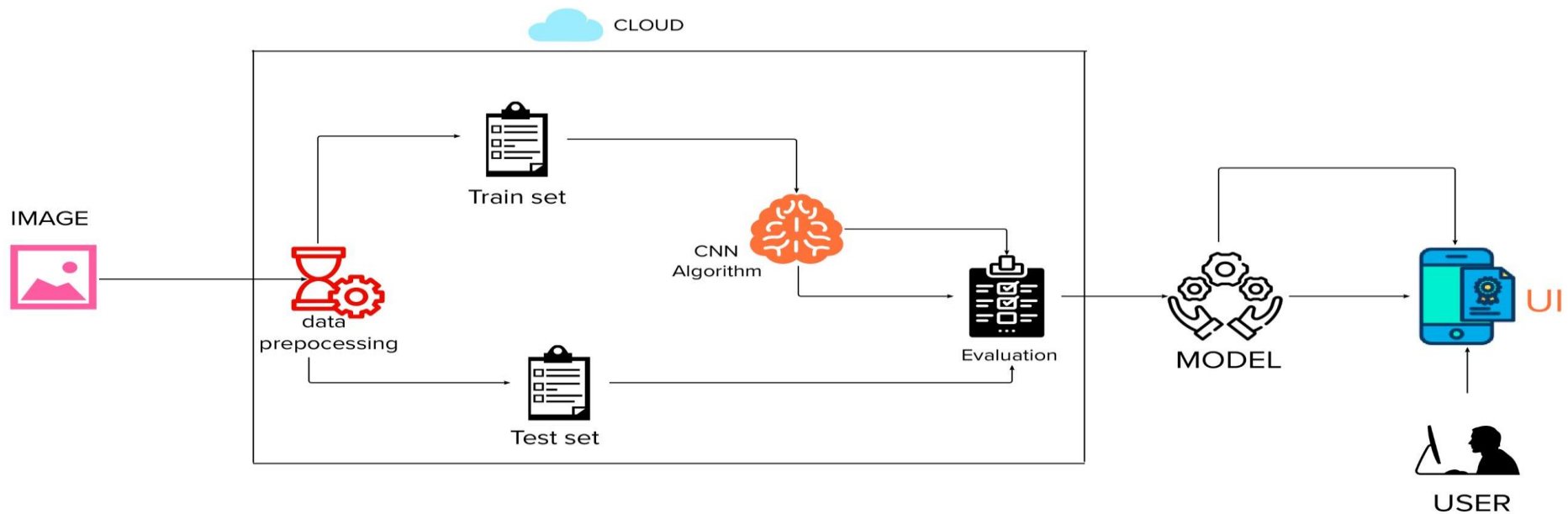


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Web UI or Website or Web app	HTML, CSS, JavaScript / React JS
2.	Application Logic-1	Model building and training	Python
3.	Application Logic-2	Getting image or text data from user for prediction	IBM Watson STT service
4.	Application Logic-3	Fetch the relevant data from the database and project them to user	IBM Watson Assistant

5.	Database	Image and text data of all the species along with detailed view of each species	MySQL/NoSQL
6.	Cloud Database	Fetch data from database and feed them to model for prediction and also used to retrieve the data required for user.	IBM DB2, IBM Cloudant etc.
7.	File Storage	Image data, login credentials, code (backend and frontend) and API keys	IBM Block Storage
8.	External API-1	To get data from the database when user give the image input	IBM Storage API
9.	External API-2	To get the username and password of the specific user	Authentication API, etc.
10.	Machine Learning Model	To predict the species (flora or fauna) through the image input and also it gives detailed view of the particular species	Image Recognition Model, etc.
11.	Infrastructure (Server / Cloud)	To deploy our 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 Service Gateway Interface)

2.	Security Implementations	For authenticating the user data and protecting the data about species in database	SHA-256 / Encryptions / IAM Controls / OWASP
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 that can handle wide range of requests at a time without any delay and to provide accuracy in prediction	IBM instance

6 Project Planning

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Ui design	USN-1	The home page looks attractive. When you click on the button “Drop the Scan ”	4	LOW	ARAVIND P
		USN-2	You will be redirected to predict section with responsive UI DESIGN	4	LOW	T THARUN KUMAR
Sprint-2	Image uploading	USN-3	The user can upload the picture of flora and fauna and get the detailed information of the species	5	MEDIUM	JANAKIRAMAN
		USN-4	Collecting the datasets of flora & fauna. Choosing the one among picture of the datasets and getting the information of the species	5	MEDIUM	MUKESH KUMAR
Sprint-3	Training Phase	USN-5	Dataset collection- Datasets are collected to train the model.	4	HIGH	JANAKIRAMAN
		USN-6	Data Pre-processing- The data is loaded and Preprocessed to train the model.	5	HIGH	T THARUN KUMAR

		USN-7	Build and Train the model- The model is trained using Training dataset.	8	HIGH	ARAVIND
	Testing Phase	USN-8	Unit test. Check the correctness of individual model components..	4	HIGH	MUKESH KUMAR
		USN-9	Minimum Functionality Test- The minimum functionality test helps you decide whether individual model components behave as you expect	4	HIGH	ARAVIND
		USN-10	Model testing involves explicit checks for behaviors that we expect our model to follow.	5	HIGH	MUKESH KUMAR S
Sprint-4	Predict the species	USN-11	Connecting the frontend and backend using API calls	4	MEDIUM	ARAVIND
		USN-12	cloud deployment – Deployment of application using IBM cloud	5	HIGH	MUKESH KUMAR
		USN-13	The detailed information of the species is displayed in the webpage	3	MEDIUM	JANAKIRAMAN

6.1 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	8	6 Days	24 Oct 2022	29 Oct 2022		
Sprint-2	10	6 Days	31 Oct 2022	05 Nov 2022		
Sprint-3	30	6 Days	07 Nov 2022	12 Nov 2022		
Sprint-4	12	6 Days	14 Nov 2022	19 Nov 2022		

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's

$$AV = \frac{\text{sprint duration}}{\text{velocity}} = \frac{20}{10} = 2$$

average velocity (AV) per iteration unit (story points per day)

Velocity:

For Sprint-1 the Average Velocity (AV) is:

$$AV = \text{Sprint Duration} / \text{velocity} = 8 / 6 =$$

1.3 For Sprint-2 the Average Velocity (AV) is:

$$AV = \text{Sprint Duration} / \text{velocity} = 10 / 6 =$$

1.6 For Sprint-3 the Average Velocity (AV) is:

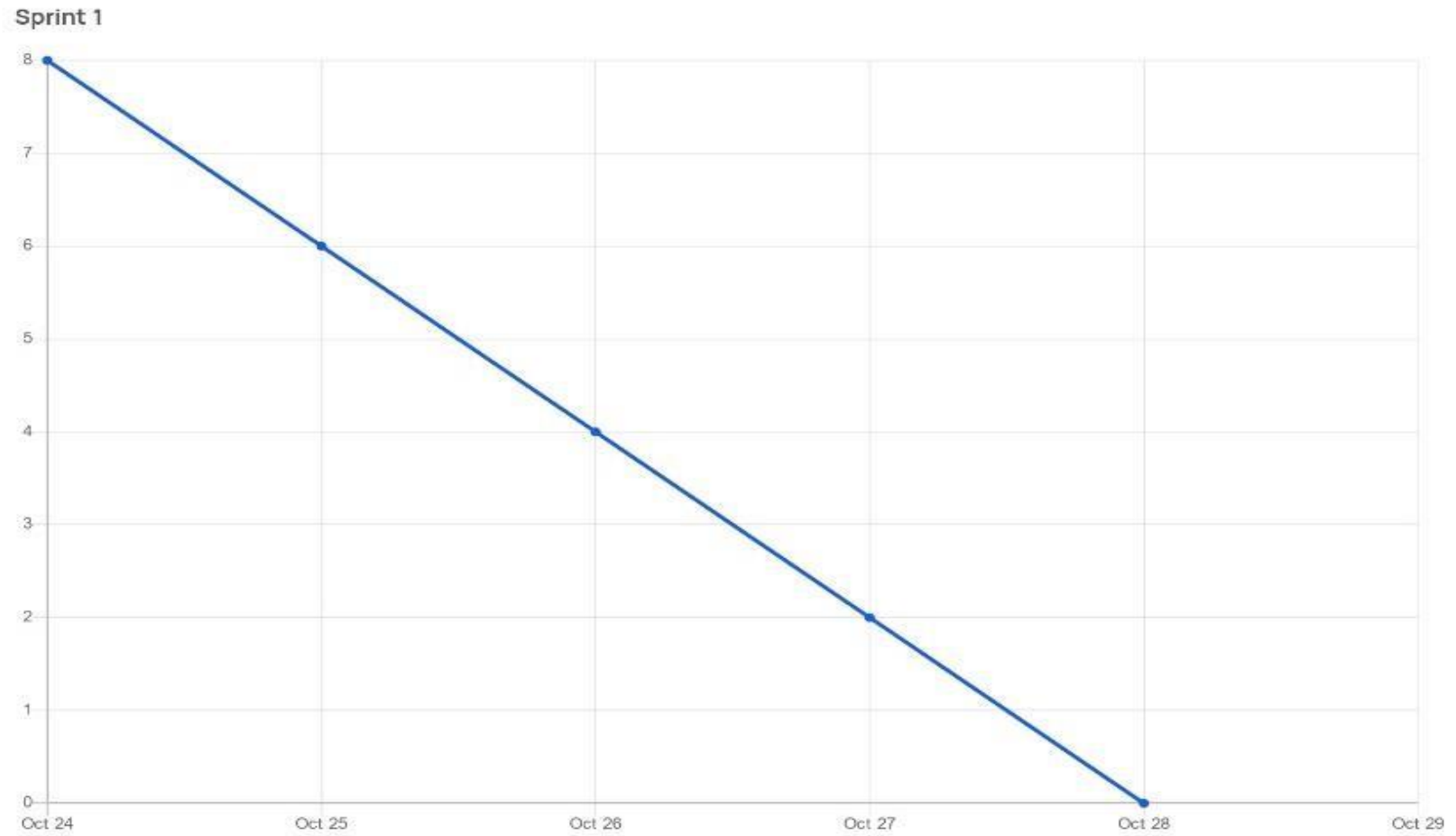
$$AV = \text{Sprint Duration} / \text{velocity} = 30 / 6 =$$

5 For Sprint-4 the Average Velocity (AV) is:

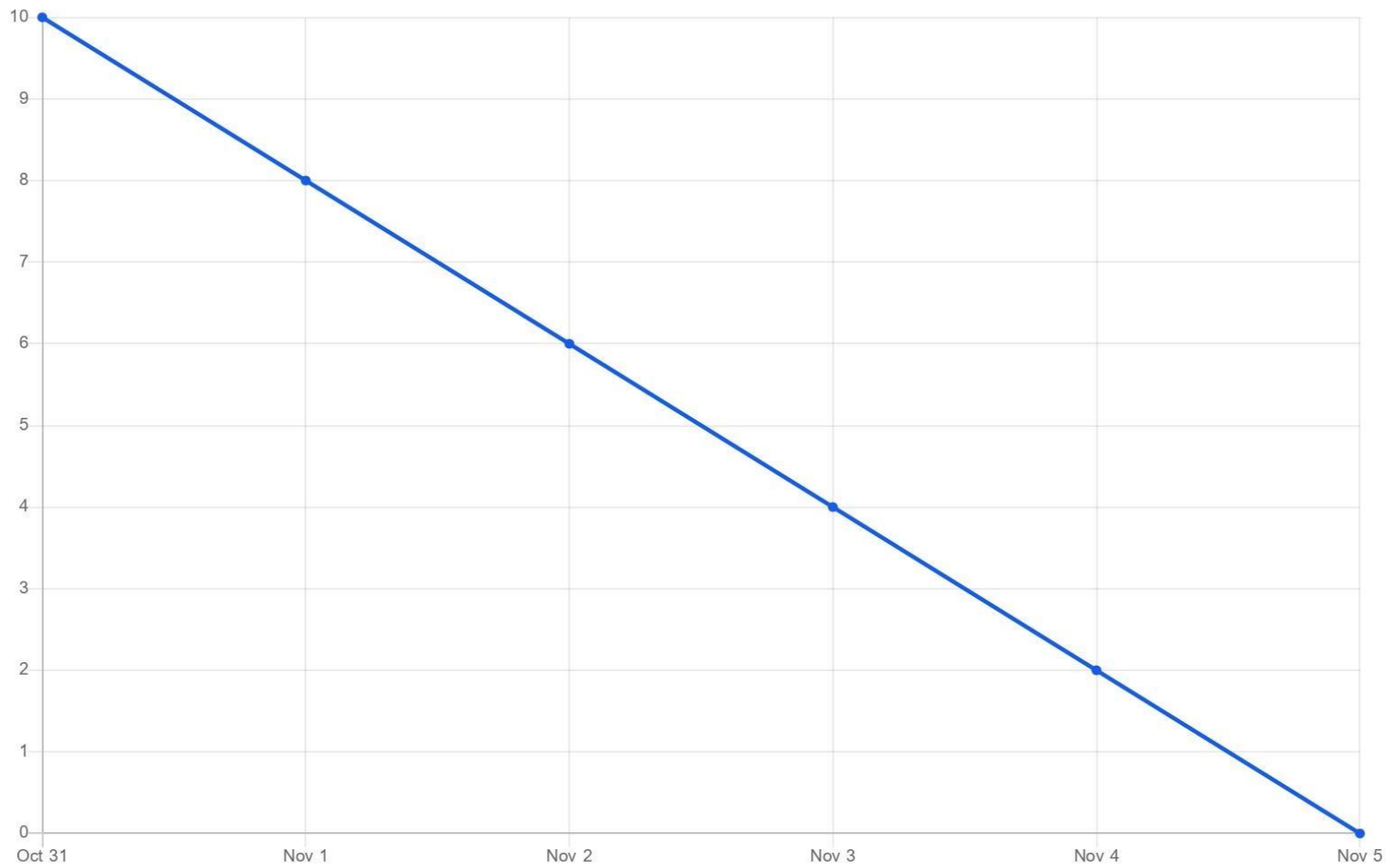
$$AV = \text{Sprint Duration} / \text{velocity} = 12 / 6 = 2$$

TOTAL AVERAGE VELOCITY = 2.475

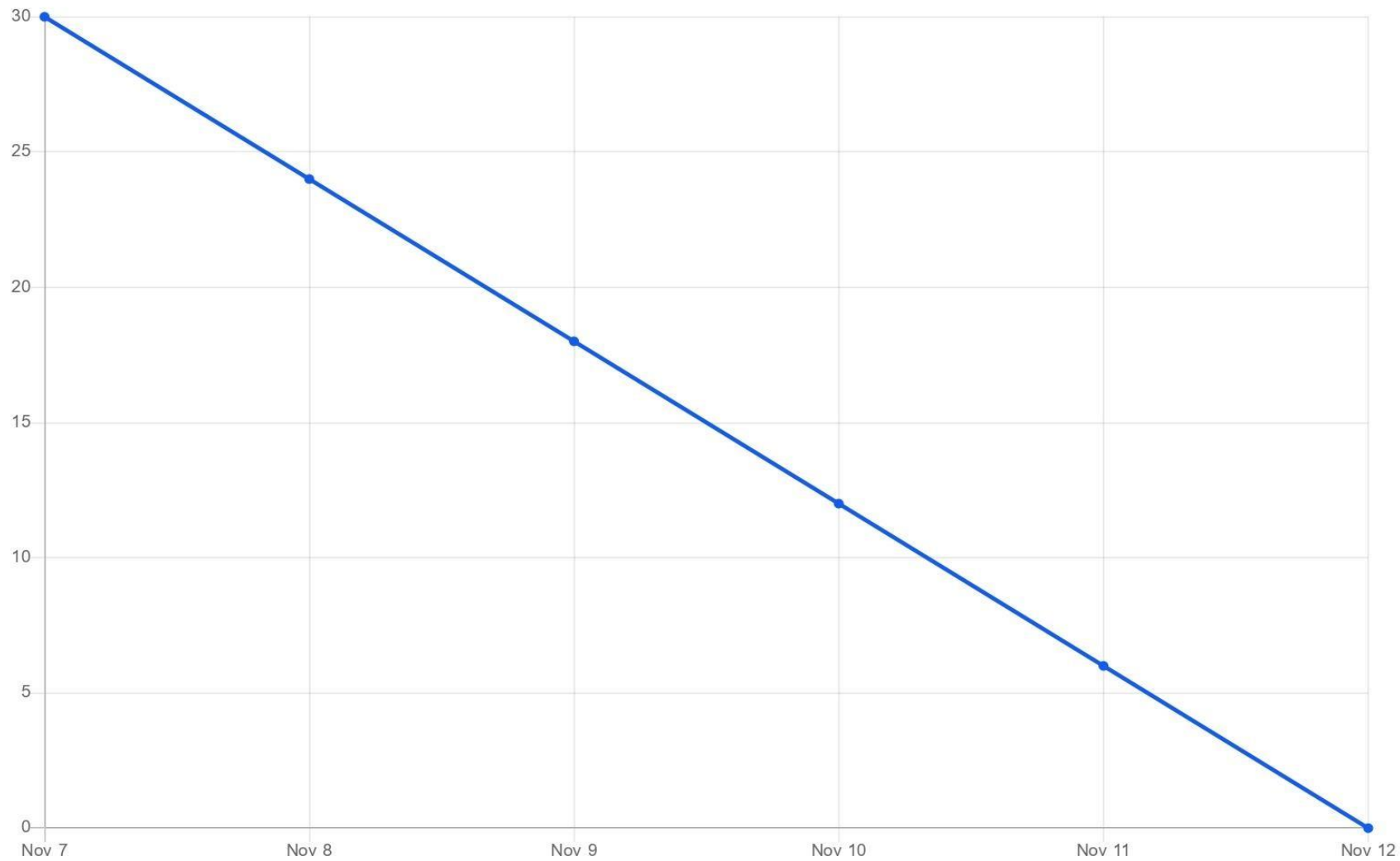
Burndown Chart:



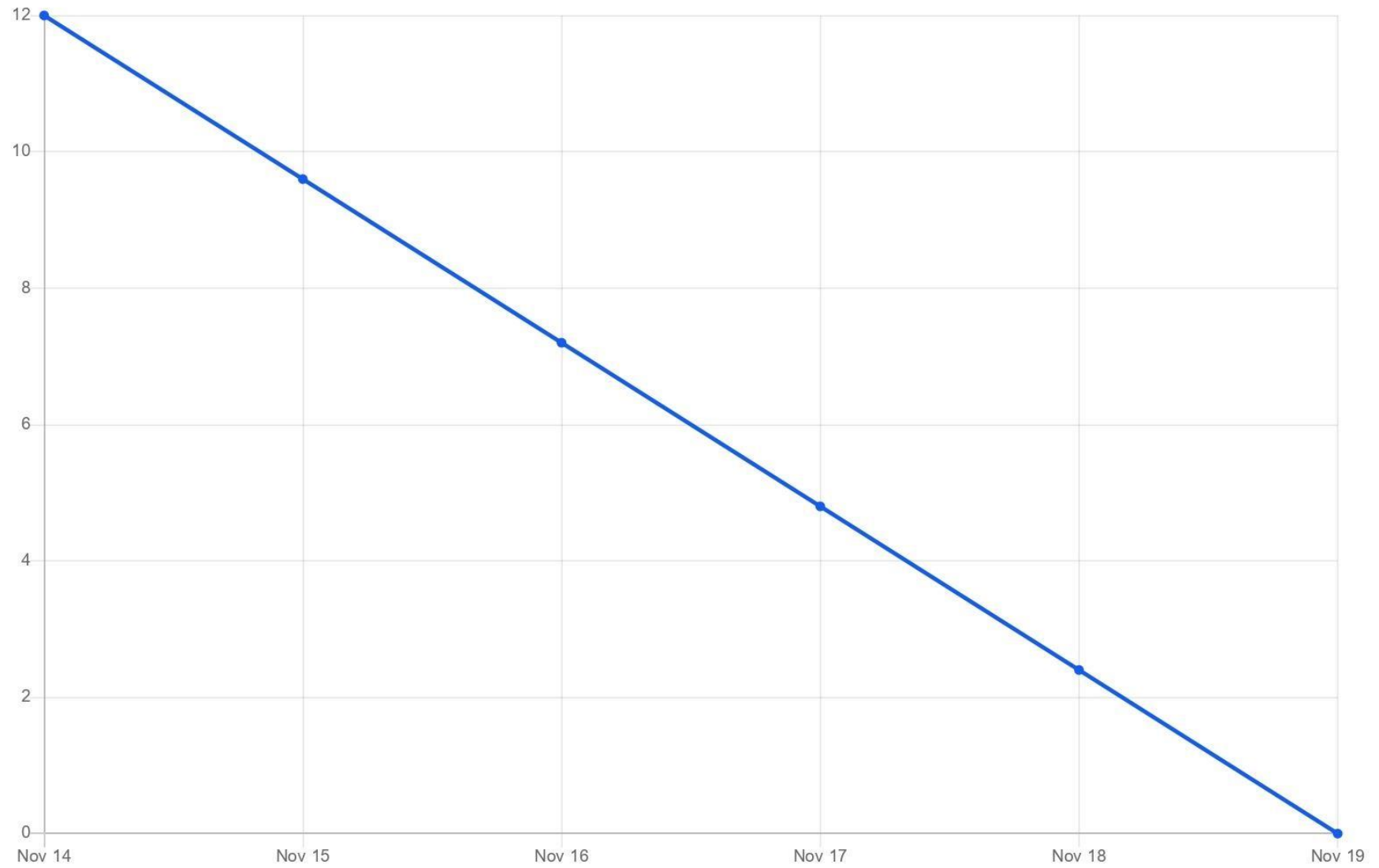
Sprint 2





















Sprint 3



Sprint 4



6.3 Reports from JIRA

	T	NOV				DEC
Sprints		Sprint...	Sprint 2	Sprint 3	Sprint 4	
▼ DNAETFBR-4 <u>Ui design</u> DNAETFBR-10 The home page looks attractive. When you click o... DONE ARAVIND P DNAETFBR-14 You will be redirected to predict section with resp... DONE THARUNKU...		  				
▼ DNAETFBR-5 <u>Image uploading</u> DNAETFBR-12 The user can upload the picture of flora and... IN PROGRESS JANAKIRA... DNAETFBR-13 Collecting the datasets of flora & fauna. Ch... IN PROGRESS MUKESH K...			  			
▼ DNAETFBR-6 <u>Training Phase</u> DNAETFBR-14 Dataset collection- Datasets are collected to train... TO DO JANAKIRA... DNAETFBR-15 Data Pre-processing- The data is loaded and Pr... TO DO THARUNKU... DNAETFBR-16 Build and Train the model- The model is trained u... TO DO ARAVIND P				   		
▼ DNAETFBR-7 <u>Testing Phase</u> DNAETFBR-17 Unit test. Check the correctness of individual mo... TO DO MUKESH K... DNAETFBR-18 Minimum Functionality Test- The minimum functio... TO DO ARAVIND P DNAETFBR-19 Model testing involves explicit checks for behavi... TO DO MUKESH K...				   		
▼ DNAETFBR-8 <u>Predict the species</u> DNAETFBR-20 Connecting the frontend and backend using API c... TO DO ARAVIND P DNAETFBR-21 cloud deployment – Deployment of application u... TO DO MUKESH K... DNAETFBR-22 The detailed information of the species is display... TO DO JANAKIRA...				   		

7 CODING & SOLUTIONING

7.1) Data Collection

You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository, etc. As of now, you can download in the next step.

i) Data Augmentation and structure creation

change the directory paths for loading a and saving according to how they appear on your system. You can create this manually or by using `os.mkdir()` function within python.

```

from keras.preprocessing.image import ImageDataGenerator
import cv2
import csv
from os import listdir
import time
from google.colab import drive
drive.mount('/content/drive')
# Nicely formatted time string to make a note of how much time it takes for augmentation
def hms_string (sec_elapsed):
    h = int(sec_elapsed / (60 * 60))
    m = int((sec_elapsed % (60 * 60)) / 60)
    s = sec_elapsed % 60
    return "{n}: {m}:{round(s,1)}"

def augment_data(file_dir, n_generated_samples, save_to_dir):
    """
    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.
    """
    #from keras.preprocessing.image import ImageDataGenerator
    #from os import listdir
    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

```

7.2) Loading Data And Preprocessing

In this, we process the image ,Importing The Libraries Import all the required libraries. Make sure to check you've everything mentioned in the Pre-Requisites in order to not face any problems while doing this.

```
#For matrix calculations and data Managememnt
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, array_to_img, img_to_array, 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.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 glob
from PIL import Image
import os
from os import listdir
```

7.3) Make A List Of Paths To All Folders Where You Have Data

Here we've made a list containing all the folders which are having the data we created for model training in the previous step, i.e. the Augmented Data.

```
#setting path to our dataset folder
dirName = '/content/drive/MyDrive/project/augmented data'
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
```

OUTPUT

```
['/content/drive/MyDrive/project/augmented data/Bird/GIB_AUG',  
 '/content/drive/MyDrive/project/augmented data/Bird/SPS_AUG',  
 '/content/drive/MyDrive/project/augmented data/Flowers/Corpse_AUG',  
 '/content/drive/MyDrive/project/augmented data/Flowers/LS_Orchid_AUG',  
 '/content/drive/MyDrive/project/augmented data/Mammal/SW_Deer_AUG',  
 '/content/drive/MyDrive/project/augmented data/Mammal/Pangolin_AUG']
```

7.4) Loading Images Into Machine Understandable Data.

Load the images from each folder and add them to data which we will use for our model training.

```

#Loading the data and pre processing it to make it in trainable format #
#x data will includes the data generated for each image
#Y data will include a id no, unique for every different species, so are having 6 classes
#there for we will get 6 ids = [0,1,2,3,4,5)
#That will be tha label we're classifying.
X_data = []
Y_data = []
id_no=0
#to make a list of tuples, where we'll store the info about the image, category and species
found = []
#itering in all folders under Augmented data folder
for paths in subfolders:
    #setting folder path for each unique class and category
    files = glob.glob(paths + "/*.jpg")
    #adding tuples to the list that contain folder name and subfolder name
    found.append((paths.split('/')[ -2],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

```


7.5) Data Splitting Into Train And Test

First thing is to convert the data into Numpy arrays so that we can feed it to the model Then, convert the Y data to categorical using keras inbuilt function which works almost like One Hot Encoding. After that, Split the data into Test and Train.

```
id_no+=1
print(X_data)
print(Y_data)

X=np.array(X_data)
Y=np.array(Y_data)
print("x-shape",X.shape,"y shape",Y.shape)
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)
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")
```

OUTPUT

```
[[0.16862746 0.16862746 0.16470589]
 [0.16862746 0.16862746 0.16470589]
 [0.16470589 0.16470589 0.16078432]
 ...
 [0.25882354 0.2784314 0.32156864]
 [0.22745098 0.24313726 0.28627452]
 [0.20784314 0.22352941 0.25882354]]

[[0.16862746 0.16470589 0.15686275]
 [0.16862746 0.16470589 0.15686275]
 [0.16862746 0.16470589 0.15686275]
 ...
 [0.27058825 0.28627452 0.3372549 ]
 [0.23529412 0.2509804 0.29803923]
 [0.21568628 0.23137255 0.26666668]]] y_cat shape [[1. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0.]
 ...
 [0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 1.]]
X shape (1251, 224, 224, 3) y_cat shape (1251, 6)
The model has 1000 inputs
```

7.6) Getting Started With Convolutional Neural Networks

Neural Networks

Neural networks are integral for teaching computers to think and learn by classifying information, similar to how we as humans learn. With neural networks, the software can learn to recognize images, for example. Machines can also make predictions and decisions with a high level of accuracy based on data inputs.

Deep Learning

Deep learning is at the cutting-edge of intelligent automation. It focuses on machine learning tools and deploying them to solve problems by making decisions. With deep learning, data is processed through neural networks, getting closer to how we think as humans. Deep learning can be applied to images, text, and speech to draw conclusions that mimic human decision making.

Start (Sequential)

Convolutional Neural Network(CNN) is a type of advanced artificial neural network. A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers and normalization layers. Before diving into the Convolution Neural Network, let us first revisit some concepts of Neural Network. In a regular Neural Network there are three types of layers:

Input Layers: It's the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in case of an image).

Hidden Layer: The input from the Input layer is then fed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layer can have

different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.

Output Layer: The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into a probability score of each class.

The data is then fed into the model and output from each layer is obtained; this step is called feedforward. Then calculate the error using an error function, some common error functions are cross-entropy, square loss error etc

```
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]
```

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

Strides decide how our weight matrix should move in the input, i.e jumping one step or two.

Padding amount of pixels added to an image when it is being processed by the kernel of a CNN

MaxPooling selects the maximum element from the region of the feature map covered by the filter. Thus, the output after the max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

DropoutLayer : Dropout works by probabilistically removing, or “dropping out,” inputs to a layer, which may be input variables in the data sample or activations from a previous

layer. It has the effect of simulating a large number of networks with very different network structure and, in turn, making nodes in the network generally more robust to the inputs.

Dropout refers to dropping (not considering in both forward and backward pass) some neurons during the training phase. The neurons which are chosen at random.

FullyConnected essential component of Convolutional Neural Networks (CNNs), which have been proven very successful in recognizing and classifying images for computer vision. layers where all the inputs from one layer are connected to every activation unit of the next layer. Fully connected layer that interprets the features extracted by the convolutional part of the model has to **Flatten** and connected to the output layer, that's what fully connected does, it flattens. It is also called the dense layer.

Flatten layer is used between the convolutional layers and the dense layer to reduce the feature maps to a single one-dimensional vector.

#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
```


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

In this example, we define a convolutional layer with filter maps and kernels. This is followed by a max pooling layer and a dense layer to interpret the input feature. An output layer is specified that predicts a single numerical value .The model is fit using the efficient [Adam version of stochastic gradient descent](#) and optimized using the mean squared error, or ‘mse’, loss function. Once the model is defined, we can fit it on the training dataset .The model expects the input shape to be three-dimensional with [samples, timesteps, features], therefore, we must reshape the single input sample before making the prediction.

Model Summary: model.summary() is used to see all parameters and shapes in each layer in our model.

```
#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
```

OUTPUT

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 (MaxPooling2D)	(None, 54, 54, 128)	0
conv2d_2 (Conv2D)	(None, 52, 52, 64)	73792
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 64)	0
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		

```
Epoch 1/10
63/63 [=====] - 473s 7s/step - loss: 1.8248 - accuracy: 0.1670 - val_loss: 1.8031 - val_accuracy: 0.1673
Epoch 2/10
63/63 [=====] - 475s 8s/step - loss: 1.7786 - accuracy: 0.2250 - val_loss: 1.5821 - val_accuracy: 0.4542
Epoch 3/10
63/63 [=====] - 479s 8s/step - loss: 1.4294 - accuracy: 0.4230 - val_loss: 1.1377 - val_accuracy: 0.5697
Epoch 4/10
63/63 [=====] - 478s 8s/step - loss: 1.1969 - accuracy: 0.5030 - val_loss: 1.2102 - val_accuracy: 0.5817
Epoch 5/10
63/63 [=====] - 474s 8s/step - loss: 1.0244 - accuracy: 0.6080 - val_loss: 1.0201 - val_accuracy: 0.5896
Epoch 6/10
63/63 [=====] - 481s 8s/step - loss: 0.9038 - accuracy: 0.6560 - val_loss: 1.1581 - val_accuracy: 0.5737
Epoch 7/10
63/63 [=====] - 482s 8s/step - loss: 0.7572 - accuracy: 0.7180 - val_loss: 0.7804 - val_accuracy: 0.7052
Epoch 8/10
63/63 [=====] - 496s 8s/step - loss: 0.5644 - accuracy: 0.7990 - val_loss: 0.9069 - val_accuracy: 0.6853
Epoch 9/10
63/63 [=====] - 489s 8s/step - loss: 0.3442 - accuracy: 0.8840 - val_loss: 1.0490 - val_accuracy: 0.7012
Epoch 10/10
63/63 [=====] - 476s 8s/step - loss: 0.2762 - accuracy: 0.9050 - val_loss: 0.6915 - val_accuracy: 0.7968
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 (MaxPooling2D)	(None, 54, 54, 128)	0
conv2d_2 (Conv2D)	(None, 52, 52, 64)	73792
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 64)	0
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		

7.9) Evaluation And Model Saving

In this part of our project, we will evaluate the model and save it. Evaluation (Accuracy And Losses)

Accuracy, Loss : Loss value implies how poorly or well a model behaves after each iteration of optimization. An accuracy metric is used to measure the algorithm's performance in an interpretable way. The accuracy of a model is usually determined after the model parameters and is calculated in the form of a percentage.

```
#printing the keys we have for the stores values
print(h.history.keys())
#appendind 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 represetns differnt 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)
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
histories_acc [[0.167      0.22499999 0.42300001 0.50300002 0.60799998 0.65600002
 0.71799999 0.79900002 0.884      0.90499997]] histories_loss [[1.8247869  1.77863705 1.4294194  1.19691491 1.02443624 0.90380472
 0.75716466 0.56439137 0.3441923  0.27620065]] histories_val_acc [[0.16733068 0.45418328 0.5697211  0.58167332 0.58964145 0.5737052
 0.70517927 0.68525898 0.70119524 0.79681277]] histories_val_loss [[1.80307472 1.58214223 1.13767743 1.21019447 1.02011144 1.15805399
 0.78037375 0.90693963 1.04902613 0.69148052]]
```

7.10) Making a Test Prediction :

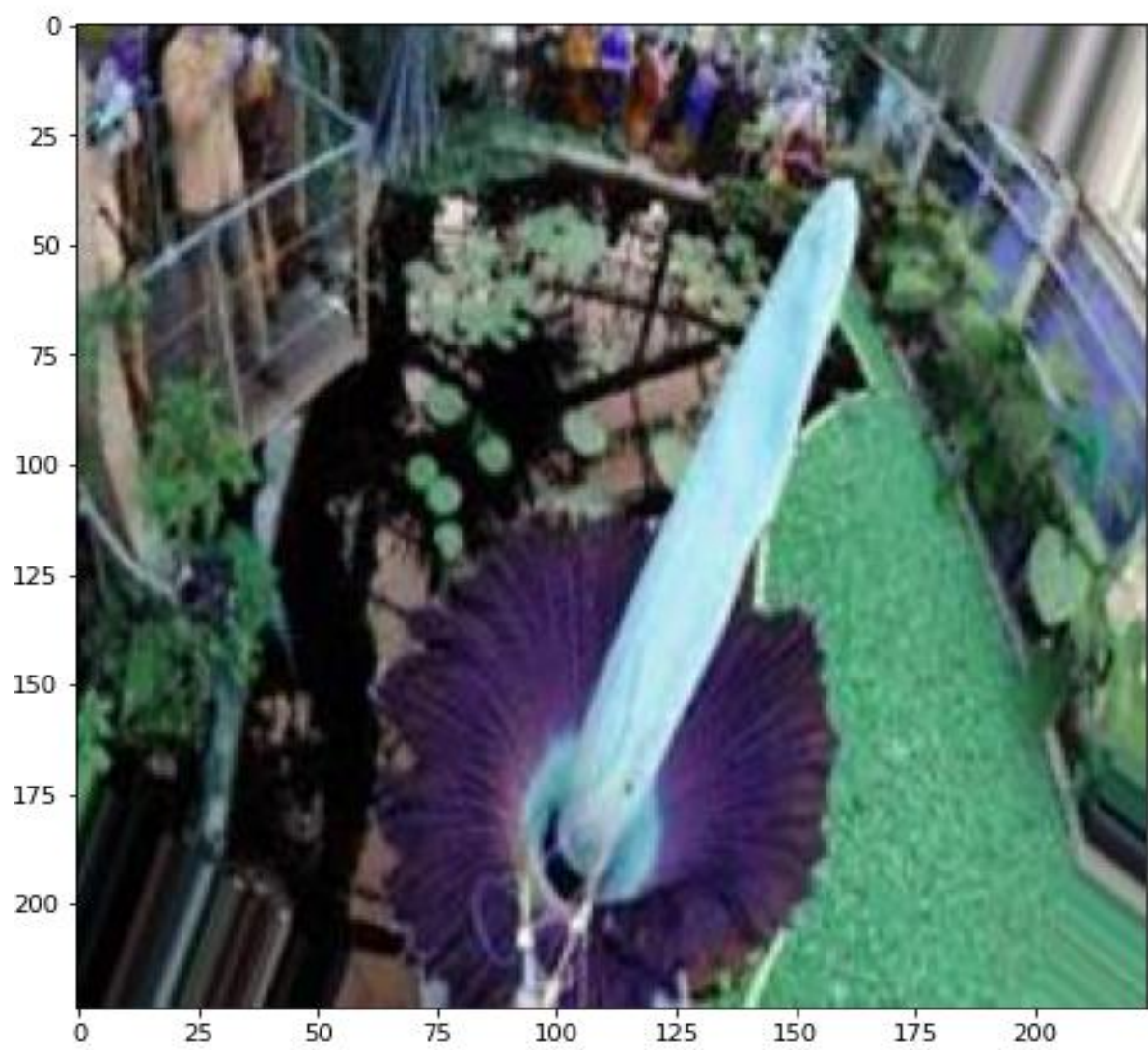
```
#Predicting the image's classes
#individual scores for each class as well as class with the highest score is printed
#making predictions ,storing result as array of probabilities of each class predicted
import random
image_number = np. random.randint(0, len(X_test))
predictions =model.predict([X_test[image_number].reshape(1, 224, 224,3)])
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].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
#printing the final output
print(found[ClassIndex[0]])
```

OUTPUT

```
1/1 [=====] - 0s 178ms/step
Label: 0, Type : Bird, Species : GIB_AUG, Score: 1.283%
Label: 1, Type : Bird, Species : SPS_AUG, Score: 1.308%
Label: 2, Type : Flowers, Species : Corpse_AUG, Score: 67.831%
Label: 3, Type : Flowers, Species : LS_Orchid_AUG, Score: 4.775%
Label: 4, Type : Mammal, Species : SW_Deer_AUG, Score: 0.505%
Label: 5, Type : Mammal, Species : Pangolin_AUG, Score: 24.298%
1/1 [=====] - 0s 173ms/step
('Flowers', 'Corpse_AUG')
```

7.11) Loading Test data

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

7.12) Model Saving And Loading

Saving The Model (get_weights, set_weights) : Given that deep learning models can take hours, days and even weeks to train, it is important to know how to save and load them from the disk .We will look into how to save your Keras models to file and load them up again to make predictions.

```
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")
```

8 TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform

basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items Valid Input : identified classes of valid input must be accepted. Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised. Systems/Procedures: interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements,

key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined. **System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phase test phase of the software lifecycle, although it is not uncommon for coding and unittesting to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually, and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

9 OUTPUT

Digital Naturalist

Bird, Flower, Mammal Recognition

Please upload an image

Choose...



Result: Lady's slipper orchids are usually terrestrial, though some are epiphytic or grow on rocks. Most species have rhizomes and fibrous roots. Unlike most other orchids, the flowers characteristically feature two fertile anthers (male, pollen-producing structures) instead of just one. The slipper-shaped lip of the flower serves as a trap for pollinating insects, forcing insect visitors to climb past the reproductive structures and deposit or receive pollinia (pollen masses) to fertilize the flower



Bird, Flower, Mammal Recognition



Please upload an image

Choose...



Result: The Seneca white deer are a rare herd of deer living within the confines of the former Seneca Army Depot in Seneca County, New York. When the 10,600-acre (43 km²) depot was created in 1941, a 24-mile (39 km) fence was erected around its perimeter, isolating a small herd of white-tailed deer, some of which had white coats

11 CONCLUSION

Field naturalists can only use this web app from anywhere to identify the birds, flowers, mammals and other species they see on their hikes, canoe trips and other excursions. In this project, we are creating a web application which uses a deep learning model, trained on different species of birds, flowers and mammals. There is great diversity among naturalists, but some common ground too. All naturalism begins with an admiring attitude towards science and its achievements. In many cases this admiring attitude is combined with a contempt or distrust for the way that philosophy has been or is conducted. This combination of views has a long history. Many of the advocates of first philosophy, Descartes, Kant and Carnap, shared the same admiration of science or nascent science and distrust of philosophy. Descartes, for example, uses scepticism as a device to sweep away the old Aristotelian foundations of knowledge, so that he can build an entirely new philosophy that makes room for the new mathematical sciences.

12 FUTURE SCOPE

Essentially, the proposed guidelines treat statistical comparison of ML based quality estimators as a multi-dimensional problem. Accordingly, we seek to assess the predictors more holistically in terms of their local performance on specific test conditions, their learning ability and the magnitude of treatment effect (to quantify the practical significance of the observed differences). In contrast, the current approach tends to reduce this task to binary and global statistical decision making and does not reveal systematic weaknesses (or strengths) of the predictors. In order to provide a tool for practical use, software implementing the proposed guidelines is made publicly available.

13 APPENDIX

Source Code

i)Data Collection

```
from keras.preprocessing.image import ImageDataGenerator
import cv2
import csv
from os import listdir
import time
from google.colab import drive
drive.mount('/content/drive')
# Nicely formatted time string to make a note of how much time it takes for augmentation
def hms_string (sec_elapsed):
    h = int(sec_elapsed / (60 * 60))
    m = int((sec_elapsed % (60 * 60)) / 60)
    s = sec_elapsed % 60
    return "{n}: {m}:{round(s,1)}"

def augment_data(file_dir, n_generated_samples, save_to_dir):
    """
    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.
    """
    #from keras.preprocessing.image import ImageDataGenerator
    #from os import listdir
    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
```

ii) Loading data and Preprocessing

```
#For matrix calculations and data Managememnt
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, array_to_img, img_to_array, 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.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 glob
from PIL import Image
import os
from os import listdir
```

iii) Started With Convolutional Neural Networks.

```
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]
```

```
#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 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
```


iv) Evaluation And Model Saving

```
#printing the keys we have for the stores values
print(h.history.keys())
#appendind 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 represetsn differnt 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)
```

```
#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
import random
image_number = np. random.randint(0, len(X_test))
predictions =model.predict([X_test[image_number].reshape(1, 224, 224,3)])
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].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
#printing the final output
print(found[ClassIndex[0]])
```



```
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")
```

13 GitHub & Project Demo Link

Github -- <https://github.com/IBM-EPBL/IBM-Project-17546-1659673195>

Project Demo Link-- <https://www.youtube.com/watch?v=K4aR19FmZC0>