**IMAGE RECOGNITION OF SPECIES USING CNN**



Submitted in partial fulfillment of the requirement

For

The award of degree of

**Bachelor of Technology**

**in**

**Computer Science & Engineering**

**2021-2025**

**Submitted By**

**Batch no: 98**

VAMSHI PANAKA – 2103A51375

VANCHANAGIRI YASHWANTH – 2103A51439

KOMURAVELLY BHARADWAJ – 2103A51169

HRUSHITH PATEL – 2103A51084

GADDAM SAI PRANAY – 2103A51401

*Under the Guidance of*

**Dr. N. Sharmila Banu**

**Assistant Professor**



**Department of CSE & AI**

**SR University**

**Anathasagar, Hasanparthy, Hanumakonda 506371**

**AY – 2023-2024**



**SCHOOL OF COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE**

**CERTIFICATE**

This is to certify that this project entitled **“PROJECT TITLE**" is the bonafied work carried out by **NAME1,NAME2,NAME3(add if any)** as a Minor Project for the partial fulfillment to award the degree **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE** during the academic year 2022-2023 under our guidance and Supervision.

**Dr. N. Sharmila Banu Dr. M.Sheshikala**

Asst. Prof , Assoc. Prof. & HOD (CSE),

SR University, SR University,Ananthasagar, Warangal. Ananthasagar, Warangal.

**External Examiner**

**ACKNOWLEDGEMENT**

We owe an enormous debt of gratitude to our project guide Dr. Guide Name, Assistant Professor as well as Head of the CSE Department Dr. M.Sheshikala, Associate Professor for guiding us from the beginning through the end of the Minor Project with their intellectual advices and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

We express our thanks to the project co-ordinators Dr. P Praveen, Assoc. Prof for their encouragement and support.

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved Dean, Dr.Indrajeet Gupta,Professor for his continuous support and guidance to complete this project in the institute.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

VAMSHI PANAKA 2103A51375

VANCHANAGIRI YASHWANTH 2103A51439

KOMURAVELLY BHARADWAJ 2103A51169

HRUSHITH PATEL 2103A51084

GADDAM SAI PRANAY 2103A51401

# ABSTRACT

Accurate species identification is the basis for all aspects of taxonomic research and is an essential component of workflows in biological research. Biologists are asking for more efficient methods to meet the identification demand. Smart mobile devices, digital cameras as well as the mass digitization of natural history collections led to an explosion of openly available image data depicting living organisms. This rapid increase in biological image data in combination with modern machine learning methods, such as deep learning, offers tremendous opportunities for automated species identification. Deep learning neural networks as a technology that enabled breakthroughs in automated species identification. A real-world animal biometric system which detects and describes animal life in image and video is an emerging subject in machine vision. These systems provide computer vision approaches for the classification of animals. A novel method for animal face classification based on one of the popular convolutional neural network (CNN) features. We are using CNN in this project which can automatically extract features, learn and classify them. The experimental results show that automatic feature extraction in CNN is better compared to other simple feature extraction techniques.

**About the Organization**

Sri Rajeshwara Educational Society, the parent body of SR University is a 45-year-old conglomerate of educational institutions with more than 90,000 students and 10,000 teaching and non-teaching staff members. SR Educational Academy governs 95 Educational Institutions across Telangana and Andhra Pradesh.

The goal of SR University is to create an innovative learning educational ecosystem whose graduates significantly contribute to the growth of Telangana and India. We plan to transform the educational system through three key differentiators.

##### Key Differentiators

At the core of SRU, exceptional faculty engage students in new experiences through innovative teaching approaches. SRU is home for such faculty who are ever curious to question existing knowledge, push the boundaries of understanding through research, and incorporate innovative teaching- pedagogies for better learning experiences.

|  |  |  |  |
| --- | --- | --- | --- |
| **LIST OF FIGURES** | | | |
| **S.NO** | **Figure No.** | **Title of Figure** | **Page No** |
| **1** | **1** | **System Architecture** | **13** |
| **2** | **2** | **Flow Chart Diagram** | **14** |
| **3** | **3** | **Class Diagram** | **15** |
| **4** | **4** | **Use Case Diagram** | **16** |
| **5** | **5** | **Sequence Diagram** | **17** |
| **6** | **6** | **Activity Diagram** | **18** |

**INDEX**

|  |  |
| --- | --- |
| **CONTENT** | **Page No** |
| **List of Figures** |  |
| **1. INTRODUCTION** |  |
| 1.1 Project Introduction | **1** |
| 1.2 Scope | **2** |
| 1.3 Project Overview | **3** |
| 1.4 Objectives | **3** |
| 1.5 Data Set | **3** |
| **2. LITERATURE SURVEY** |  |
| 2.1 Existing System | **4** |
| 2.2 Proposed System | **5** |
| 2.3 Related Work | **6** |
| **3. SYSTEM ANALYSIS** |  |
| 3.1 Functional Requirements | **7** |
| 3.2 Performance Requirements | **7** |
| 3.3 Software Requirements | **8** |
| 3.4 Hardware Requirements | **9** |
| 3.5 Feasibility study (Technical/Economical/Operational) | **9** |
| **4. SYSTEM DESIGN** |  |
| 4.1 System Architecture | **12** |
| 4.2 Data Flow Diagram | **14** |
| 4.3 UML Diagrams | **14** |
| 4.3.1 Class Diagram | **14** |
| 4.3.2 Use Case Diagram | **15** |
| 4.3.3 Sequence Diagram | **16** |
| 4.3.4 Activity Diagram | **17** |
| 4.3.5 Collaboration Diagram | **18** |
| 4.4 Modules | **19** |
| **5. IMPLIMENTATION AND RESUTS** |  |
| 5.1 Language / Technology Used | **20** |
| 5.2 Methods / Algorithms Used | **20** |
| 5.3 Sample Code | **21** |
| 5.4 Results (Accuracy) / Output Screens | **25** |
| **6. TESTING** | **26** |
| **7. CONCLUSIONS** | **27** |
| **8. FUTURE SCOPE** | **28** |
| **9. BIBLIOGRAPHY** | **29** |

# INTRODUCTION

## Project Introduction

The various deep learning methods use data to train neural network algorithms to do a variety of machine learning tasks, such as the classification of different classes of objects. Convolutional neural networks are deep learning algorithms that are very powerful for the analysis of images. This will explain to you how to construct, train and evaluate convolutional neural networks.

You will also learn how to improve their ability to learn from data, and how to interpret the results of the training. Deep Learning has various applications like image processing, natural language processing, etc. It is also used in Medical Science, Media & Entertainment, Autonomous Cars, etc.

There are an estimated 8.7 million wildlife species spread around the globe. Some of which are common to us and we see every day while some are unknown to us since either they are far-fetched, or have gone extinct. Humans are curious and we always quest the existence of anything we find interesting. So in this fast-paced world, it is difficult to go out in the wild and quickly become familiar with those dwelling there. So it’s better to always go equipped with a handful of knowledge that’s right there on your palm - of course, you would need some net!. That’s why we came up with the idea to build a web app that would help you in recognizing wildlife species - using the power of machine learning, that you encounter during your adventure and get familiar with them.

## Scope

CNNs can be used to identify different animal species based on their visual characteristics. This is valuable in biodiversity studies, wildlife monitoring, and ecological research.

CNNs can aid in the identification and monitoring of endangered species, assisting conservation efforts. They can be used to track animal populations, detect illegal wildlife trade, and identify potential threats to wildlife habitats.

CNNs can analyze animal behavior by recognizing and tracking specific actions or movements. This helps in studying animal interactions, migration patterns, and understanding ecological dynamics.

CNNs can automate the analysis of camera trap images, which are widely used for wildlife monitoring. They can detect and classify different animal species, enabling efficient data processing and reducing the need for manual inspection.

CNNs can assist veterinarians in diagnosing diseases and conditions in animals by analyzing medical images, such as radiographs or histopathological slides. This can aid in faster and more accurate diagnoses.

## Project Overview

Overview of project is to recognize animal species by using deep CNN. Since there are a large number of different animals, manually identifying them can be a difficult task. In this phase of work, the limitation of using data mining is researched and a new method is proposed which employs CNN called Lightweight Machine Convolution Network for Animal Recognition. It will have both good accuracy and minimal computational cost.

## Objectives

The Objective of the project is to identify different animal species using deep CNN. Because there are so many different kinds of animals, manually recognizing them may be difficult. Data mining limits are studied in this part of the research, and a unique technique termed. Lightweight Machine Convolution Network for Animal Recognition is devised. It will have low computational expenses and great precision.

CNN performs comparisons between the input data and the training dataset and predicts the output of the animal species with the accuracy percentage.

* 1. **Data Sets**









# LITERATURE SURVEY

There was a fine grained visual categorization, which tries to distinguish between objects of the same kind. With just the original image as its input, this novel description was able to automatically provide visual depictions that are distinct enough for precise visual categorization. Fine-grained visual categorization has a number of drawbacks, but its fundamental drawback is that it is computationally expensive and unsuited for large-scale picture classification.

They trained a CNN to classify 20 African wildlife species with an overall accuracy of 87.5% using a dataset that contained reportedly 111,467 images. The authors demonstrated the application of a gradient weighted class activation-mapping (Grad-CAM) procedure which wasused to extract the most salient pixels in the final convolution layer.

## Existing System

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

## Proposed System

Convolution neural network algorithm is a multilayer perceptron that is the special designfor the identification of two-dimensional image information. It has four layers: an input layer, a convolution layer, a sample layer, and an output layer. In a deep networkarchitecture, the convolution layer and sample layer may have multiple.

CNN is not as restricted as the Boltzmann machine, it needs to be before and after the layer of neurons in the adjacent layer for all connections, convolution neural network algorithms, each neuron doesn’t need to experience the global image, just feel the local region of the image.

In addition, each neuron parameter is set to the same, namely, the sharing of weights, namely each neuron with the same convolution kernels to the deconvolution image. The key era of CNN is the local receptive field, sharing of weights, sub-sampling by using time or space, with a purpose to extract features and reduce the size of the training parameters.

The advantage of CNN algorithm is to avoid the explicit feature extraction, and implicitlyto learn from the training data. The same neuron weights on the surface of the feature mapping, thus the network can learn parallel, and reduce the complexity of the network Adopting sub- sampling structure by time robustness, scale, and deformation displacement. Input information and network topology can be a very good match. It has unique advantages in image processing.

## Related Work

There have been many attempts to automatically identify animals in camera-trap images; however, most relied on manually designed features to detect animals while others used small datasets (few thousand images only for this application) Yu et al manually cropped and selected images, which only contained the entire animal body. This conditioning allowed them to obtain 82% accuracy by using linear support vector machine (SVM) to classify 18 animal species. They used their own dataset which consists of over 7,000camera- trap images from two different field sites. Several recent works used deep learning to classify camera-trap images. Chen et al. [12] used a deep convolutional neural network (CNN) to classify 20animal species in their own dataset of 20,000 images. The authors used an automatic segmentation algorithm (ensemble video object cut) for cropping the animals from the images and used these crops to train and to test their system. The convolution network which was used had 6 layers (3 convolutional layers and 3 max pooling layers) and it gave them a 38.31% accuracy. Gomez [13] used very deep CNNs to identify animal species in multiple versions of the Snapshot Serengeti dataset. This method reached 88.9% accuracy in the evaluation set. Norouzzadeh [14] used CNN and reported accuracies of 93.8% in classifying images that contain only a single animal. The performance matched human accuracy in their experiments. Most of the published results used the publicly available Snapshot Serengeti dataset which only contains African animals. There are more than one million sets of pictures, with each set containing three photographs. Before the release of the Snapshot Serengeti dataset, there was no publicly available and reliable dataset of animal photographs to work with. Our research differs from previous similar work as we aim to achieve high accuracy in object detection and animal species identification.

# SYSTEM ANALYSIS

## Functional Requirements

A training dataset has to be created on which training is performed. A testing dataset has to be created on which testing is performed.

## Performance Requirements Accuracy:

The primary performance requirement is achieving high accuracy in species recognition. The CNN models should be able to correctly classify and identify species with a high degree of precision.

## Speed and Efficiency:

In many real-time or resource-constrained applications, the speed and efficiency of species recognition are crucial. The CNN models should be optimized to provide fast and real-time predictions without compromising accuracy. This is particularly important for applications that require quick responses, such as wildlife monitoring or field studies.

## Generalization:

CNN models should demonstrate the ability to generalize well to new and unseen species. They should not only perform well on the training dataset but also exhibit good generalization on validation and test datasets that contain species instances not encountered during training. Generalization is crucial for the models to accurately identify species in real-world scenarios.

## Adaptability and Transfer Learning:

The CNN models should have the capability to adapt to new species or incorporate new knowledge over time. Transfer learning techniques can be employed to leverage pretrained models on related tasks or datasets and fine-tune them on specific species recognition tasks.

## Interpretability:

In certain applications, the interpretability of the CNN models is important. The ability to understand and explain the model's decision-making process can enhance trust and enable domain experts to validate the species identification. Interpretability can be achieved through techniques like visualizing activations, saliency maps, or attention mechanisms.

## Software Requirements

**Programming Language and Libraries:**

Choose a programming language that supports deep learning and has robust libraries for neural networks. Python is a popular choice due to its extensive ecosystem. You'll need libraries like TensorFlow, PyTorch, or Keras for implementing CNN architectures.

## Dataset:

Acquire or create a labeled dataset containing images of different species. This dataset will be used for training and evaluating the CNN model. Ensure the dataset is diverse, well- annotated, and covers a wide range of species.

## Image Processing Libraries:

Utilize libraries such as OpenCV or Pillow to handle image loading, preprocessing, and manipulation tasks. These libraries provide functions to resize, crop, normalize, and augment images to prepare them for training or inference.

## Hardware Requirements

**CPU:**

A powerful multi-core CPU is essential for data preprocessing, model training, and inference. Although deep learning heavily relies on GPU acceleration, a strong CPU is still necessary for overall system performance and handling non-GPU tasks.

## Storage:

Sufficient storage is required for storing your dataset, trained models, and intermediate checkpoints. Solid-State Drives (SSDs) are recommended for faster data access and model loading during training or inference.

## Memory:

Deep learning models can consume a significant amount of memory, especially when working with large datasets. It is advisable to have a generous amount of RAM to avoid memory limitations. A minimum of 16GB is recommended, but for more complex tasks or larger models, 32GB or higher is preferable.

## Cooling:

Deep learning tasks can generate substantial heat, particularly during long training sessions. Proper cooling, such as additional fans or liquid cooling systems, is necessary to maintain optimal temperatures and prevent overheating.

## Feasibility study (Technical/Economical/Operational) Technical Feasibility:

**Image Dataset Availability**:

Evaluate the availability and suitability of image datasets for training and testing your CNN models. Assess the size, quality, diversity, and labeling accuracy of the datasets. Determine if there are existing datasets or if you need to collect and label your own dataset.

## Model Architecture Selection:

Investigate different CNN architectures suitable for image recognition tasks. Consider well- established models such as VGGNet, ResNet, InceptionNet, or efficient models like MobileNet or EfficientNet. Assess their performance, complexity, and compatibility with the available hardware and software resources.

## Real-Time Inference:

Assess the feasibility of achieving real-time or near real-time inference on the desired hardware platform. Consider the model size, computational requirements, and latency constraints for inference in different deployment scenarios.

## Security and Privacy:

Identify potential security and privacy concerns related to the image recognition system. Assess the feasibility of implementing necessary security measures, such as data encryption, access control, or anonymization techniques, to protect sensitive data and ensure compliance with relevant regulations.

## Economical Feasibility:

**Cost Analysis:**

Determine the total costs associated with the project, including hardware and software expenses, data acquisition or labeling costs, personnel salaries, infrastructure costs, and any other relevant expenditures. Estimate both upfront costs and ongoing operational expenses.

## Return on Investment (ROI):

Evaluate the potential returns that the image recognition system can generate. Consider the market demand, potential customer base, and revenue generation opportunities. Estimate the revenue streams, such as licensing fees, subscription models, or service charges, and assess the expected ROI over a specific time period.

## Cost Savings or Benefits:

Identify potential cost savings or benefits that the image recognition system can bring to stakeholders or end-users. For example, if the system is intended for species identification in agriculture, it could help optimize crop management, reduce crop loss, or enhance productivity, leading to cost savings for farmers.

## Market Analysis:

Conduct a market analysis to understand the size of the target market and the potential demand for the image recognition system. Identify the market trends, growth potential, and competitive landscape. Evaluate the pricing strategy, pricing elasticity, and potential market penetration.

## Financial Risk Assessment:

Identify and analyze potential financial risks associated with the project. Consider factors such as market uncertainties, competitive challenges, technological advancements, and regulatory changes. Evaluate risk mitigation strategies and assess the project's resilience to potential risks.

## Operational Feasibility:

**User Acceptance:**

Assess the willingness and ability of the users (such as researchers, scientists, or enthusiasts) to adopt and use the image recognition system. Consider conducting surveys or user interviews to gather feedback and ensure that the system meets their requirements and expectations.

## Data Availability and Quality:

Assess the availability and quality of the required image datasets for training and testing the CNN models

## Operational Resources:

Evaluate the availability of the necessary resources to support the operational aspects of the system. This includes hardware resources and software resources required to run the image recognition system effectively.

# SYSTEM DESIGN

System design process involves identification and defining of components, modules, interfaces, data of a system and finally the architecture. The main goal of the system design is to satisfy the requirements specified the and design the system in such a way. In a sentence, it can be described as application of systemstheory to product development.

## System Architecture

**Data Collection and Storage:**

Collect a dataset of labeled images representing various species.

Store the dataset in a suitable data storage system, such as a database or file system.

## Preprocessing:

Perform preprocessing tasks on the input images, such as resizing, cropping, normalization, and augmentation.Apply any necessary transformations to align the images with the requirements of the CNN model.

## Convolutional Neural Network (CNN) Model:

Design the architecture of the CNN model specifically for species recognition.

The CNN typically consists of multiple convolutional layers, pooling layers, and fully connected layers.

Each layer performs operations like convolution, activation, pooling, and normalization to extract relevant features from the input images.

## Training:

Split the dataset into training, validation, and testing sets.

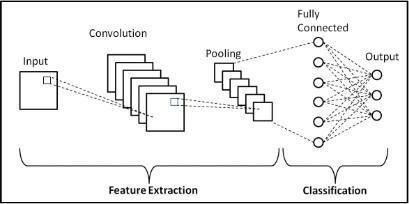
Train the CNN model using the training set, optimizing the model's parameters to minimize the loss function.

Use techniques like stochastic gradient descent and backpropagation to update the weights and biases of the model during training.

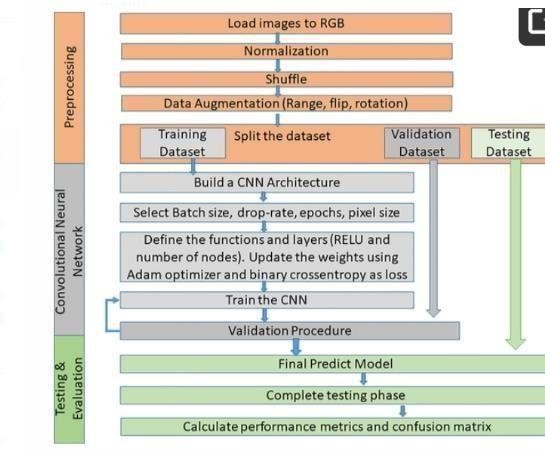
Monitor the model's performance on the validation set and adjust hyperparameters as needed.

## Model Evaluation:

Evaluate the trained CNN model's performance using the testing set. Measure metrics such asaccuracy, precision, recall, or F1-score to assess the model's ability torecognize species accurately.



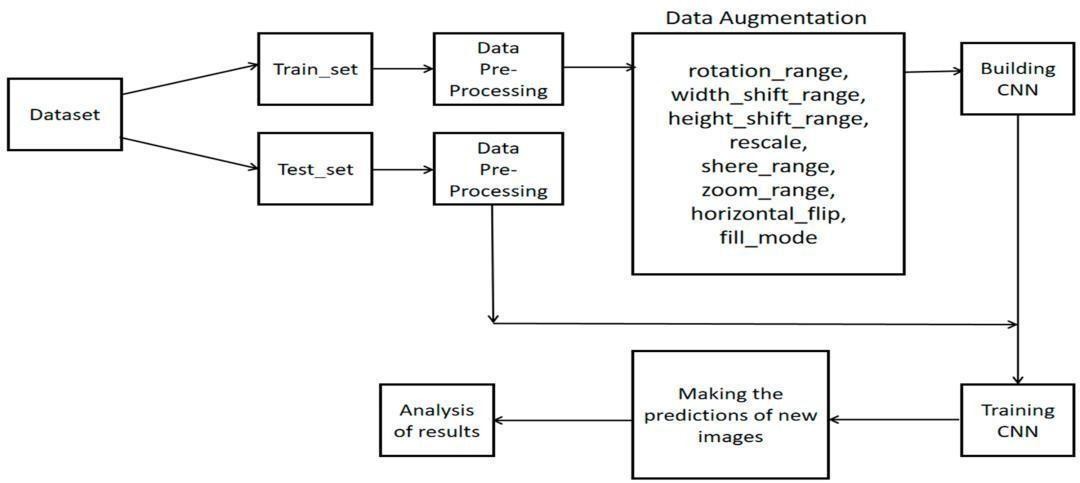
**Fig 4.1a System Architecture**



**Fig 4.1b System Architecture**

## Flow Chart Diagram

Data flow diagram depicts the flows of a data stream in a system and various information sources and their flows. Generally a data flow diagram is used to represent how the data is being processed or its flow, how data is stored, controlled and manipulated within many streams.



**Fig 4.2 Data Flow Diagram**

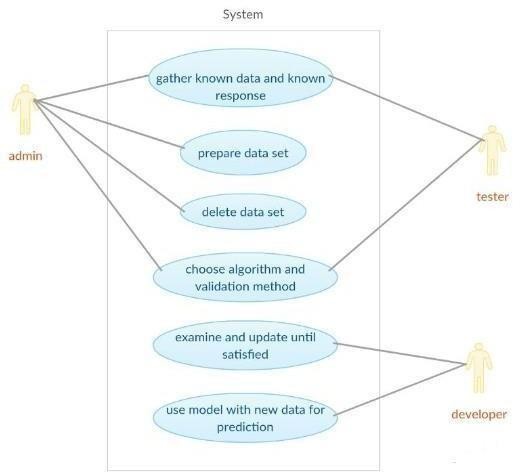
A data flow diagram (DFD) is a graphical representation that illustrates the flow of data within a system.

## UML Diagrams

A UML diagram is a partial graphical representation (view) of a model of a system under design, implementation, or already in existence. UML diagram contains graphical elements (symbols) - UML nodes connected with edges (also known as paths or flows) - that represent elements in the UML model of the designed system. The UML model of the system might also contain other documentation such as use cases written as templated texts.

## Use Case Diagram

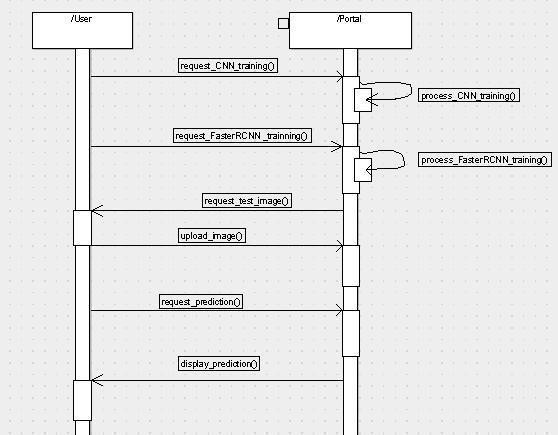
A use case diagram is a graphical depiction of a user’s possible interactions with the system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagram as well .For image recognition of species using cnn the elements are admin, testor and developer system and relationships. Process starts from the gathering of known data



**Fig 4.3.1 Use Case Diagram**

## Sequence Diagram

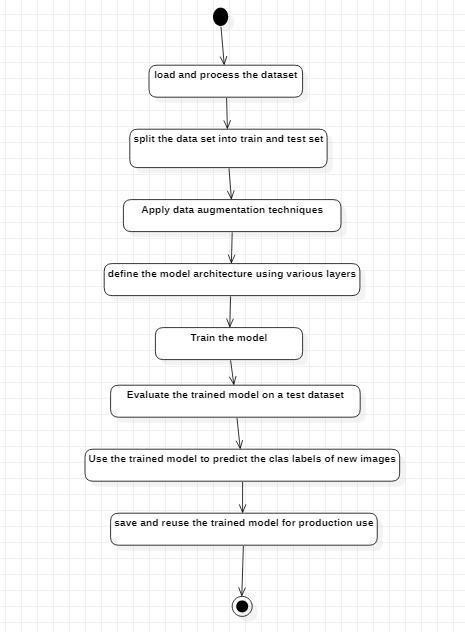
The sequence diagram shows the sequence of actions of the user, application, system based upon this sequence of actions, a sequence diagram had drawn. Messages are the most important elements of a sequence diagram. They indicate when one object calls an operation on another object (or itself). They are also used to indicate return values.



**Fig 4.3.2 Sequence Diagram**

## Activity Diagram

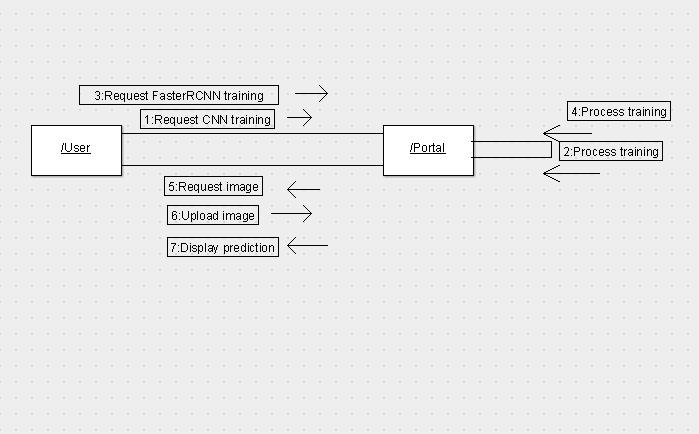
An activity diagram is a type of Unified Modeling Language (UML) flowchart that shows the flow from one activity to another in a system or process. It's used to describe the different dynamic aspects of a system and is referred to as a 'behavior diagram' because it describes what should happen in the modeled system



**Fig 4.3.3 Activity Diagram**

## Collaboration Diagram

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented.



**Fig 4.3.4 Collaboration Diagram**

## Modules

**Data Collection and Preprocessing Module:**

Responsible for collecting a diverse and well-labeled dataset of images representing differentspecies.

Performs preprocessing tasks such as resizing, cropping, normalization, and augmentation toprepare the images for training.

## CNN Model Architecture Module:

Defines the architecture of the Convolutional Neural Network (CNN) model used for speciesrecognition.

Specifies the structure and parameters of the CNN layers, including convolutional layers,pooling layers, and fully connected layers.

## Training Module:

Trains the CNN model using the labeled dataset, optimizing the model's parameters to minimize the loss function.

Utilizes techniques like stochastic gradient descent and backpropagation to update theweights and biases of the model during training.

## Model Evaluation Module:

Evaluates the performance of the trained CNN model using a separate testing dataset. Measures metrics such as accuracy, precision, recall, or F1-score to assess the model'seffectiveness in recognizing species.

## Inference Module:

Performs the inference or prediction step using the trained CNN model.

Accepts an input image and applies the trained model to generate predictions or probabilityscores for the species classes.

## Post-processing Module:

Applies post-processing techniques to interpret and refine the model's predictions.

## User Interface Module:

Provides a user-friendly interface for users to interact with the system.

Enables users to upload images, view recognition results, retrieve species information, andprovide feedback.

# IMPLEMENTATION AND RESULTS

## Language Used

Keras is a minimalist Python library for deep learning that can run on top of Theano or TensorFlow. It was developed to make implementing deep learning models as fast and easy as possible for research and development. It runs on Python 2.7 or 3.5 and can seamlessly execute on GPUs and CPUs given the underlying frameworks. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

Keras was developed and maintained by François Chollet, a Google engineer using fourguiding principles:

## Modularity:

A model can be understood as a sequence or a graph alone. All the concerns of a deep learningmodel are discrete components that can be combined in arbitrary ways.

## Minimalism:

The library provides just enough to achieve an outcome, no frills and maximizing readability.

## Extensibility:

New components are intentionally easy to add and use within the framework, intended forresearchers to trial and explore new ideas.

## Python:

No separate model files with custom file formats. Everything is native Python.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.

Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep- learning models on clusters of Graphics processing units (GPU) and tensor processing units

## TENSORFLOW

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that letsresearchers push the state- of- the-art in ML and developers easily build and deploy ML poweredapplications.

TensorFlow is a free and open source software for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. TensorFlow is Google Brain's second- generation system. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general- purpose computing on graphics processing units). TensorFlow is available on 64- bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors.

* 1. **Sample code** import numpy as np import tensorflow as tf

from matplotlib import pyplot as plt from tensorflow import keras

from tensorflow.keras.preprocessing.image import ImageDataGenerator from tensorflow.keras import layers

from tensorflow.keras.models import Model

from keras.layers import Dropout, Dense, GlobalAveragePooling2D from keras.applications.inception\_v3 import InceptionV3

from google.colab import drive drive.mount('/content/drive')

batch\_size=64

data\_dir = "/content/drive/MyDrive/Colab Notebooks/Mini\_Project/data/animals"

train\_datagen = ImageDataGenerator(rescale = 1./255, validation\_split=0.2,

rotation\_range=35, width\_shift\_range=0.25,

preprocessing\_function=tf.keras.applications.resnet.preprocess\_input, height\_shift\_range=0.25,

shear\_range=0.25, zoom\_range=0.25, horizontal\_flip=True, fill\_mode='nearest')

validation\_datagen = ImageDataGenerator(rescale = 1./255,validation\_split=0.2)

train\_generator = train\_datagen.flow\_from\_directory(data\_dir, target\_size=(299,299),

class\_mode='categorical', batch\_size=batch\_size, subset = "training")

validation\_generator = validation\_datagen.flow\_from\_directory(data\_dir, target\_size=(299,299),

class\_mode='categorical', batch\_size=batch\_size, subset = "validation")

labels = {v: k for k, v in train\_generator.class\_indices.items()}

pre\_trained\_model=InceptionV3(input\_shape=(299,299,3),include\_top=True,weights='imagenet') for layer in pre\_trained\_model.layers:

layer.trainable = False

last\_output = pre\_trained\_model.get\_layer('mixed10').output x = layers.Dense(1024, activation='relu')(last\_output)

x = layers.Dropout(0.2)(x)

x = layers.GlobalAveragePooling2D()(x)

x = layers.Dense(90, activation='softmax')(x) model = Model(pre\_trained\_model.input, x)

model.summary()

model.compile(optimizer='rmsprop', loss=tf.keras.losses.CategoricalCrossentropy(from\_logits=True), metrics=['accuracy'])

load\_model = True if load\_model:

model=keras.models.load\_model('/content/drive/MyDrive/Colab Notebooks/Mini\_Project/FinalModel.h5')

if not load\_model:

history = model.fit(train\_generator,

validation\_data=validation\_generator, epochs=10)

model.evaluate(validation\_generator) model.save('FinalModel.h5')

testing\_datagen = ImageDataGenerator(rescale = 1./255,validation\_split=0) testing\_data=testing\_datagen.flow\_from\_directory('/content/drive/MyDrive/Colab Notebooks/Mini\_Project/testing\_images',

target\_size=(299,299), class\_mode=None, batch\_size=1, subset="training")

np.argmax(model.predict(testing\_data)) labels[np.argmax(model.predict(testing\_data))]

!pip install gradio

import gradio as gr

from google.colab import drive from PIL import Image

import shutil import io

drive.mount('/content/gdrive')

def upload\_image(image):

pil\_image = Image.fromarray(image)

image\_path='/content/gdrive/MyDrive/Colab Notebooks/Mini\_Project/testing\_images/images/uploaded\_image.jpg'

pil\_image.save(image\_path)

testing\_datagen = ImageDataGenerator(rescale = 1./255,validation\_split=0)

testing\_data = testing\_datagen.flow\_from\_directory('/content/gdrive/MyDrive/Colab Notebooks/Mini\_Project/testing\_images',

target\_size=(299,299), class\_mode=None, batch\_size=1, subset="training")

output=labels[np.argmax(model.predict(testing\_data))] return output

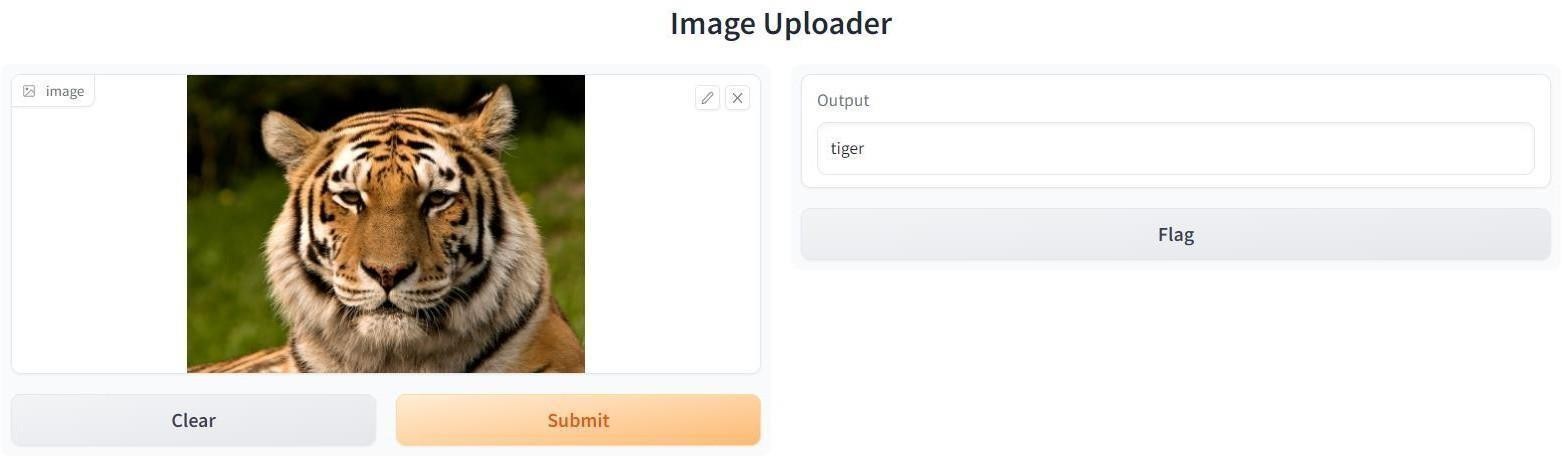
image\_upload = gr.inputs.Image() text\_output = gr.Textbox(label="Output")

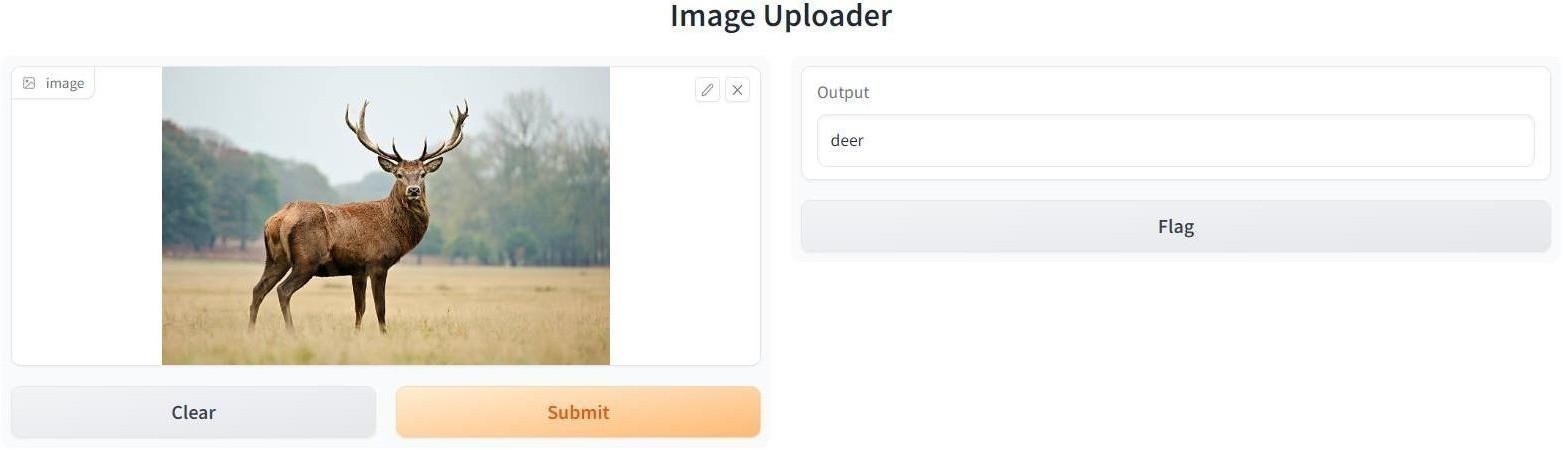
interface = gr.Interface(fn=upload\_image, inputs=image\_upload, outputs=text\_output, title='Image Uploader')

interface.launch(debug=True)

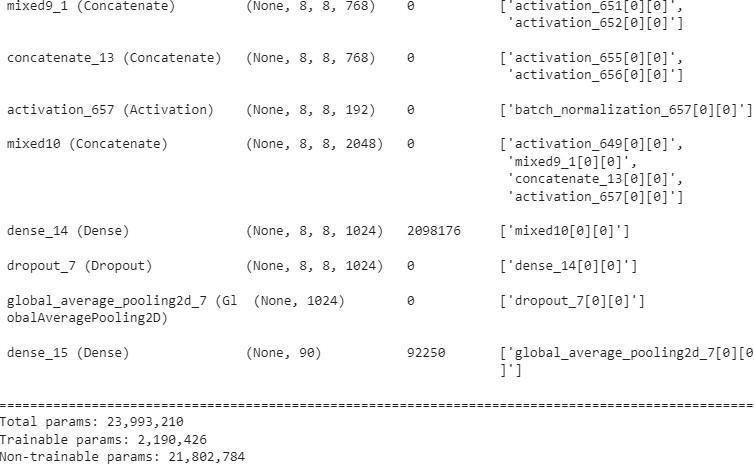
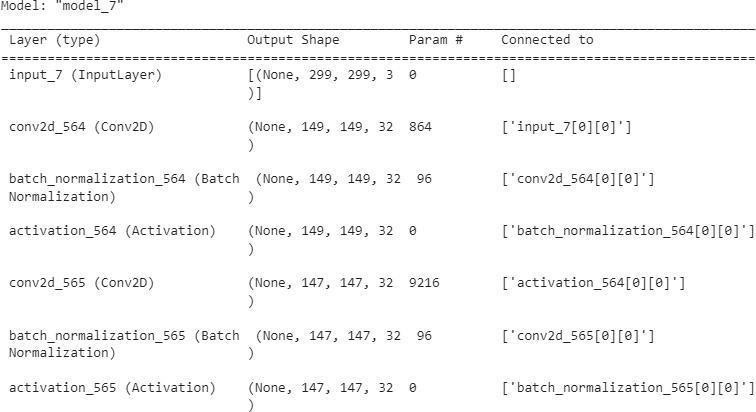
* 1. **Results/ Output Screens**







# TESTING



1. **CONCLUSIONS**

Animal recognition using Convolutional Neural Networks (CNNs) has emerged as a powerful and versatile tool in various fields such as zoology, ecology, conservation, and veterinary medicine. The application of CNNs in animal classification has demonstrated significant potential and continues to evolve.

CNNs have proven effective in species identification, facilitating biodiversity studies, wildlife monitoring, and ecological research. They contribute to wildlife conservation efforts by aiding in the identification and tracking of endangered species and detecting illegal wildlife trade. Furthermore, CNNs can analyze animal behavior, providing insights intoanimal interactions, migration patterns, and ecological dynamics.

The future scope of animal classification using CNNs looks promising. Advancements in fine- grained classification, behavior recognition, and multi-modal data integration will enable more detailed and comprehensive analyses. Real-time monitoring systems and collaborative citizen science initiatives involving CNNs will enhance conservation efforts and public participation.

Transfer learning, data augmentation, and automation of species monitoring programs will further advance the capabilities of CNNs in animal recognition. These developments have the potential to revolutionize wildlife conservation, ecological research, and veterinary medicine.

Overall, the use of CNNs in animal recognition holds tremendous promise in deepening our understanding of the animal kingdom, contributing to conservation efforts, and promoting sustainable coexistence with wildlife.

# FUTURE SCOPE

CNNs can be further developed to achieve more precise and detailed classification of animal species, including distinguishing between closely related species or identifying specific subspecies. This can provide deeper insights into biodiversity and evolutionary studies.

CNNs can be trained to recognize and classify complex animal behaviors, including social interactions, mating rituals, and feeding patterns. This can lead to a better understanding of animal behavior and contribute to fields such as ethology and ecology.

CNNs can be combined with other data sources, such as audio recordings or environmental sensors, to create multi-modal models for animal classification. This holistic approach can enhance accuracy and provide a more comprehensive understanding of animal populations and ecosystems.

CNNs can be deployed in real-time monitoring systems, enabling continuous surveillance and immediate detection of rare or endangered species. This can support conservation efforts, wildlife management, and early warning systems for environmental threats.

With the increasing availability of wildlife images and videos captured by citizen scientists, CNNs can be utilized to involve the public in animal classification projects. Crowdsourcing the identification process can accelerate data collection and contribute to larger-scale conservation initiatives.

Transfer learning techniques can be leveraged to adapt pre-trained CNN models to new animal species or specific ecological contexts. Additionally, data augmentation methods can be employed to generate synthetic training examples, augmenting the size and diversity of available datasets.

CNNs can play a crucial role in automating species monitoring programs, where cameras or drones continuously capture images or videos for analysis. This can significantly reduce manual effort and enable efficient and large-scale monitoring of wildlife populations.

# BIBLIOGRAPHY

* + 1. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems (pp. 1097-1105).
    2. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
    3. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).
    4. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
    5. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Berg, A. C. (2015). ImageNet large scale visual recognition challenge. International Journal of Computer Vision, 115(3), 211-252.
    6. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large- scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255).
    7. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In European conference on computer vision (pp. 21- 37).
    8. Russakovsky, O., Zhao, J., Deng, J., & Fei-Fei, L. (2015). Detecting avocados to zucchinis: What have we done, and where are we going?. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 1-8).
    9. Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).
    10. Kussul, N., Lavreniuk, M., Skakun, S., & Shelestov, A. (2017). Deep learning classification of land cover and crop types using remote sensing data. IEEE Geoscience and Remote Sensing Letters, 14(5), 778-782.