Sri Sivasubramaniya Nadar College of Engineering
(An Autonomous Institution, Affiliated to Anna University, Chennai)
Rajiv Gandhi Salai (OMR), Kalavakkam – 603110

Professional Readiness for Innovation, Employability & Entrepreneurship

PROJECT REPORT - 2022

DIGITAL NATURALIST -AI ENABLED TOOL FOR BIODIVERSITY RESEARCHERS

SUBMITTED BY :

VINU ABINAYAA R (193002120)
RAKSHAA MADHURI K (193002077)
VIJAYARAGHAVAN P (193002117)
VARSHINI G (193002114)
SHWATHI RAMANATHAN (193002097)

in partial fulfillment for the award of the degree
of
BACHELOR OF ENGINEERING
IN
ELECTRONICS AND COMMUNICATION ENGINEERING

TABLE OF CONTENTS

CHAPTER-1
INTRODUCTION

1.1 Project overview:

A naturalist is someone who observes natural patterns and recognises various flora and fauna in the wild. Field naturalists frequently rely on conventional strategies when exploring the woods, such as always bringing a guidebook with them or asking knowledgeable ornithologists for assistance. They ought to have access to a useful tool that allows them to notice, catalog, and communicate the beauty to others. To identify the birds, flowers, animals, and other species they see on their treks, canoe journeys, and other excursions, field naturalists can utilize this web app from anywhere.

In this project we will be collecting images and information about different flora and fauna and training a deep learning model to predict the species for the given image using a web application.

1.2 Purpose:

The purpose of this project is to create a web application which consists of information about different flora and fauna which will serve as a guide for digital naturalists on their hikes. This can be used by many naturalists, archaeologists, ecologists, biologists, other scientists and curious laymen to reduce the time of identification of the species and progress with their research or work. Society of Scientists, researchers, knowledge seekers or students can attract better. This will help ease the work of digital naturalists.

CHAPTER-2
LITERATURE SURVEY:

2.1 Existing problem:

A naturalist is someone who studies the patterns of nature, identifies a different kind of flora and fauna in nature. 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.

2.2 References

- Adapted approach for Species Classification,Schmid Huber, J. (2015)
- Detection And Classification of images using Detection Line Haibing Wu and Xiaodong Gu (2015)
- Study on Classification of Plants Images using Combined Classifier. International Journal Ghosh S, Kumar H, Nayak JS (2015)
- Deep Learning for Plant Identification in Natural Environment Computational Intelligence and Neuroscience, Yu Sun, Yuan Liu, Guan Wang, Haiyan Zhang (2017)
- Classification of Cotton Leaf Diseases Using AlexNet and Machine Learning Models. Current Journal of Applied Science and Technology Borugadda, P., Lakshmi, R., & Govindu, S. (2021).

TITLE	AUTHOR	YEAR OF PUBLICATION	ADVANTAGES	DISADVANTAGES
Adapted approach for Species Classification	Schmid Huber, J.	2015	K-Means clustering is used for image segmentation and images are classified under one of the classes by using multi-class support vector machine. The classification accuracy is achieved up to 89%.	K means clustering cannot be implemented for a higher degree of classes.
Detection	Haibing Wu	2015	Images were segmented	Complexity is a very

TITLE	AUTHOR	YEAR OF PUBLICATION	ADVANTAGES	DISADVANTAGES
And Classification of images using Detection Line	and Xiaodong Gu		by a detection line (DL) method. Six geometric features and 3 color features methodology were presented classifying the image to an accuracy of 90.9%.	high line detection line method.
Study on Classification of Plants Images using Combined Classifier. International Journal	Ghosh S, Kumar H, Nayak JS	2015	The nearest-neighbor method is the simplest of all algorithms for predicting the class of a test example. VM was found competitive with the best available machine learning algorithms in classifying high-dimensional data sets. Genetic algorithms are good at refining irrelevant and noisy features selected for classification.	Disadvantage of the k-NN method is the time complexity of making predictions,Classifying using PNN and SVM is not explored.
Deep Learning for Plant Identification in Natural Environment Computational Intelligence and Neuroscience ,	Yu Sun, Yuan Liu, Guan Wang, Haiyan Zhang	2017	Using SGD optimization, the proposed ResNet26 model results in 91.78% accuracy.e ResNet26 model on the public Flavia dataset with 1907 images achieved an accuracy of 99.65%.	Deep learning model used does not focus on prediction, insect detection, disease segmentation. The BJFU100 database does not give information on different phases of life cycles of species, instead focusing only on one phase.
Classification of Cotton Leaf Diseases Using	Borugadda, P., Lakshmi, R., &	2021	For a given imbalanced dataset, the optimal deployment model is chosen based on the	Computational complexity is high due to more no. of convolutional layers

TITLE	AUTHOR	YEAR OF PUBLICATION	ADVANTAGES	DISADVANTAGES
AlexNet and Machine Learning Models. Current Journal of Applied Science and Technology	Govindu, S.		macro F1_score. AlexNet model gives the best result among all classification models with an F1_score of 94.92%.	and connected layers along with different classes thereby application of ML algorithms for classification is high.

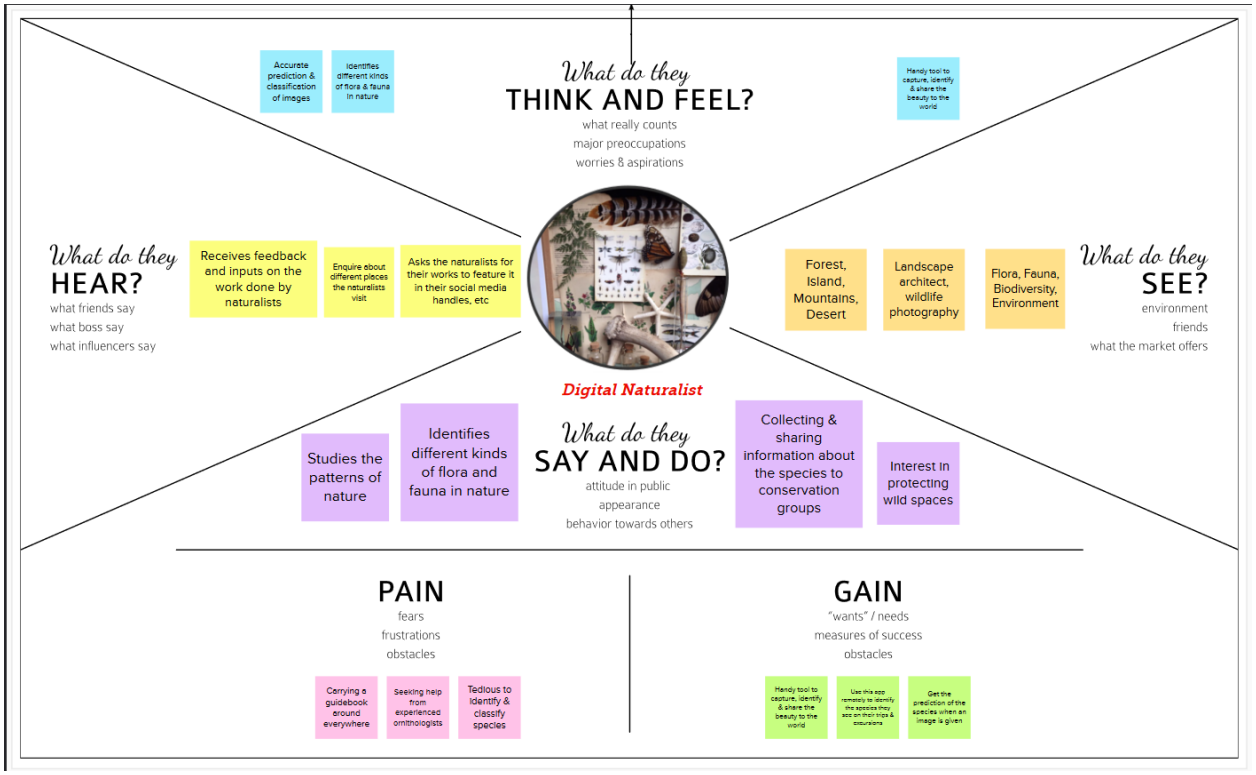
2.3 Problem Statement Definition

Botanist who Focuses on flora and their nature with respect to different environments, Zoologist who focuses on fauna and their behavior with respect to different scenarios and Naturalist who focuses on different life forms and their surroundings are subjected to classification, research, identification and eradication problems due to the unknown and large variety of species, making it difficult for them to process the information fastly and efficiently at anywhere and anytime.

CHAPTER-3
IDEATION & PROPOSED SOLUTIONS

3.1 EMPATHY MAP CANVAS

An empathy map is a widely-used visualization tool used to build a shared understanding of the user's needs and provide context to a user-centered solution. Empathy map occurs in the early research process. It is used to get into the user's head and heart. Empathy map helps to picture user's situation as well as to consider what future research might be needed.

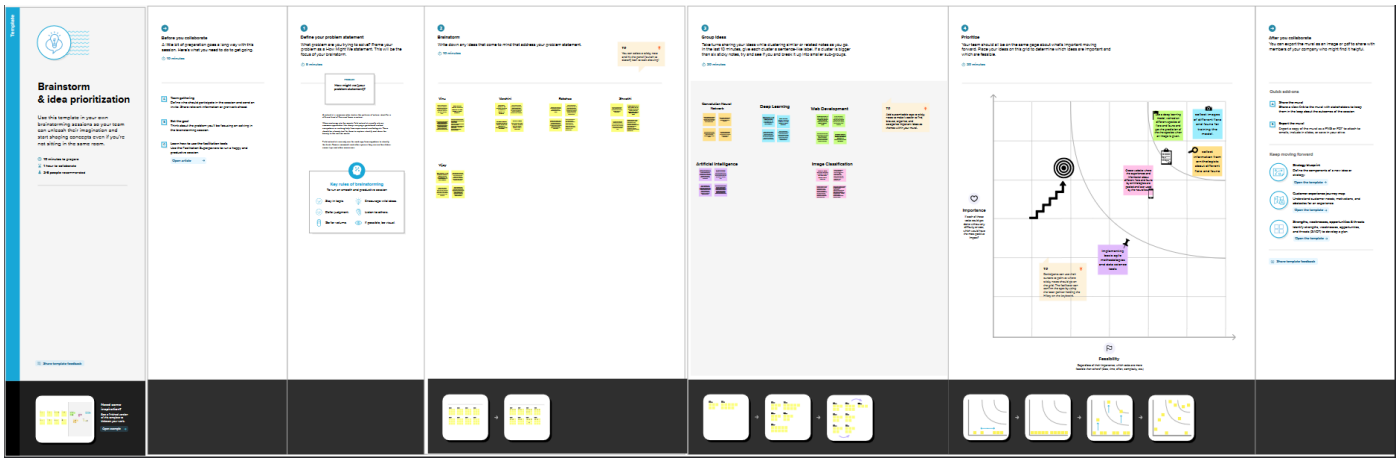


3.2 IDEATION AND BRAINSTORMING :

Brainstorming is a method of generating ideas and sharing knowledge to solve a particular commercial or technical problem, in which participants are encouraged to think without interruption. Brainstorming is a group activity where each participant shares their ideas as soon as they come to mind. At the conclusion of the session, ideas are categorized and ranked for follow-on action.

When planning a brainstorming session it is important to define clearly the topic to be addressed. A topic which is too specific can constrict thinking, while an ill-defined topic will not generate enough directly applicable ideas. The composition of the brainstorming group is important too. It

should include people linked directly with the subject as well as those who can contribute novel and unexpected ideas. It can comprise staff from inside or outside the organization.

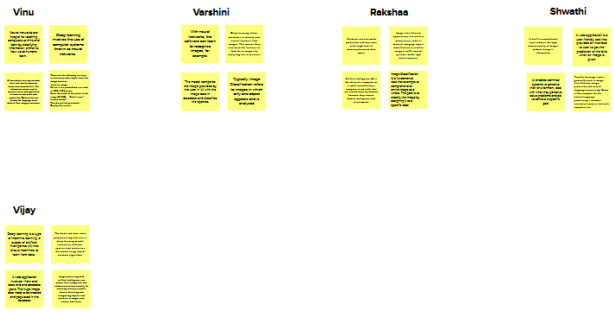


3.2.1 Idea Prioritization

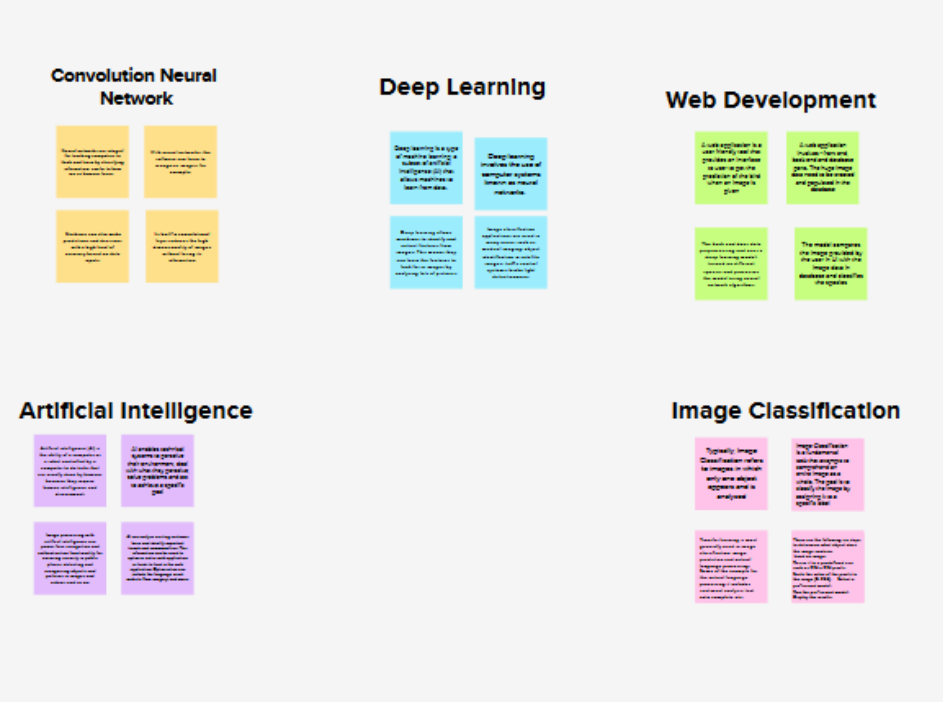
3.2.1.1 Defining Problem Statement

A naturalist is someone who studies the patterns of nature, identifies a different kind of flora and fauna in nature.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.

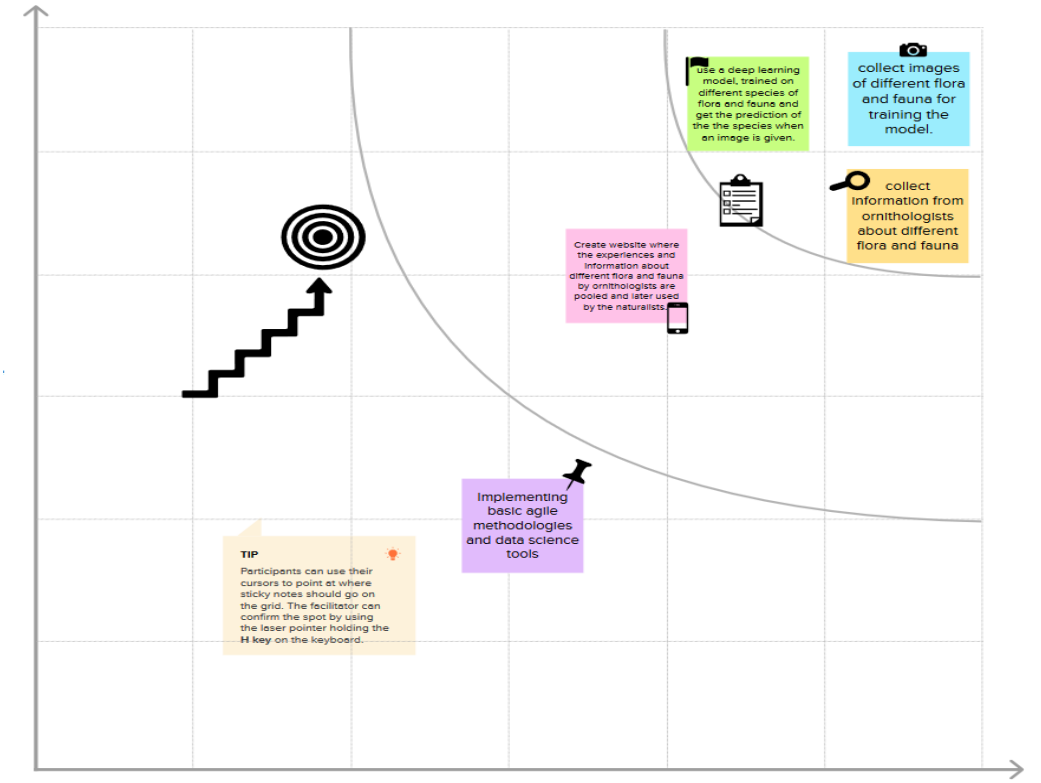
3.2.1.2 Brain Storm :



3.2.1.3 Group Ideas:



3.2.1.4 Prioritize:



Plot of Importance (vs) Feasibility

3.3 PROPOSED SOLUTION

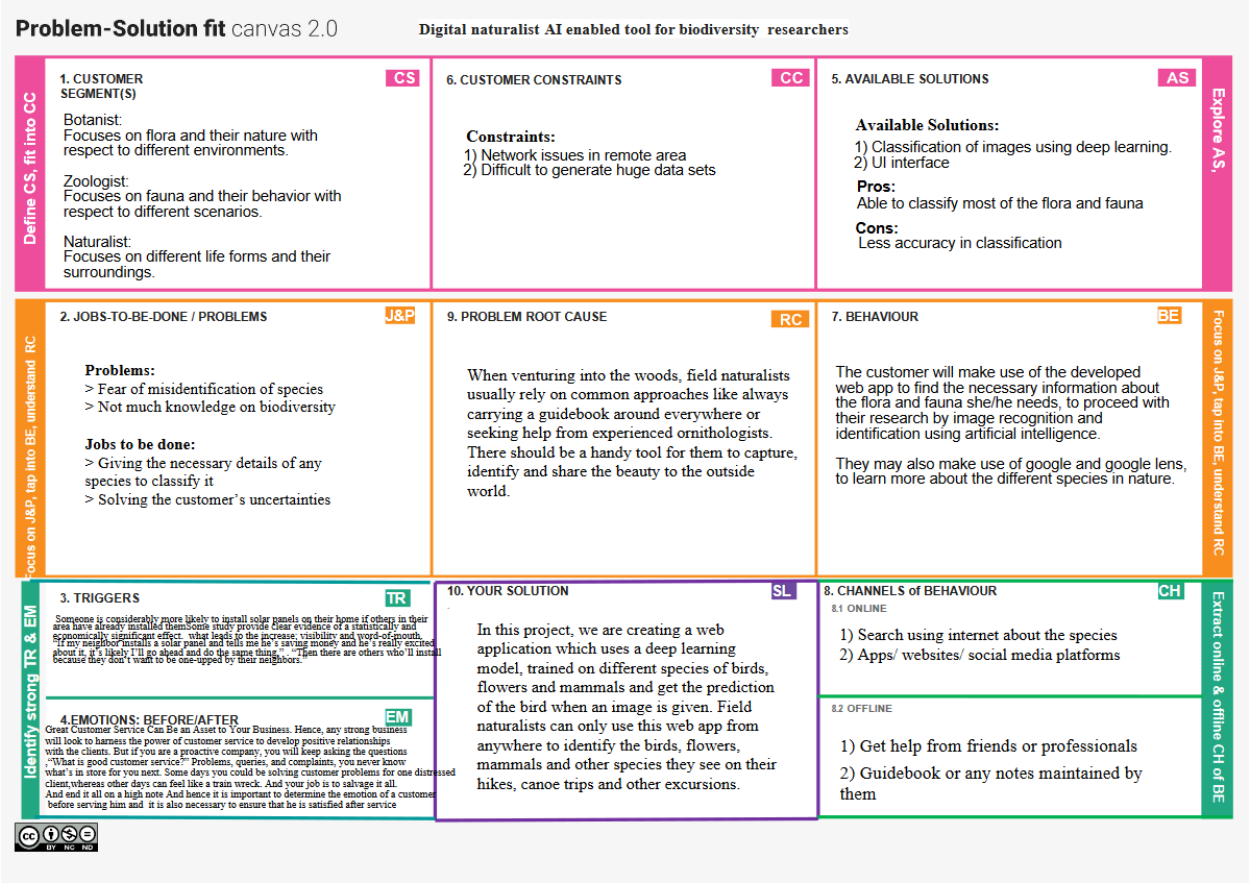
S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	A naturalist is someone who studies the patterns of nature, identifies a different kind of flora and fauna in nature. When venturing into the woods, field naturalists usually rely on common approaches like always carrying a guidebook 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.
2.	Idea / Solution description	To collect images and information from ornithologists and other sources about different flora and fauna to train a deep learning model to predict the species for the given image using a web application.
3.	Novelty / Uniqueness	create a web application which consists of information about different flora and fauna which will serve as a guide for digital naturalists on their hikes.
4.	Social Impact / Customer Satisfaction	The impact that naturalism has left on society is colossal, leading to the evolution of the modern movement. Generally, **naturalistic works expose dark sides of life such as prejudice, racism, poverty, prostitution, filth, and disease** which may not be of public interest leading to various difference in opinion and in turn conflicts . But apart from all these the happiness that they get when a customer is satisfied because of their ideologies is unmatchable .

		A few ways to improve customer satisfactions in major field works are : ● How predictive analytics can reshape customer experience in field services ● Personalization for customers ● Equipping co-naturalists with better knowledge ● Work Assignment ● Anticipate Service Requests ● Optimize Business Operations
5.	Business Model (Revenue Model)	Field Naturalists and conservation groups will be benefited which can generate revenue. Revenue can also come from affiliate marketing. Advertisements can also add to the revenue of the product.
6.	Scalability of the Solution	This can be used by any naturalists, archaeologists, ecologists, biologists, other scientists and curious laymen to reduce the time of identification of the species to progress with their research or work. Society of Scientists, researchers, knowledge seekers or students can attract better. Thus, it is very much scalable and efficient for a long period of time, as the database can be regularly updated and can be used anywhere, anytime by the users.

3.4 PROBLEM - SOLUTION FIT

What is a Problem-Solution Fit?

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem.. There are 10 easy steps to obtain your project - solution fit which is shown below



CHAPTER-4
REQUIREMENT ANALYSIS

4.1 Functional Requirement:

Functional requirements are features / functions that a developer must implement in order to enable users to accomplish their tasks.

The following are the functional requirements of the proposed solution:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Gmail Registration through form
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Interface	A web page for user interaction
FR-4	AI Model	A model to classify the images
FR-5	User Input	User provides image input through camera
FR-6	Output	Results shown through prompt
FR-7	Discussion Forum	Discussion and comments of other users are displayed
FR-8	User History	Provides history of user classified species

4.2 Non-Functional Requirements:

Non-Functional Requirements basically define system attributes such as scalability, security, reliability, etc.

The following are the non functional requirements of the proposed solution:

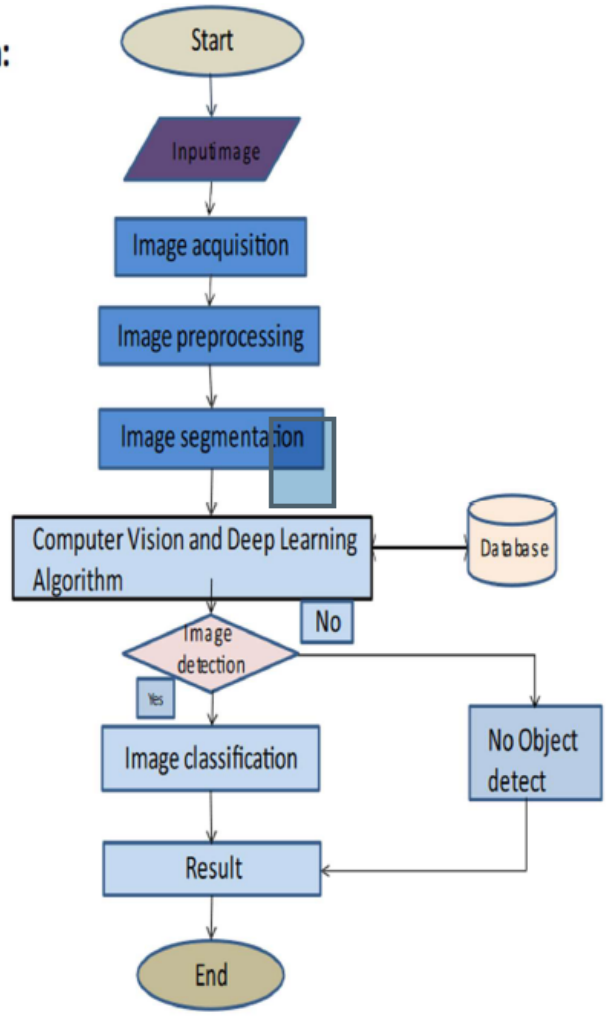
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Easy to use and provides accurate results
NFR-2	**Security**	Secure user information
NFR-3	**Reliability**	User gets the name of the species accurately
NFR-4	**Performance**	Provides faster and accurate results
NFR-5	**Availability**	Can be used @4/7
NFR-6	**Scalability**	Can classify large number of species

CHAPTER-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 depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

Data Flow Diagram:



5.2 SOLUTION & TECHNICAL ARCHITECTURE

5.2.1 SOLUTION ARCHITECTURE

A solution architecture (SA) is an architectural description of a specific solution. SAs combine guidance from different enterprise architecture viewpoints (business, information and technical), as well as from the enterprise solution architecture (ESA)

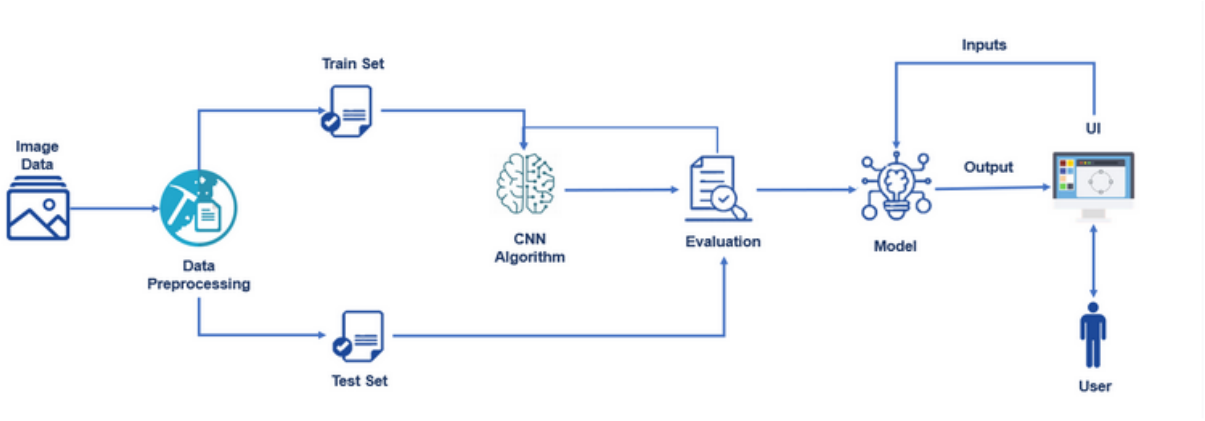


Fig : Solution Architecture Diagram

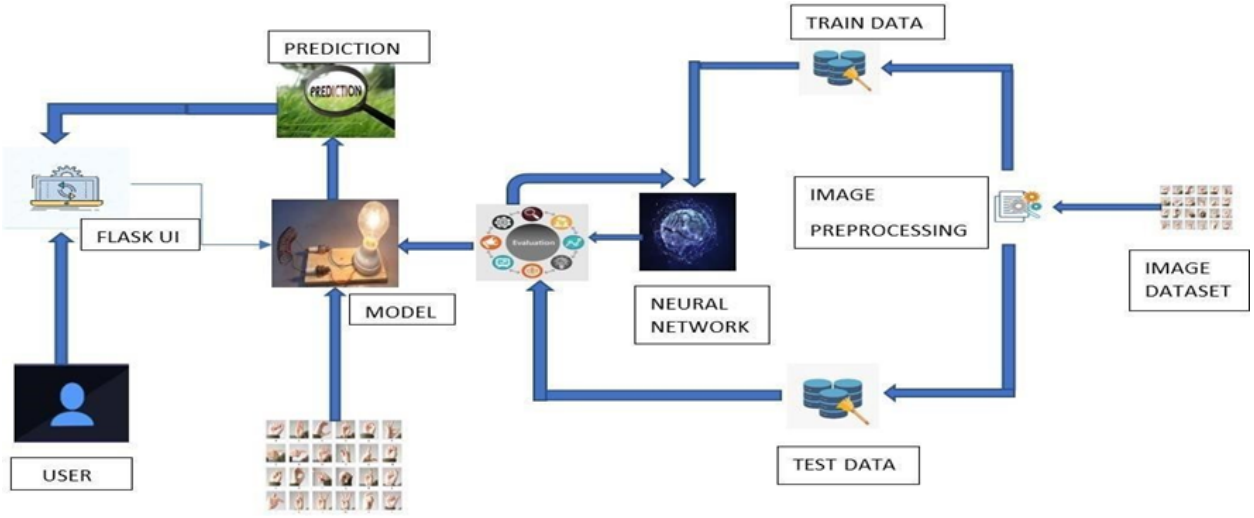
Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions.
Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed,and delivered.

Our Solution Architecture Describes the following :

Images of different flora and fauna are collected and preprocessed. This data set is then segregated into a training set and a test set. The training set is used to train a model using CNN algorithm. The test set is then used to evaluate the trained model to determine its accuracy. The evaluated model is integrated with UI. When the UI sends input (images in this case) to the model, it predicts the type of species in the image and sends the output back to the UI.

5.2.2 TECHNOLOGICAL ARCHITECTURE



5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer/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

CHAPTER-6
PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Modeling Phase	USN-2	Data Collecting and digitalizing for analyzing	3	Medium	Vinu
Sprint-1		USN-3	Building a CNN model using the collected data	5	High	Vijay
Sprint-1		USN-3	Evaluating the model to check the accuracy and precision	3	High	Varshini
Sprint-2	Development Phase	USN-1	Home page Creation – Shows the features of our application	4	Medium	Shwathi
Sprint-2		USN-1	Setting up facilities for registering and logging into the website	4	Medium	Vinu
Sprint-2		USN-2	Setting up facilities for users to upload image	4	Medium	Rakshaa
Sprint-2		USN-2	Creating prediction page which shows the result	5	High	Rakshaa

Sprint-3	Deployment Phase	USN-3	Integrating UI & backend using API	3	Medium	Varshini
Sprint-4	Testing Phase	USN-3	Functional testing – testing the functional requirements	5	High	Shwathi
Sprint-4		USN-3	Non-Functional testing – Checking non-functional requirements such as scalability, robustness, etc.	5	High	Vijay

6.2 SPRINT DELIVERY SCHEDULE

Project Tracker & Velocity:

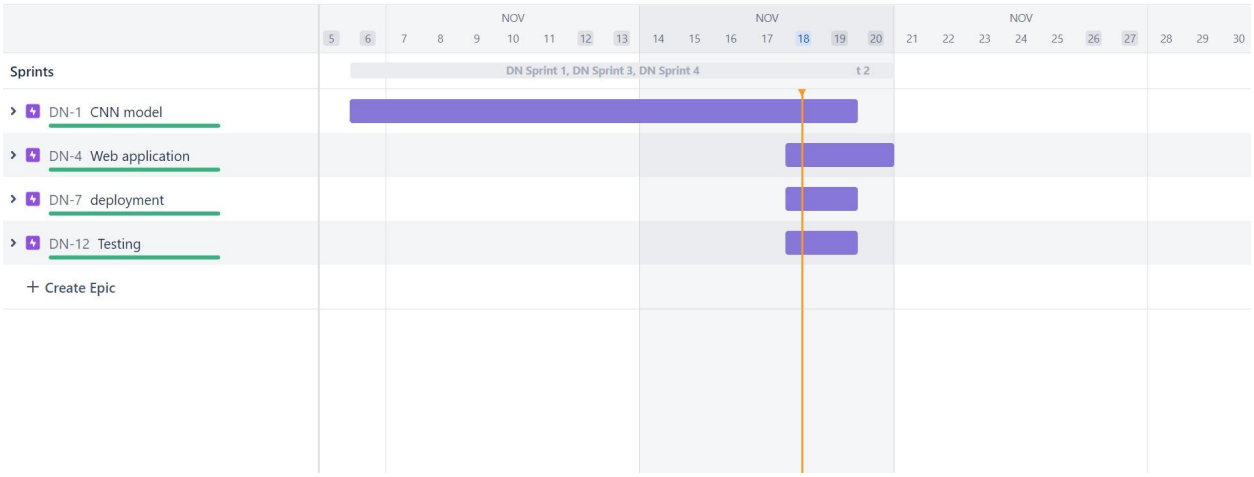
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	8	6 Days	24 Oct 2022	29 Oct 2022	8	29 Oct 2022
Sprint-2	12	6 Days	31 Oct 2022	05 Nov 2022	12	05 Nov 2022
Sprint-3	21	6 Days	07 Nov 2022	12 Nov 2022	21	12 Nov 2022

Velocity:

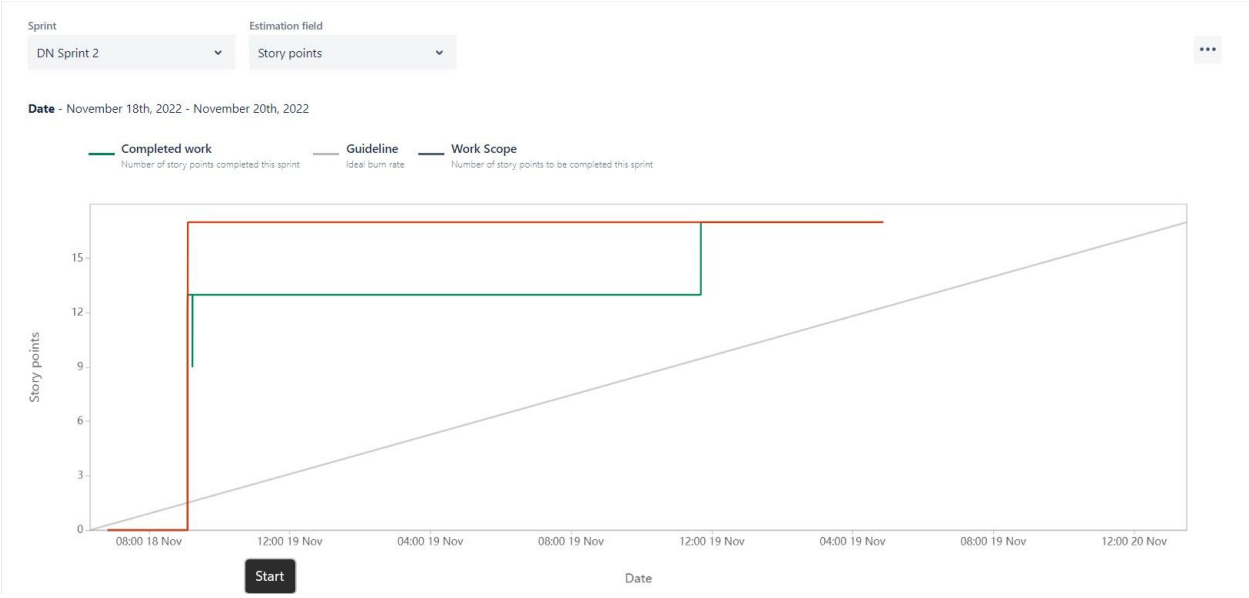
Average Velocity = 41/18 = 2.27

6.3) REPORTS FROM JIRA :

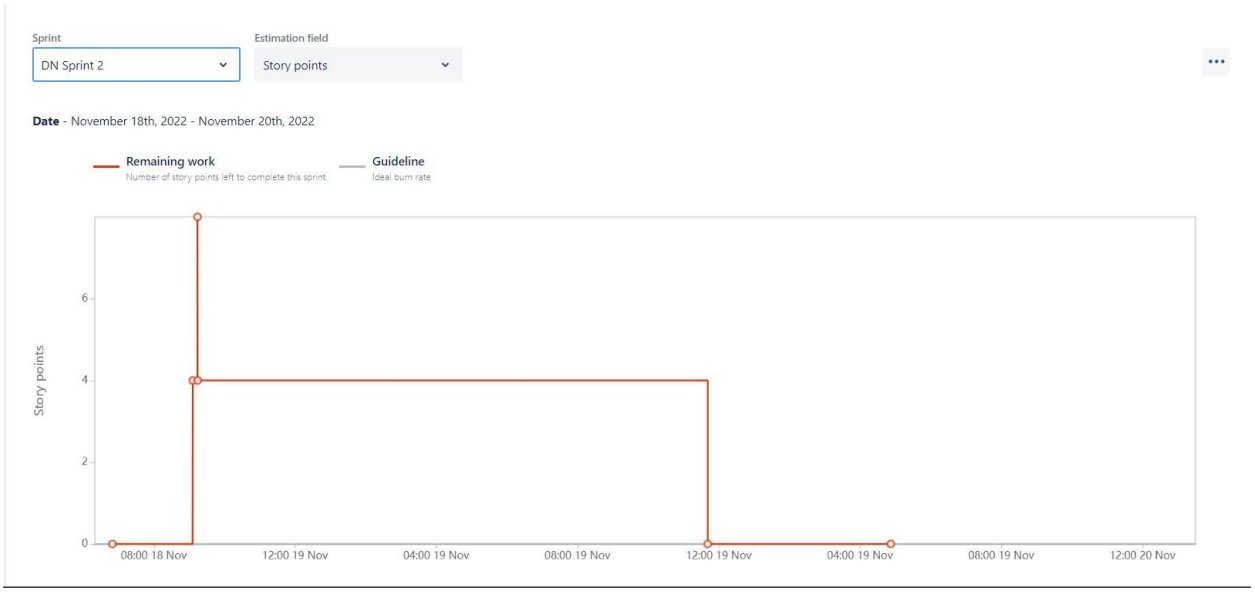
Roadmap:



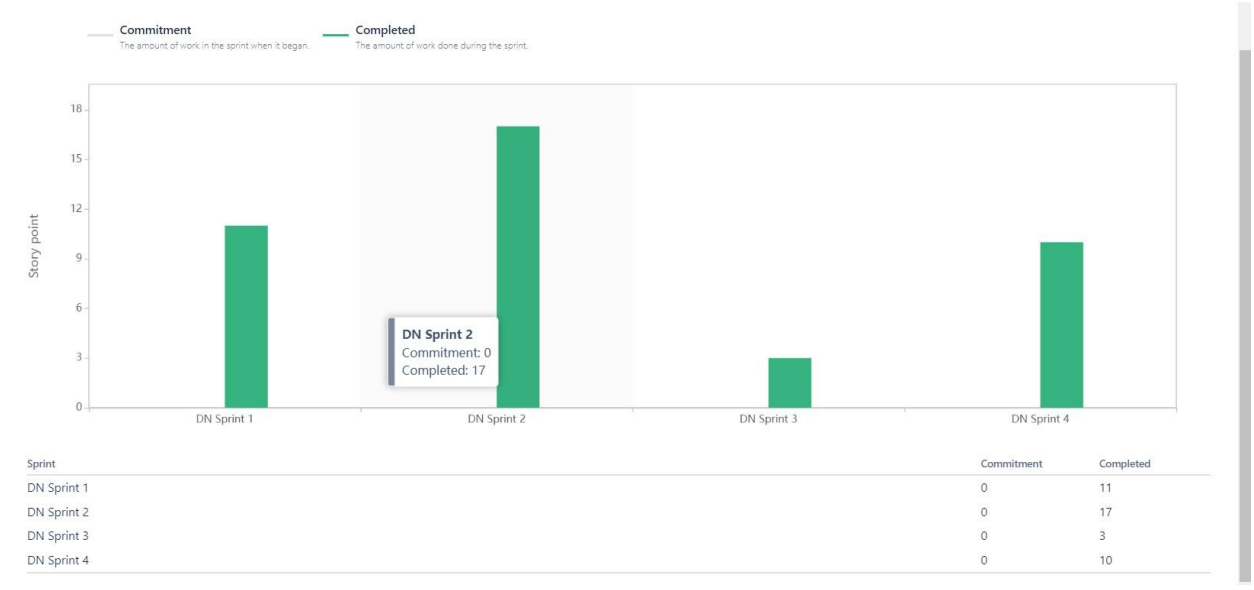
Burnup Chart:



Sprint Burndown Chart:



Velocity Report:



Sprint	Commitment	Completed
DN Sprint 1	0	11
DN Sprint 2	0	17
DN Sprint 3	0	3
DN Sprint 4	0	10

CHAPTER-7
CODING & SOLUTIONING

7.1 Feature -1 Augmenting The Data

Import the required libraries and define an augmentation function using keras inbuilt image pre-processing package which is Image Data Generator. Define parameters that are needed to be used while augmenting the data. Inputs to this function are defined in the comment.


```
"cells": [
 {
  "cell_type": "code",
  "execution_count": 1,
  "id": "c292aad3",
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "Elapsed Time : 13.581131219863892\n"
    ]
   }
  ],
  "source": [
   "from keras.preprocessing.image import ImageDataGenerator\n",
   "import cv2\n",
   "from os import listdir\n",
   "import time    \n",
   "\n",
   "def hms_string(sec_elapsed):\n",
   "    h = int(sec_elapsed / (60 * 60))\n",
   "    m = int((sec_elapsed % (60 * 60)) / 60)\n",
   "    s = sec_elapsed % 60\n",
   "    return f\"{h}:{m}:{round(s,1)}\"\n",
   "\n",
   "def augment_data(file_dir, n_generated_samples, save_to_dir):\n",
   "    data_gen = ImageDataGenerator(rotation_range=30, \n",
   "                                  width_shift_range=0.1,\n",
   "                                  height_shift_range=0.15, \n",
   "                                  shear_range=0.25, \n",
   "                                  zoom_range = 0.2,\n",
   "                                  horizontal_flip=True, \n",
   "                                  vertical_flip=False, \n",
   "                                  fill_mode='nearest',\n",
   "                                  brightness_range=(0.5,1.2)\n",
   "                                 )\n",
   "\n",
```


This is how to save all the images back to a given path where we're going to create our augmented data set.

```
    for filename in listdir(file_dir):\n",
        image = cv2.imread(file_dir + '/' + filename)\n",
        image = image.reshape((1,)+image.shape)\n",
        save_prefix = 'aug_' + filename[:-4]    \n",
        i=0\n",
        for batch in data_gen.flow(x=image, batch_size=1, save_to_dir=save_to_dir,\n",
                                   save_prefix=save_prefix, save_format='jpg'):\n",
            i += 1\n",
            if i > n_generated_samples:\n",
                break\n",
            \n",
```

Next step is to call the function for each of our subfolders and give all the input parameters.

```
"start_time = time.time()\n",
"\n",
"augmented_data_path = \"/Users/Vinu Abinayaa/Desktop/Augmented Dataset\"\n",
"augment_data(file_dir='/Users/Vinu Abinayaa/Desktop/Digital Naturaliist Dataset/Bird/Great Indian Bustard Bird', n_generated_samples=8, save_to_dir=augmented_data_path+'/Bird/GIB_AUG')\n",
"augment_data(file_dir='/Users/Vinu Abinayaa/Desktop/Digital Naturaliist Dataset/Flower/Corpse Flower', n_generated_samples=8, save_to_dir=augmented_data_path+'/Flower/Corpse_AUG')\n",
"augment_data(file_dir='/Users/Vinu Abinayaa/Desktop/Digital Naturaliist Dataset/Flower/Lady Slipper Orchid Flower', n_generated_samples=8, save_to_dir=augmented_data_path+'/Flower/LS_Orchid_AUG')\n",
"augment_data(file_dir='/Users/Vinu Abinayaa/Desktop/Digital Naturaliist Dataset/Mammal/Pangolin Mammal', n_generated_samples=8, save_to_dir=augmented_data_path+'/Mammal/LS_Pangolin_AUG')\n",
"augment_data(file_dir='/Users/Vinu Abinayaa/Desktop/Digital Naturaliist Dataset/Mammal/Senenca White Deer Mammal', n_generated_samples=8, save_to_dir=augmented_data_path+'/Mammal/SW_Deer_AUG')\n",
"\n",
"\n",
"end_time = time.time()\n",
"execution_time = (end_time - start_time)\n",
"print(\"Elapsed Time : \"+str(execution_time))"
]
},
```

IMPORTING LIBRARIES

```
"#For matrix calculations and data Management\n",
"import numpy as np\n",
"\n",
"\n",
"\n",
"#Importing libraries \n",
"import tensorflow as tf\n",
"\n",
"import keras \n",
"import keras.backend as k\n",
"#from keras.optimizers import SGD, Adam, Adagrad, RMSprop\n",
"\n",
"from tensorflow.keras.optimizers import Adam,Nadam, SGD\n",
"\n",
"\n",
"from keras.applications import *\n",
"\n",
"from keras.preprocessing import *\n",
"\n",
"from keras.preprocessing. image import ImageDataGenerator, array_to_img, img_to_array, load_img\n",
"\n",
"from keras.callbacks import EarlyStopping, ModelCheckpoint\n",
"\n",
"from keras.models import Sequential\n",
"\n",
"from keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Activation, BatchNormalization, Dropout\n",
"\n",
"from keras.utils.np_utils import to_categorical \n",
"\n",
"from sklearn.model_selection import train_test_split\n",
"\n",
"#For plotting charts ed for data visualizations\n",
"\n",
"import matplotlib.pyplot as plt #Libraries for Locating and loading date\n",
"\n",
"import glob \n",
"\n",
"from PIL import Image\n",
"\n",
"import os \n",
"from os import listdir"
]
},
```

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

```
    "#Setting path to our dataset folder\n",
    "dirName = \"/Users/Vinu Abinayaa/Desktop/Augmented Dataset\"\n",
    "folders = listdir(dirName)\n",
    "#Getting the names for all the folders containing data\n",
    "def getListOfFiles (dirName):\n",
    "# create a list of sub directories and files (if any) # names in the given directory\n",
    "    listOfFile = os.listdir (dirName) \n",
    "    allFiles = list()\n",
    "    for fol_name in listOfFile:\n",
    "        fullPath = os.path.join(dirName, fol_name) \n",
    "        allFiles.append(fullPath)\n",
    "    return allFiles\n",
    "Folders = getListOfFiles(dirName) \n",
    "len (Folders)\n",
    "subfolders = []\n",
    "for num in range(len(Folders)):\n",
    "    sub_fols = getListOfFiles (Folders [num]) \n",
    "    subfolders+=sub_fols\n",
    "#Now, the subfolders contains the address to all our data folders for each class \n",
    "subfolders"
   ]
  },
```

Loading Images Into Machine Understandable Data

```
    "#\\\\\\\\\n",
    "#\\\\\\\\\\\\\\\/////////\\\\\\\\\\\\\\\////////\\\\\\\\\\\\\\\\\////////\n",
    "#Loading the data and pre processing it to make it in trainable format #\\\\\\'/////\\\\\\\\\\n",
    "#X data will includes the data generated for each image\n",
    "#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]\n",
    "#That will be tha label we're classifying.\n",
    "X_data = []\n",
    "Y_data = []\n",
    "id_no=0\n",
    "#to make a list of tuples, where we'll store the info about the image, category and species\n",
    "found = []\n",
    "#itering in all folders under Augmented data folder\n",
    "for paths in subfolders:\n",
    "#setting folder path for each unique class and category\n",
    "    files= glob.glob (paths + \"/*.jpg\")\n",
    "#adding tuples to the list that contain folder name and subfolder name \n",
    "    found.append((paths.split('\\\\')[-2],paths.split('\\\\' ) [-1] ) )\n",
    "#itering all files under the folder one by one\n",
    "    for myFile in files:\n",
    "        img=Image.open(myFile)\n",
    "#img.thumbnail((width, height), Image.ANTIALIAS) # resizes image in-place keeps ratio \n",
    "        img = img.resize((224,224), Image.ANTIALIAS) # resizes image without ratio #convert the images to numpy arrays\n",
    "        img = np.array(img)\n",
    "        if img.shape == ( 224, 224, 3):\n",
    "# Add the numpy image to matrix with all data\n",
    "            X_data.append (img)\n",
    "            Y_data.append (id_no)\n",
    "    id_no+=1"
   ]
```

Data Splitting Into Train And Test

```
    "#to see our data\n",
    "print(X_data) \n",
    "print(Y_data)\n",
    "#converting lists to np arrays again\n",
    "X = np.array(X_data)\n",
    "Y = np.array(Y_data)\n",
    "# Print shapes to see if they are correct\n",
    "print(\"x-shape\", X. shape, \"y shape\", Y.shape)\n",
    "X = X.astype('float32')/255.0\n",
    "#The Keras library offers a function called to_categorical() #integer data. The sequence has an example of all known values #so we can use the to_categorical() function directly \n",
    "y_cat=to_categorical (Y_data, len(subfolders))\n",
    "print(\"X shape\",X,\"y_cat_shape\", y_cat)\n",
    "print(\"X shape\", X. shape, \"y_cat shape\", y_cat.shape)\n",
    "#\\\\\\\\\\\\\\you can use to one hot encode\n",
    "\n",
    "#\\\\\\\\\\\\\\\\\\\\\\\/////////\\\\\\ #Splitting the data to Test and Train\n",
    "\n",
    "X_train, X_test, y_train, y_test = train_test_split(X, y_cat, test_size=0.2) \n",
    "print(\"The model has \" + str(len(X_train)) + \" inputs\")"
   ]
```

7.2 Feature -2 Convolutional Neural Network

7.2.1 Getting Started With Convolutional Neural Networks

7.2.1.1 Neural Network:

Neural networks are essential for teaching computers to think and learn by classifying information in much the same way humans learn. For example, neural networks allow software to learn to recognize images. Machines can also make predictions and decisions with high accuracy based on data input.

7.2.1.2 Deep Learning:

Deep Learning is at the forefront of intelligent automation. It focuses on machine learning tools and using them to solve problems through decision making. In deep learning, data is processed by neural networks to bring us closer to human thinking. Deep learning can be applied to images, text, and speech to make inferences that mimic human decision making.

7.2.2 Introduction

A convolutional neural network (CNN) is a type of advanced artificial neural network. A CNN consists of an input layer, an output layer, and some hidden layers. Hidden layers in CNNs usually consist of convolutional layers, pooling layers, fully connected layers, and normalization layers.

Input layer is a layer that receives the input to the model. The number of neurons in this layer corresponds to the total number of features in the data (pixels for images).

Hidden levels are the inputs from the input layer that are then sent to the hidden layer. Depending on your model and data size, there may be many hidden layers. Each hidden layer can have a different number of neurons, but this is typically larger than the number of features. Each layer's output is computed by a matrix that multiplies the previous layer's output by the learnable weights for that layer, plus the learnable biases, followed by an activation function that makes the network nonlinear.

The output of the hidden layer is fed into a logistic function such as Sigmoid or Softmax, which transforms each class output into a probability value for each class.

7.2.3 LAYERS

- Stride determines how the weight matrix moves on the input. That is, jump one or two steps.
- Padding determines amount of pixel added to the image when processed by the CNN's kernel

- Max Pooling selects the largest elements from the area of the feature map covered by the filter. So the output after the max pooling layer will be a feature map containing the most salient features of the previous feature map.
- Dropout layer is Dropout works by probabilistically removing or "dropping out" inputs to a level. This could be the input variable in the data sample or the activation from the previous level. This has the effect of simulating a large number of networks with very different network structures, making the nodes in the network generally more robust to inputs.
- Dropout refers to the dropping of some neurons during the training phase (ignoring both forward and backward traversals) Randomly selected neurons.It is a fully connected essential component of convolutional neural networks (CNNs) and has proven very successful in image detection and classification for computer vision. A layer in which all inputs in one layer are connected to all activation units in the next layer. A fully connected layer that interprets features extracted from the convolutional part of the model should be flattened and connected to the output layer. That is the function of a fully connected layer, which is flattened. Also called dense layer.
- Use a flattening layer between the convolutional and dense layers to reduce the feature map to a single one-dimensional vector.

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

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

```
"C:\\Users\\Vinu Abinayaa\\anaconda3\\lib\\site-packages\\keras\\optimizer_v2\\adam.py:105: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.\n",
"  super(Adam, self).__init__(name, **kwargs)\n"
]
}
],
"source": [
"#Building a model based on the above defined function \n",
"model = load_CNN(6) #Number of Columns / Outputs \n",
"model.compile(loss='categorical_crossentropy', optimizer=Adam (lr=0.001), metrics=['accuracy'])\n",
"model.summary() #to print model summary\n",
"weights = model.get_weights () #to get the weights from our model"
]
},
```

7.2.4.2 Fitting Model : After compiling our model, we will train our model by the fit() method, then evaluate it.

```
"source": [
 "#\\\\\\\\\\\\\\\\//////////\\\\\\\n",
 "#\\\\\\\\\\\\\\\\//////////\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\/\\\\\\\\\\\\\\\\\\\\////////\n",
 "#Fitting the model on different batch sizes to see which ones turns out to be the best #\\\n",
 "\n",
 "#some arrays to store the result of each model (model trained on each bath size)\n",
 "histories_acc = [] \n",
 "histories_val_acc = []\n",
 "histories_loss = [] \n",
 "histories_val_loss = []\n",
 "model.set_weights (weights)\n",
 "h=model.fit(X_train, y_train, batch_size=16,\n",
 "epochs=1,\n",
 "verbose=1,\n",
 "callbacks=[early_st\n",
 "           1op_loss],\n",
 "shuffle=True,\n",
 "validation_data= (X_test, y_test))\n",
 "model.summary() #to print model summary"
]
},
```

7.3 Database Schema

There are three datasets namely **Bird, Flower, Mammal.**

7.3.1 Bird:

Great Indian Bustard Bird
Spoon Billed Sandpiper Bird

7.3.2 Flower:

Corpse Flower
Lady Slipper Orchid Flower

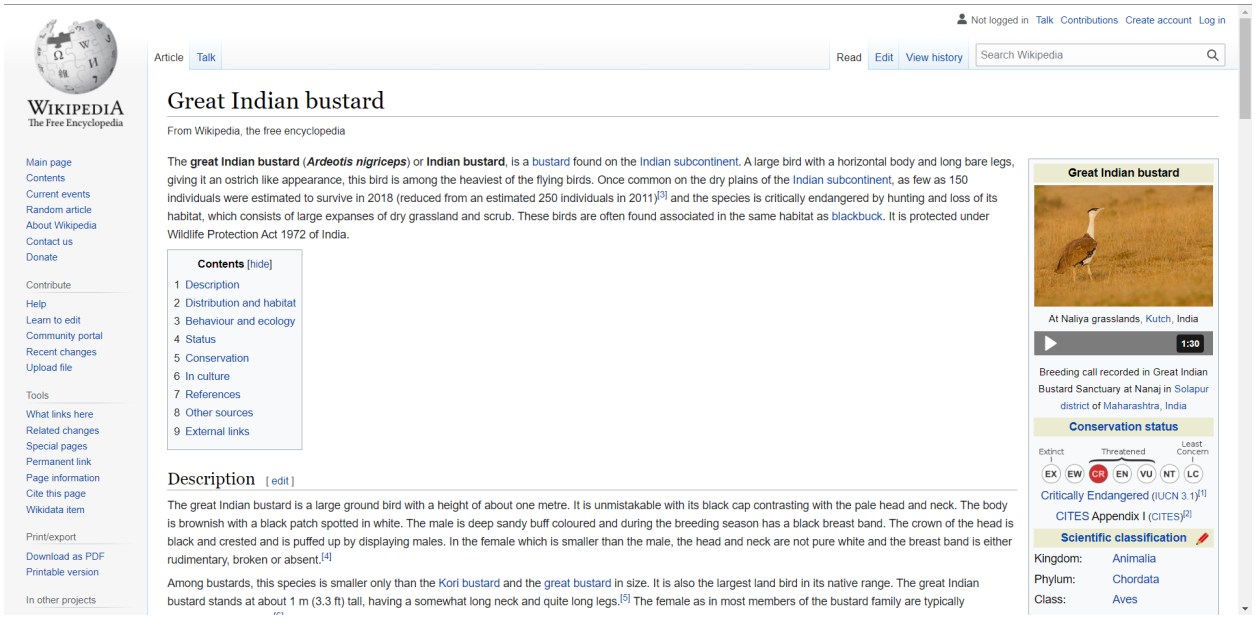
7.3.3 Mammal:

Pangolin Mammal
Seneca White Deer Mammal

CHAPTER-8
TESTING

8.1 Test Cases

- **Great Indian Bustard Bird**



Not logged in Talk Contributions Create account Log in

Article Talk | Read Edit View history | Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute
Help
Learn to edit
Community portal
Recent changes
Upload file

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page
Wikidata item

Print/export
Download as PDF
Printable version

In other projects

Great Indian bustard

From Wikipedia, the free encyclopedia

The **great Indian bustard** (***Ardeotis nigriceps***) or **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. Once common on the dry plains of the Indian subcontinent, as few as 150 individuals were estimated to survive in 2018 (reduced from an estimated 250 individuals in 2011)[3] and the species is critically endangered by hunting and loss of its habitat, which consists of large expanses of dry grassland and scrub. These birds are often found associated in the same habitat as blackbuck. It is protected under Wildlife Protection Act 1972 of India.

Contents [hide]
1 Description
2 Distribution and habitat
3 Behaviour and ecology
4 Status
5 Conservation
6 In culture
7 References
8 Other sources
9 External links

Description [edit]

The great Indian bustard is a large ground bird with a height of about one metre. It is unmistakable with its black cap contrasting with the pale head and neck. The body is brownish with a black patch spotted in white. The male is deep sandy buff coloured and during the breeding season has a black breast band. The crown of the head is black and crested and is puffed up by displaying males. In the female which is smaller than the male, the head and neck are not pure white and the breast band is either rudimentary, broken or absent.[4]

Among bustards, this species is smaller only than the Kori bustard and the great bustard in size. It is also the largest land bird in its native range. The great Indian bustard stands at about 1 m (3.3 ft) tall, having a somewhat long neck and quite long legs.[5] The female as in most members of the bustard family are typically

Great Indian bustard

At Naliya grasslands, Kutch, India

1:30

Breeding call recorded in Great Indian Bustard Sanctuary at Nanaj in Solapur district of Maharashtra, India

Conservation status

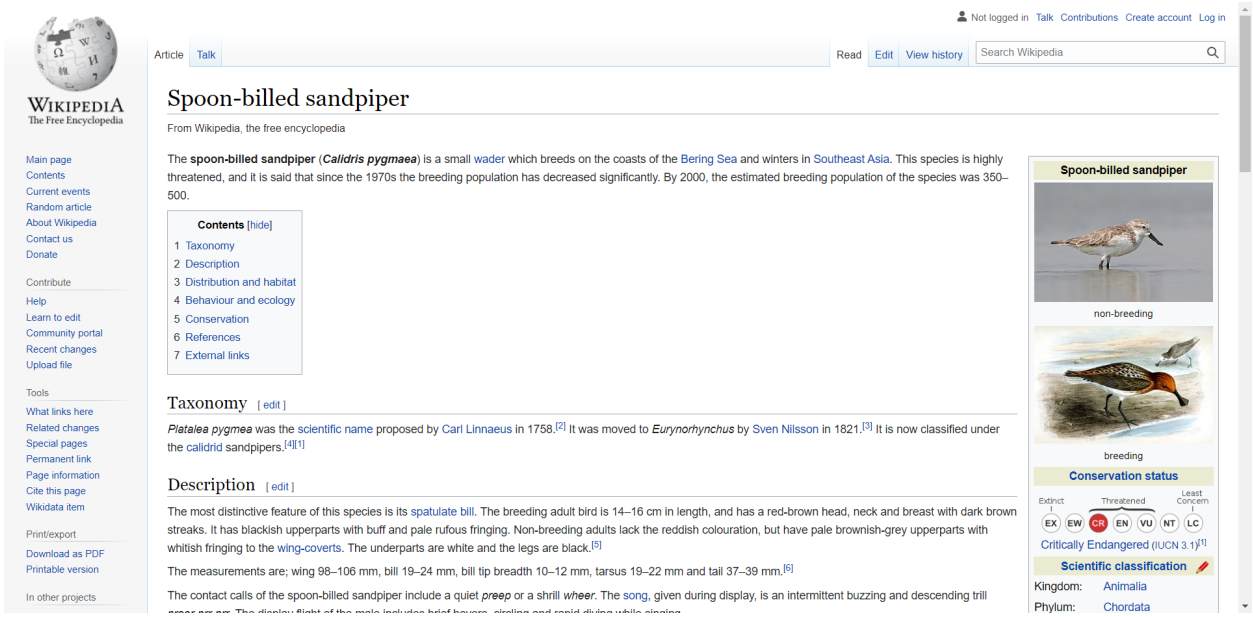
Extinct — Threatened — Least Concern
EX EW CR EN VU NT LC

Critically Endangered (IUCN 3.1)[1]
CITES Appendix I (CITES)[2]

Scientific classification

Kingdom: Animalia
Phylum: Chordata
Class: Aves

- **Spoon Billed Sandpiper Bird**



Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

WIKIPEDIA The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute
Help
Learn to edit
Community portal
Recent changes
Upload file

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page
Wikidata item

Print/export
Download as PDF
Printable version

In other projects

Spoon-billed sandpiper

From Wikipedia, the free encyclopedia

The **spoon-billed sandpiper** (***Calidris pygmaea***) is a small wader which breeds on the coasts of the Bering Sea and winters in Southeast Asia. This species is highly threatened, and it is said that since the 1970s the breeding population has decreased significantly. By 2000, the estimated breeding population of the species was 350–500.

Contents [hide]
1 Taxonomy
2 Description
3 Distribution and habitat
4 Behaviour and ecology
5 Conservation
6 References
7 External links

Taxonomy [edit]

Platalea pygmea was the scientific name proposed by Carl Linnaeus in 1758.[2] It was moved to *Eurynorhynchus* by Sven Nilsson in 1821.[3] It is now classified under the calidrid sandpipers.[4][1]

Description [edit]

The most distinctive feature of this species is its spatulate bill. The breeding adult bird is 14–16 cm in length, and has a red-brown head, neck and breast with dark brown streaks. It has blackish upperparts with buff and pale rufous fringing. Non-breeding adults lack the reddish colouration, but have pale brownish-grey upperparts with whitish fringing to the wing-coverts. The underparts are white and the legs are black.[5]

The measurements are; wing 98–106 mm, bill 19–24 mm, bill tip breadth 10–12 mm, tarsus 19–22 mm and tail 37–39 mm.[6]

The contact calls of the spoon-billed sandpiper include a quiet *preep* or a shrill *wheer*. The song, given during display, is an intermittent buzzing and descending trill

Spoon-billed sandpiper

non-breeding

breeding

Conservation status

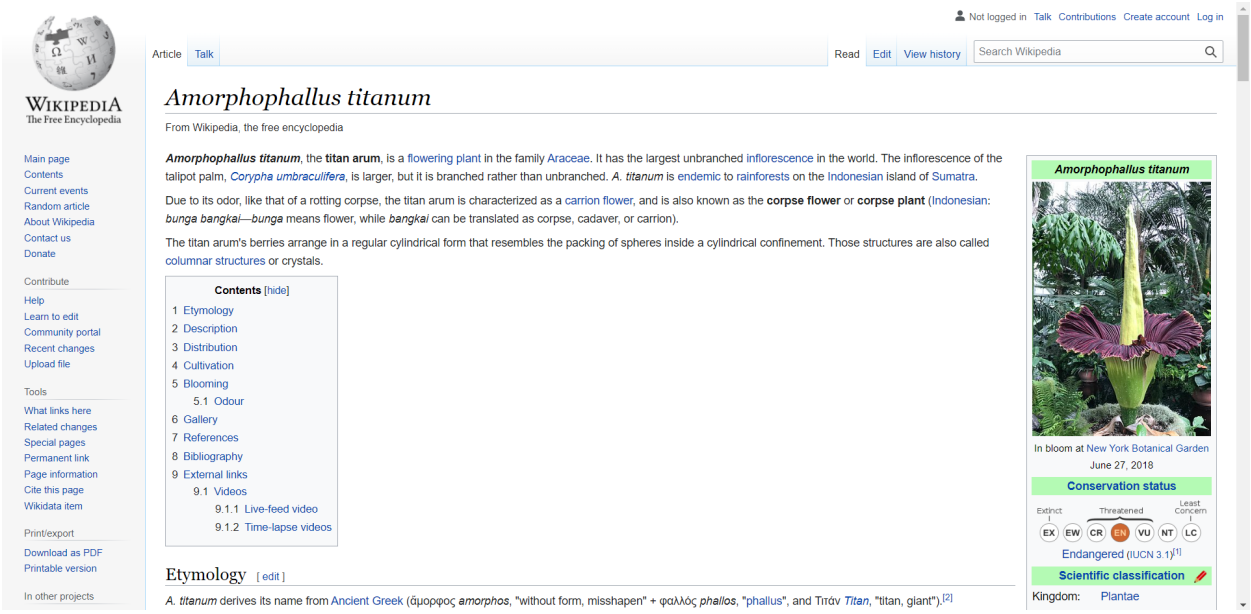
Extinct Threatened Least concern
EX EW CR EN VU NT LC

Critically Endangered (IUCN 3.1)[1]

Scientific classification

Kingdom: Animalia
Phylum: Chordata

- **Corpse Flower**



Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

WIKIPEDIA The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute
Help
Learn to edit
Community portal
Recent changes
Upload file

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page
Wikidata item

Print/export
Download as PDF
Printable version

In other projects

Amorphophallus titanum

From Wikipedia, the free encyclopedia

Amorphophallus titanum, the **titan arum**, is a flowering plant in the family Araceae. It has the largest unbranched inflorescence in the world. The inflorescence of the talipot palm, *Corypha umbraculifera*, is larger, but it is branched rather than unbranched. *A. titanum* is endemic to rainforests on the Indonesian island of Sumatra.

Due to its odor, like that of a rotting corpse, the titan arum is characterized as a carrion flower, and is also known as the **corpse flower** or **corpse plant** (Indonesian: *bunga bangkai*—*bunga* means flower, while *bangkai* can be translated as corpse, cadaver, or carrion).

The titan arum's berries arrange in a regular cylindrical form that resembles the packing of spheres inside a cylindrical confinement. Those structures are also called columnar structures or crystals.

Contents [hide]

1 Etymology
2 Description
3 Distribution
4 Cultivation
5 Blooming
5.1 Odour
6 Gallery
7 References
8 Bibliography
9 External links
9.1 Videos
9.1.1 Live-feed video
9.1.2 Time-lapse videos

Etymology [edit]

A. titanum derives its name from Ancient Greek (ἄμορφος *amorphos*, "without form, misshapen" + φαλλός *phallos*, "phallus", and Τιτάν *Titan*, "titan, giant").[2]

Amorphophallus titanum

In bloom at New York Botanical Garden
June 27, 2018

Conservation status

Extinct — Threatened — Least concern

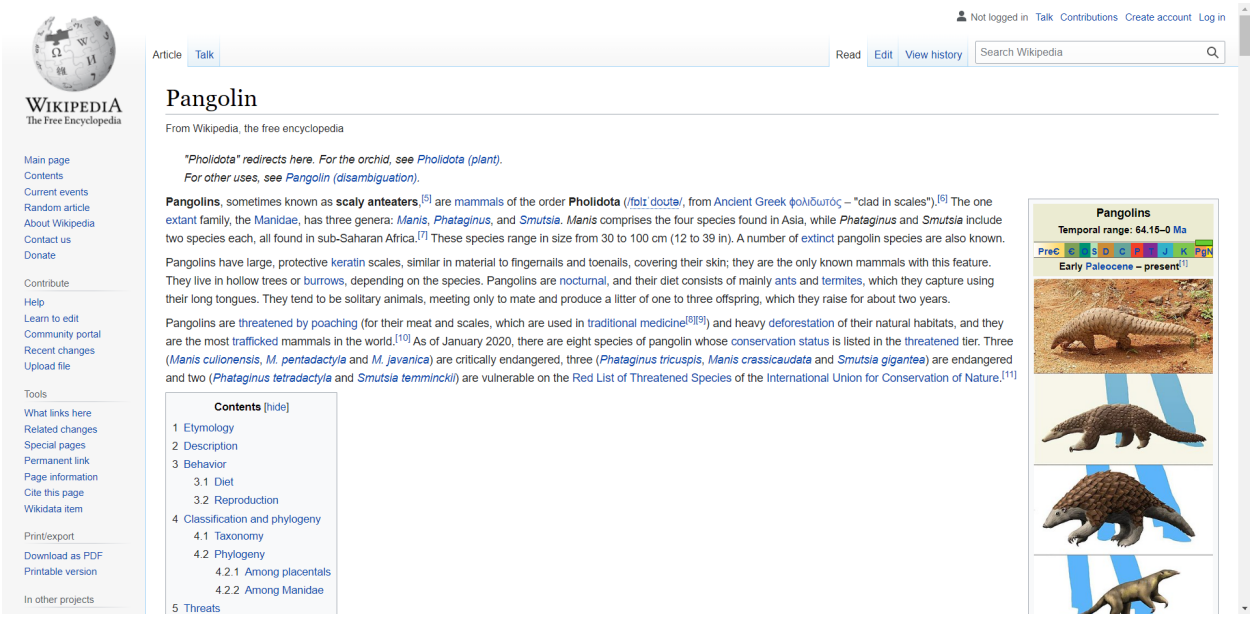
EX EW CR EN VU NT LC

Endangered (IUCN 3.1)[1]

Scientific classification

Kingdom: Plantae

- **Pangolin Mammal**



Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

WIKIPEDIA The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute
Help
Learn to edit
Community portal
Recent changes
Upload file

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page
Wikidata item

Print/export
Download as PDF
Printable version

In other projects

Pangolin

From Wikipedia, the free encyclopedia

"Pholidota" redirects here. For the orchid, see Pholidota (plant).
For other uses, see Pangolin (disambiguation).

Pangolins, sometimes known as **scaly anteaters**,[5] are mammals of the order **Pholidota** (/ˌfɒlɪˈdoʊtə/, from Ancient Greek φολιδωτός – "clad in scales").[6] The one extant family, the Manidae, has three genera: *Manis*, *Phataginus*, and *Smutsia*. *Manis* comprises the four species found in Asia, while *Phataginus* and *Smutsia* include two species each, all found in sub-Saharan Africa.[7] These species range in size from 30 to 100 cm (12 to 39 in). A number of extinct pangolin species are also known.

Pangolins have large, protective keratin scales, similar in material to fingernails and toenails, covering their skin; they are the only known mammals with this feature. They live in hollow trees or burrows, depending on the species. Pangolins are nocturnal, and their diet consists of mainly ants and termites, which they capture using their long tongues. They tend to be solitary animals, meeting only to mate and produce a litter of one to three offspring, which they raise for about two years.

Pangolins are threatened by poaching (for their meat and scales, which are used in traditional medicine[8][9]) and heavy deforestation of their natural habitats, and they are the most trafficked mammals in the world.[10] As of January 2020, there are eight species of pangolin whose conservation status is listed in the threatened tier. Three (*Manis culionensis*, *M. pentadactyla* and *M. javanica*) are critically endangered, three (*Phataginus tricuspis*, *Manis crassicaudata* and *Smutsia gigantea*) are endangered and two (*Phataginus tetradactyla* and *Smutsia temminckii*) are vulnerable on the Red List of Threatened Species of the International Union for Conservation of Nature.[11]

Contents [hide]

1 Etymology
2 Description
3 Behavior
3.1 Diet
3.2 Reproduction
4 Classification and phylogeny
4.1 Taxonomy
4.2 Phylogeny
4.2.1 Among placentals
4.2.2 Among Manidae
5 Threats

Pangolins

Temporal range: 64.15–0 Ma

PreC € S D C P T J K PgN

Early Paleocene – present[1]

8.2 User Acceptance Testing

8.2.1 Defect Analysis

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

Resolution	Low Severity	Medium Severity	High Severity	Subtotal of bugs
By UI	1	3	2	6
By Functionality	1	2	2	5
Duplicate	0	4	5	9
External	2	2	0	4
Fixed	1	4	4	9
Not Reproduced	0	2	1	3
Skipped	0	0	0	0
Won't Fix	0	0	1	1
Totals	5	17	15	37

8.2.2 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	5	0	0	5
Flask Application	4	0	0	4
Exception Reporting	1	0	0	1
Final Report Output	1	0	0	1
Version Control	2	0	0	2

CHAPTER-9
RESULTS

9.1 Performance Metrics

9.1.1 Accuracy:

Training Accuracy – **71%**

Validation Accuracy – **76%**

```
In [28]: #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'])
         histories_acc [[0.49351853]] histories_loss [[1.2544775]] histories_val_acc [[0.71496564]] histories_val_loss [[0.76086265]]
```

```
In [27]: #\\\\\\\/////////\\\
         #\\\\\\\/////////\\\\\\\\\\\\\\\\\/\\\\\\\\\/////////
         #Fitting the model on different batch sizes to see which ones turns out to be the best #\

         #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=1,
         verbose=1,
         callbacks=[early_stop_loss],
         shuffle=True,
         validation_data= (X_test, y_test))
         model.summary() #to print model summary
```

```
473/473 [==============================] - 2275s 5s/step - loss: 1.2545 - accuracy: 0.4935 - val_loss: 0.7609 - val_accuracy:
0.7150
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


 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
_________________________________________________________________
```

CHAPTER-10
ADVANTAGES & DISADVANTAGES

10.1 ADVANTAGES

● It helps field naturalists build their own searching tools.
● It helps naturalists to explore new interactions with nature.
● It discovers new ways of experiencing the natural world.
● It helps in creating a sustainable world by saving endangered species.
● Digital Naturalism unites biologists, zoologists, engineers, and artists to build and analyze new devices.
● Users can get detailed descriptions of any kind of species.
● It is a handy application for a person who is a travel freak.

10.2 DISADVANTAGES

● Proper network is needed at times to avoid glitches.
● Difficult to classify sub-classes of the same species.

CHAPTER-11
CONCLUSION

In a new era of extinction, humans (endogenous or otherwise) understand that our state's biodiversity is threatened. Fewer observations, data points and discoveries made in nature will help measure their impact on nature. This Provides an opportunity to build a positive identity with science or recognize its value. A more holistic view, such as traditional ecological knowledge helps to participate from resource conservation to construction, in green action resilience among vulnerable communities. By acknowledging our evolutionary origins, A scientific perspective also strengthens our sense of relatedness to other species. This model, used for digital naturalist detection using pictures of species from wildlife and species with plant parts and animal parts is also displayed. The resulting graph proves that the model is accurate and has good scope. It also helps a lot of people when used in real-time scenarios by differentiating the two without wasting money on different machines. If the photo is confirmed with a model, you can know the characteristics of the species. As you know, data plays an important role . In all deep learning models, if the data is more specific and accurate with respect to species, higher accuracy and better results in real-time applications can be observed.

CHAPTER-12
FUTURE SCOPE

Recent reports of global biodiversity decline make it more important than ever to monitor biodiversity so that we can detect changes and infer their drivers. Online digital media, such as social media images, digital magazines may be a new source of biodiversity observations,but they are far too numerous for a human to practically review. It promotes the arts and technology in combination can stimulate connections in heritage gardens, and also nurture care for non-human. Geographical identification of species along with their source of origin. 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 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 weakness of the predictors.

CHAPTER-13
APPENDIX

13.1 Source Code

```
"from keras.preprocessing.image import ImageDataGenerator\n",
   "import cv2\n",
   "from os import listdir\n",
   "import time    \n",
   "\n",
   "def hms_string(sec_elapsed):\n",
   "    h = int(sec_elapsed / (60 * 60))\n",
   "    m = int((sec_elapsed % (60 * 60)) / 60)\n",
   "    s = sec_elapsed % 60\n",
   "    return f\"{h}:{m}:{round(s,1)}\"\n",
   "\n",
   "def augment_data(file_dir, n_generated_samples, save_to_dir):\n",
   "    data_gen = ImageDataGenerator(rotation_range=30, \n",
   "                                  width_shift_range=0.1,\n",
   "                                  height_shift_range=0.15, \n",
   "                                  shear_range=0.25, \n",
```

```
"                           zoom_range = 0.2,\n",
"                           horizontal_flip=True, \n",
"                           vertical_flip=False, \n",
"                           fill_mode='nearest',\n",
"                           brightness_range=(0.5,1.2)\n",
"                          )\n",
"\n",
"    \n",
"    for filename in listdir(file_dir):\n",
"        image = cv2.imread(file_dir + '/' + filename)\n",
"        image = image.reshape((1,)+image.shape)\n",
"        save_prefix = 'aug_' + filename[:-4]    \n",
"        i=0\n",
"        for batch in data_gen.flow(x=image, batch_size=1, save_to_dir=save_to_dir,\n",
"                                   save_prefix=save_prefix, save_format='jpg'):\n",
"            i += 1\n",
"            if i > n_generated_samples:\n",
"                break\n",
"            \n",
"start_time = time.time()\n",
"\n",
"augmented_data_path = \"/Users/Vinu Abinayaa/Desktop/Augmented Dataset\"\n",
"augment_data(file_dir='/Users/Vinu Abinayaa/Desktop/Digital Naturalist Dataset/Bird/Great Indian Bustard Bird', n_generated_samples=8, save_to_dir=augmented_data_path+'/Bird/GIB_AUG')\n",
"augment_data(file_dir='/Users/Vinu Abinayaa/Desktop/Digital Naturalist Dataset/Flower/Corpse Flower', n_generated_samples=8, save_to_dir=augmented_data_path+'/Flower/Corpse_AUG')\n",
"augment_data(file_dir='/Users/Vinu Abinayaa/Desktop/Digital Naturalist Dataset/Flower/Lady Slipper Orchid Flower', n_generated_samples=8, save_to_dir=augmented_data_path+'/Flower/LS_Orchid_AUG')\n",
"augment_data(file_dir='/Users/Vinu Abinayaa/Desktop/Digital Naturalist Dataset/Mammal/Pangolin Mammal', n_generated_samples=8, save_to_dir=augmented_data_path+'/Mammal/LS_Pangolin_AUG')\n",
```

"augment_data(file_dir='/Users/Vinu Abinayaa/Desktop/Digital Naturalist Dataset/Mammal/Senenca White Deer Mammal', n_generated_samples=8, save_to_dir=augmented_data_path+'/Mammal/SW_Deer_AUG')\n",
"\n",
"\n",
"end_time = time.time()\n",
"execution_time = (end_time - start_time)\n",
"print(\"Elapsed Time : \"+str(execution_time))"
]
},
{
"cell_type": "code",
"execution_count": 2,
"id": "85d87892",
"metadata": {},
"outputs": [],
"source": [
"#For matrix calculations and data Management\n",
"import numpy as np\n",
"\n",
"\n",
"\n",
"#Importing libraries \n",
"import tensorflow as tf\n",
"\n",
"import keras \n",
"import keras.backend as k\n",
"#from keras.optimizers import SGD, Adam, Adagrad, RMSprop\n",
"\n",
"from tensorflow.keras.optimizers import Adam,Nadam, SGD\n",
"\n",
"\n",
"from keras.applications import *\n",
"\n",

"from keras.preprocessing import *\n",
"\n",
"from keras.preprocessing. image import ImageDataGenerator, array_to_img, img_to_array, load_img\n",
"\n",
"from keras.callbacks import EarlyStopping, ModelCheckpoint\n",
"\n",
"from keras.models import Sequential\n",
"\n",
"from keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Activation, BatchNormalization, Dropout\n",
"\n",
"from keras.utils.np_utils import to_categorical \n",
"\n",
"from sklearn.model_selection import train_test_split\n",
"\n",
"#For plotting charts ed for data visualizations\n",
"\n",
"import matplotlib.pyplot as plt #Libraries for Locating and loading date\n",
"\n",
"import glob \n",
"\n",
"from PIL import Image\n",
"\n",
"import os \n",
"from os import listdir"
]
},
{
"cell_type": "code",
"execution_count": 3,
"id": "1794b155",
"metadata": {},
"outputs": [

```
    {
     "data": {
      "text/plain": [
       "['/Users/Vinu Abinayaa/Desktop/Augmented Dataset\\\\Bird\\\\GIB_AUG',\n",
       " '/Users/Vinu Abinayaa/Desktop/Augmented Dataset\\\\Bird\\\\SPS_AUG',\n",
       " '/Users/Vinu Abinayaa/Desktop/Augmented Dataset\\\\Flower\\\\Corpse_AUG',\n",
       " '/Users/Vinu Abinayaa/Desktop/Augmented Dataset\\\\Flower\\\\LS_Orchid_AUG',\n",
       " '/Users/Vinu Abinayaa/Desktop/Augmented Dataset\\\\Mammal\\\\LS_Pangolin_AUG',\n",
       " '/Users/Vinu Abinayaa/Desktop/Augmented Dataset\\\\Mammal\\\\SW_Deer_AUG']"
      ]
     },
     "execution_count": 3,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
"source": [
    "#Setting path to our dataset folder\n",
    "dirName = \"/Users/Vinu Abinayaa/Desktop/Augmented Dataset\"\n",
    "folders = listdir(dirName)\n",
    "#Getting the names for all the folders containing data\n",
    "def getListOfFiles (dirName):\n",
    "# create a list of sub directories and files (if any) # names in the given directory\n",
    "    listOfFile = os.listdir (dirName) \n",
    "    allFiles = list()\n",
    "    for fol_name in listOfFile:\n",
    "        fullPath = os.path.join(dirName, fol_name) \n",
    "        allFiles.append(fullPath)\n",
    "    return allFiles\n",
    "Folders = getListOfFiles(dirName) \n",
    "len (Folders)\n",
    "subfolders = []\n",
    "for num in range(len(Folders)):\n",
    "    sub_fols = getListOfFiles (Folders [num]) \n",
```

```
   "    subfolders+=sub_fols\n",
   "#Now, the subfolders contains the address to all our data folders for each class \n",
   "subfolders"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 4,
  "id": "e5aecf54",
  "metadata": {},
  "outputs": [],
  "source": [
   "#\\\\\\\\\n",
   "#\\\\\\\\\\\\\\\\\\\\/////////\\\\\\\\\\\\\\\\\\\\/////////\\\\\\\\\\\\\\\\\\\\\\\\/////////\n",
   "#Loading the data and pre processing it to make it in trainable format #\\\\\\'////\\\\\\\\\\\n",
   "#X data will includes the data generated for each image\n",
    "#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]\n",
   "#That will be tha label we're classifying.\n",
   "X_data = []\n",
   "Y_data = []\n",
   "id_no=0\n",
   "#to make a list of tuples, where we'll store the info about the image, category and species\n",
   "found = []\n",
   "#itering in all folders under Augmented data folder\n",
   "for paths in subfolders:\n",
   "#setting folder path for each unique class and category\n",
   "    files= glob.glob (paths + \"/*.jpg\")\n",
   "#adding tuples to the list that contain folder name and subfolder name \n",
   "    found.append((paths.split('\\\\')[-2],paths.split('\\\\' ) [-1] ) )\n",
   "#itering all files under the folder one by one\n",
   "    for myFile in files:\n",
   "        img=Image.open(myFile)\n",
   "#img.thumbnail((width, height), Image.ANTIALIAS) # resizes image in-place keeps ratio \n",
```

```
    "          img = img.resize((224,224), Image.ANTIALIAS) # resizes image without ratio #convert the images to numpy arrays\n",
    "       img = np.array(img)\n",
    "       if img.shape == ( 224, 224, 3):\n",
    "# Add the numpy image to matrix with all data\n",
    "           X_data.append (img)\n",
    "           Y_data.append (id_no)\n",
    "   id_no+=1"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "da3fb45f",
   "metadata": {},
   "outputs": [],
   "source": [
    "#to see our data\n",
    "print(X_data) \n",
    "print(Y_data)\n",
    "#converting lists to np arrays again\n",
    "X = np.array(X_data)\n",
    "Y = np.array(Y_data)\n",
    "# Print shapes to see if they are correct\n",
    "print(\"x-shape\", X. shape, \"y shape\", Y.shape)\n",
    "X = X.astype('float32')/255.0\n",
    "   "#The Keras library offers a function called to_categorical() #integer data. The sequence has an example of all known values #so we can use the to_categorical() function directly \n",
    "y_cat=to_categorical (Y_data, len(subfolders))\n",
    "print(\"X shape\",X,\"y_cat_shape\", y_cat)\n",
    "print(\"X shape\", X. shape, \"y_cat shape\", y_cat.shape)\n",
    "#\\\\\\\\\\\\you can use to one hot encode\n",
    "\n",
    "#\\\\\\\\\\\\\\\\\\\\/////////\\\\ #Splitting the data to Test and Train\n",
```

```
  "\n",
  "X_train, X_test, y_train, y_test = train_test_split(X, y_cat, test_size=0.2) \n",
  "print(\"The model has \" + str(len(X_train)) + \" inputs\")"
 ]
},
{
 "cell_type": "code",
 "execution_count": 8,
 "id": "e51e4824",
 "metadata": {},
 "outputs": [],
 "source": [
  "#\\\\\n",
  "#MODEL BUILDING\n",
  "#\\\\\n",
  "early_stop_loss = EarlyStopping (monitor='loss', patience=3, verbose=1) \n",
  "early_stop_val_acc = EarlyStopping (monitor='val_accuracy', patience=3, verbose=1) \n",
  "model_callbacks=[early_stop_loss, early_stop_val_acc]"
 ]
},
{
 "cell_type": "code",
 "execution_count": 9,
 "id": "fd9e35b1",
 "metadata": {},
 "outputs": [],
 "source": [
  "#defining our model, All the layers and configurations \n",
  "def load_CNN (output_size):\n",
  "    k.clear_session()\n",
  "    model= Sequential()\n",
  "    model.add(Dropout (0.4,input_shape=(224, 224, 3)))\n",
  "    model.add(Conv2D (256, (5, 5), input_shape=(224, 224, 3), activation='relu')) \n",
  "    model.add(MaxPool2D(pool_size=(2, 2)))\n",
```

```
  "#model.add(BatchNormalization())\n",
  "    model.add(Conv2D(128, (3, 3), activation='relu')) \n",
  "    model.add(MaxPool2D(pool_size=(2, 2))) #model.add(BatchNormalization())\n",
  "    model.add(Conv2D(64, (3, 3), activation='relu'))\n",
  "    model.add(MaxPool2D(pool_size=(2, 2))) #model.add(BatchNormalization())\n",
  "    model.add(Flatten())\n",
  "    model.add(Dense (512, activation='relu'))\n",
  "    model.add(Dropout (0.3))\n",
  "    model.add(Dense (256, activation='relu'))\n",
  "    model.add(Dropout (0.3))\n",
  "    model.add(Dense (128, activation='relu'))\n",
  "    model.add(Dropout (0.3))\n",
  "    model.add(Dense(output_size, activation='softmax'))\n",
  "    return model"
 ]
},
{
 "cell_type": "code",
 "execution_count": 10,
 "id": "0121930e",
 "metadata": {
  "scrolled": true
 },
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Model: \"sequential\"\n",
    "_________________________________________________________________\n",
    " Layer (type)                Output Shape              Param #   \n",
    "=================================================================\n",
    " dropout (Dropout)           (None, 224, 224, 3)       0         \n",
    "                                                                 \n",
```

```
" conv2d (Conv2D)            (None, 220, 220, 256)     19456     \n",
"                                                                 \n",
" max_pooling2d (MaxPooling2D  (None, 110, 110, 256)    0         \n",
" )                                                               \n",
"                                                                 \n",
" conv2d_1 (Conv2D)          (None, 108, 108, 128)     295040    \n",
"                                                                 \n",
" max_pooling2d_1 (MaxPooling  (None, 54, 54, 128)      0         \n",
" 2D)                                                            \n",
"                                                                 \n",
" conv2d_2 (Conv2D)          (None, 52, 52, 64)        73792     \n",
"                                                                 \n",
" max_pooling2d_2 (MaxPooling  (None, 26, 26, 64)       0         \n",
" 2D)                                                            \n",
"                                                                 \n",
" flatten (Flatten)          (None, 43264)             0         \n",
"                                                                 \n",
" dense (Dense)              (None, 512)               22151680  \n",
"                                                                 \n",
" dropout_1 (Dropout)        (None, 512)               0         \n",
"                                                                 \n",
" dense_1 (Dense)            (None, 256)               131328    \n",
"                                                                 \n",
" dropout_2 (Dropout)        (None, 256)               0         \n",
"                                                                 \n",
" dense_2 (Dense)            (None, 128)               32896     \n",
"                                                                 \n",
" dropout_3 (Dropout)        (None, 128)               0         \n",
"                                                                 \n",
" dense_3 (Dense)            (None, 6)                 774       \n",
"                                                                 \n",
"=================================================================\n",
"Total params: 22,704,966\n",
"Trainable params: 22,704,966\n",
```

```
    "Non-trainable params: 0\n",
    "_________________________________________________________________\n"
   ]
  },
  {
   "name": "stderr",
   "output_type": "stream",
   "text": [
        "C:\\Users\\Vinu  Abinayaa\\anaconda3\\lib\\site-packages\\keras\\optimizer_v2\\adam.py:105: UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.\n",
    "  super(Adam, self).__init__(name, **kwargs)\n"
   ]
  }
 ],
 "source": [
  "#Building a model based on the above defined function \n",
  "model = load_CNN(6) #Number of Columns / Outputs \n",
  "model.compile(loss='categorical_crossentropy', optimizer=Adam (lr=0.001), metrics=['accuracy'])\n",
  "model.summary() #to print model summary\n",
  "weights = model.get_weights () #to get the weights from our model"
 ]
},
{
 "cell_type": "code",
 "execution_count": 11,
 "id": "b8683783",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
      "315/315 [==============================] - 1233s 4s/step - loss: 1.6302 - accuracy: 0.3270 - val_loss: 1.1657 - val_accuracy: 0.5429\n",
```

```
"Model: \"sequential\"\n",
"_________________________________________________________________\n",
" Layer (type)                Output Shape              Param #   \n",
"=================================================================\n",
" dropout (Dropout)           (None, 224, 224, 3)       0         \n",
"                                                                 \n",
" conv2d (Conv2D)             (None, 220, 220, 256)     19456     \n",
"                                                                 \n",
" max_pooling2d (MaxPooling2D  (None, 110, 110, 256)    0         \n",
" )                                                               \n",
"                                                                 \n",
" conv2d_1 (Conv2D)           (None, 108, 108, 128)     295040    \n",
"                                                                 \n",
" max_pooling2d_1 (MaxPooling  (None, 54, 54, 128)      0         \n",
" 2D)                                                            \n",
"                                                                 \n",
" conv2d_2 (Conv2D)           (None, 52, 52, 64)        73792     \n",
"                                                                 \n",
" max_pooling2d_2 (MaxPooling  (None, 26, 26, 64)       0         \n",
" 2D)                                                            \n",
"                                                                 \n",
" flatten (Flatten)           (None, 43264)             0         \n",
"                                                                 \n",
" dense (Dense)               (None, 512)               22151680  \n",
"                                                                 \n",
" dropout_1 (Dropout)         (None, 512)               0         \n",
"                                                                 \n",
" dense_1 (Dense)             (None, 256)               131328    \n",
"                                                                 \n",
" dropout_2 (Dropout)         (None, 256)               0         \n",
"                                                                 \n",
" dense_2 (Dense)             (None, 128)               32896     \n",
"                                                                 \n",
" dropout_3 (Dropout)         (None, 128)               0         \n",
```

```
    "                                                 \n",
    " dense_3 (Dense)            (None, 6)               774       \n",
    "                                                 \n",
    "=================================================================\n",
    "Total params: 22,704,966\n",
    "Trainable params: 22,704,966\n",
    "Non-trainable params: 0\n",
    "_________________________________________________________________\n"
   ]
  }
 ],
 "source": [
  "#\\\\\\\\\\\\\\\/////////\\\\\\\n",
  "#\\\\\\\\\\\\\\\/////////\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\/\\\\\\\\\\\\\\\\\\\/////////\n",
  "#Fitting the model on different batch sizes to see which ones turns out to be the best #\\\n",
  "\n",
  "#some arrays to store the result of each model (model trained on each bath size)\n",
  "histories_acc = [] \n",
  "histories_val_acc = []\n",
  "histories_loss = [] \n",
  "histories_val_loss = []\n",
  "model.set_weights (weights)\n",
  "h=model.fit(X_train, y_train, batch_size=16,\n",
  "epochs=1,\n",
  "verbose=1,\n",
  "callbacks=[early_st\n",
  "          1op_loss],\n",
  "shuffle=True,\n",
  "validation_data= (X_test, y_test))\n",
  "model.summary() #to print model summary"
 ]
},
{
 "cell_type": "code",
```

```
  "execution_count": 13,
  "id": "2fc1e797",
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])\n",
          "histories_acc [[0.32697278]] histories_loss [[1.6302495]] histories_val_acc [[0.5429253]] histories_val_loss [[1.16568089]]\n"
    ]
   }
  ],
  "source": [
   "#printing the keys we have for the stores values\n",
   "print(h.history.keys())\n",
   "#appendind the data for each epoch in a arr, and for each batch size\n",
   "histories_acc.append(h.history['accuracy'])\n",
   "histories_val_acc.append(h.history['val_accuracy'])\n",
   "histories_loss.append(h.history['loss']) \n",
   "histories_val_loss.append(h.history['val_loss'])\n",
   "#converting into numpy arrays\n",
   "histories_acc = np.array(histories_acc)\n",
   "histories_val_acc=np.array(histories_val_acc)\n",
   "histories_loss = np.array (histories_loss)\n",
   "histories_val_loss = np.array(histories_val_loss)\n",
   "\n",
    "#here we have 3 columns and 6 rows each, ever row represetns differnt bath size, #every column represent different epoch scores. \n",
   "print('histories_acc',histories_acc,\n",
   "'histories_loss', histories_loss,\n",
   "'histories_val_acc', histories_val_acc, 'histories_val_loss', histories_val_loss)"
  ]
```

```
 },
{
  "cell_type": "code",
  "execution_count": 24,
  "id": "75a923dc",
  "metadata": {},
  "outputs": [],
  "source": [
   "\n",
   "#Predicting the image's classes\n",
   "#individual scores for each class as well as class with the highest score is printed\n",
   "\n",
   "#\\\\\n",
   "\n",
   "#making predictions, storing result as array of probabilities of each class predicted \n",
   "#predictions = model.predict([X_test[image_number].reshape(1, 224,224,3)])\n",
   "#for idx, result, x in zip(range(0,6), found, predictions[0]):\n",
       "        #print(\"Label: {}, Type: {}, Species: {}, Score: {}%\".format(idx, result[0], result[1], 
round(x*100,3)))\n",
   "#predicting the class with max probability\n",
   "#ClassIndex=model.predict([X_test[image_number].reshape(1, 224,224,3)])\n",
   "#getting the index of the class which we can pass #to the boat_types list to get the boat type name \n",
   "#ClassIndex\n",
   "#printing the final output\n",
   "#print (found [ClassIndex[0]])"
  ]
 },
 {
  "source": [
   "#loading Test Data\n",
   "image_number = np.random.randint(0,len (X_test))\n",
   "print(image_number)\n",
   "#plotting the test image \n",
   "plt.figure(figsize=(8, 8))\n",
```

```
  "plt.imshow(X_test[image_number])"
 ]
},
{
 "cell_type": "code",
 "execution_count": 18,
 "id": "12b49215",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "Saved model to disk\n"
   ]
  }
 ],
 "source": [
  "\n",
  "#\\\\ #\\\\\\\\\\\\\\n",
  "#savig necessary model files #\\\\\\\\\\\\\\\ #\\\\\\\n",
  "# serialize model to JSON\n",
  "model_json = model.to_json() #indent=2\n",
  "with open(\"final_model.json\", \"w\") as json_file: \n",
  "    json_file.write(model_json)\n",
  "# serialize weights to H5\n",
  "model.save_weights(\"final_model.h5\") \n",
  "print(\"Saved model to disk\")\n"
 ]
},
{
 "cell_type": "code",
 "execution_count": null,
 "id": "5eff2ddc",
```

```
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.8.8"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```

GitHub & Project Demo Link

Git Hub Link:

https://github.com/IBM-EPBL/IBM-Project-22456-1659852338

Project Demo Link:

https://drive.google.com/file/d/1-4ejOrS7fz5PWhBXB17QdXkiZrzSlYL2/view