ANIMAL DETECTION USING DEEP LEARNING

A MINI PROJECT REPORT

Submitted by

LOKESH V (710020104017)

PERUMAL RAJ A (710020104021)

VIGNESH V (710020104035)

SANTHOSH KUMAR B(710020104315)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY REGIONAL CAMPUS

COIMBATORE – 641046

ANNA UNIVERSITY : CHENNAI 600 025

JUNE 2023

BONAFIDE CERTIFICATE

Certificate that this project report "**ANIMAL DETECTION USING DEEP LEARNING"** is the bonafide work of "**PERUMAL RAJ A (710020104021), LOKESH V (710020104017), VIGNESH V (710020104035), SANTHOSH KUMAR B (710020104323)**" who carried out the project work under my supervision.

SIGNATURE

Dr. P. MARIKKANNU

M. Tech., Ph.D

HEAD OF THE DEPARTMENT

Department of Computer Science and Engineering,

Anna University Regional Campus Coimbatore.

SIGNATURE

Dr. P. MARIKKANNU

M. Tech., Ph.D

SUPERVISOR

Department of Computer Science and Engineering,

Anna University Regional Campus Coimbatore.

Submitted for the project viva voce examination held at ANNA UNIVERSITY REGIONAL CAMPUS COIMBATORE held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

Animal detection plays a crucial role in various domains such as wildlife conservation, agricultural monitoring, and autonomous vehicles. With the advancements in deep learning techniques, the application of deep neural networks for animal detection has gained significant attention. This abstract presents an overview of the research conducted on animal detection using deep learning methods.

The proposed approach leverages deep convolutional neural networks (CNNs) to automatically learn and extract discriminative features from visual data, enabling accurate identification and localization of animals in images or video streams. The process involves several key steps, including data collection, preprocessing, model training, and inference.

To train the animal detection model, a large-scale annotated dataset containing diverse species and backgrounds is compiled. The dataset is carefully curated, ensuring a balanced representation of various animal classes and encompassing variations in pose, lighting conditions, and occlusions. Preprocessing techniques such as image augmentation and normalization are applied to enhance the model's robustness and generalization capabilities.

Experimental results demonstrate that the proposed deep learning-based animal detection system achieves high accuracy and robustness across diverse datasets and real-world scenarios. The system showcases potential applications in wildlife monitoring, animal behavior analysis, and environmental conservation efforts.

TABLE OF CONTENT

CHAPTER NO	EXPERIMENT	PAGE NO
1	Introduction	1
	1.1 General Introduction	1
	1.2 Scope and Objective	2
	1.3 Definition of Terms	4
	1.4 Literature Survey	5
	1.4.1 Introduction	5
	1.4.2 Literature	5
	1.4.3 Summary of Literature Survey	6
2	System Methodology	7
	2.1 Problem Statement	7
	2.2 Existing system	8
	2.3 Proposed System	9
	2.4 Technologies used	11
	2.4.1 YOLO V8	11
	2.4.2 Tools used	12
	2.5 Algorithm	13
	2.6 Modules	14
	2.6.1 Dataset Collection	14
	2.6.2 Annotation Module	14
	2.6.3 Training and Testing	14
3	UML Diagram	15

	3.1 Use case Diagram	15
	3.2 Class Diagram	16
	3.3 Data Flow Diagram	17
	3.4 Architecture Diagram	18
	3.5 Sequence Diagram	19
	3.6 Activity Diagram	20
	3.7 Collaboration Diagram	21
4	Implementation and Results	22
	4.1 Implementation	22
	4.2 Results	27
5	Conclusion and Future work	30
	5.1 Conclusion	30
	5.2 Future Work	31

LISTS OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
3.1	Use case Diagram	16
3.2	Class Diagram	17
3.3	Data Flow Diagram	18
3.4	Architecture Diagram	19
3.5	Sequence Diagram	20
3.6	Activity Diagram	21
3.7	Collaboration Diagram	22
4.1	Detection output 1	27
4.2	Detected output 2	28
4.3	Alert Message Output	29

CHAPTER 1
INTRODUCTION

1.1 GENERAL INTRODUCTION

Animal detection is a fundamental task with wide-ranging applications in fields such as wildlife conservation, ecological research, precision agriculture, and autonomous vehicles. The ability to accurately identify and localize animals in images or video streams is crucial for understanding animal behavior, monitoring populations, mitigating human-wildlife conflicts, and assessing environmental impacts. Traditional methods for animal detection heavily rely on manual annotation and handcrafted features, which can be laborious, time-consuming, and limited in their ability to handle variations in animal appearance and environmental conditions.

Deep learning, a subset of machine learning, has revolutionized computer vision tasks by leveraging deep neural networks to automatically learn hierarchical representations from raw data. In recent years, deep learning techniques have been successfully applied to animal detection tasks, offering promising results in terms of accuracy, scalability, and adaptability. By training deep neural networks on large annotated datasets, these models can learn to recognize complex patterns and discriminate between different animal classes.

The key advantage of deep learning in animal detection lies in its ability to extract high-level features and spatial relationships directly from raw visual data. Convolutional neural networks (CNNs), a prevalent deep learning architecture for image processing tasks, excel at learning local and global visual features through hierarchical convolutions and pooling operations.

The process of animal detection using deep learning typically involves several stages. First, a diverse and representative dataset of annotated images is collected, covering various animal species, poses, environmental conditions, and imaging modalities.

1.2 SCOPE AND OBJECTIVE

The scope for animal detection using deep learning is vast and encompasses various domains and applications. Here are some key areas where deep learning-based animal detection can make a significant impact.

Wildlife Conservation: Deep learning-based animal detection can aid in wildlife conservation efforts by automatically monitoring and tracking animal populations in their natural habitats. This information can help assess population dynamics, identify endangered species, and detect poaching or illegal activities.

Ecological Research: Deep learning enables researchers to analyze large-scale ecological datasets, including camera trap images or aerial surveys, to study animal behaviors, migration patterns, habitat preferences, and interactions with their environment. Automated animal detection can provide valuable insights into ecosystem dynamics and support conservation planning.

Precision Agriculture: Animal detection can be applied in agricultural settings to monitor and manage wildlife interactions with crops and livestock. Deep learning models can detect and classify animals that pose a threat to agricultural production, such as pests, predators, or invasive species, enabling targeted intervention and reducing crop damage or livestock losses.

Animal Behavior Analysis: Deep learning-based animal detection can facilitate the study of animal behavior by automatically detecting and tracking individuals within a group. This technology enables researchers to quantify social interactions, movement patterns, foraging behavior, and other behavioral characteristics, contributing to a better understanding of animal social dynamics and cognitive abilities.

Accurate Animal Detection: The main objective is to achieve high accuracy in animal detection, ensuring that the deep learning models can correctly identify the presence and location of animals within visual data. This involves training models that can handle variations in animal appearance, environmental conditions, occlusions, and complex backgrounds.

Species Classification: In addition to detecting animals, the objective is to classify them into specific species or taxonomic categories. Deep learning models should be capable of learning discriminative features that enable accurate species classification, contributing to biodiversity monitoring, conservation efforts, and ecological research.

Real-time and Efficient Detection: Another objective is to develop deep learning models that can perform animal detection in real-time or near real-time. This includes optimizing the model's architecture, inference speed, and computational efficiency to enable timely processing and analysis of large volumes of visual data.

Robustness to Environmental Variations: Deep learning models for animal detection should be robust to various environmental conditions, including changes in lighting, weather, and seasonal variations. Ensuring the models' resilience to these variations enhances their applicability in real-world scenarios and improves the reliability of detection results.

Scalability and Generalization: The objective is to develop scalable deep learning frameworks that can handle large-scale datasets and generalize well to unseen data. This involves designing architectures and training methodologies that can effectively leverage diverse training data and adapt to different animal species, habitats, and imaging modalities.

1.3 DEFINITION OF TERMS

Animal Detection: The process of automatically identifying and localizing animals in images or video streams. It involves using computer vision techniques, such as deep learning algorithms, to analyze visual data and detect the presence of animals, typically represented as bounding boxes or pixel-level annotations.

Deep Learning: A subset of machine learning that utilizes deep neural networks to learn hierarchical representations from raw data. Deep learning models consist of multiple layers of interconnected artificial neurons, allowing them to automatically extract complex features and patterns directly from input data.

Convolutional Neural Networks (CNNs): A type of deep neural network specifically designed for image processing tasks. CNNs employ convolutional layers to scan and extract local features from images, capturing spatial relationships and patterns. They are commonly used in animal detection to learn discriminative features from visual data.

Supervised Learning: A learning approach in which a model is trained on labeled data, where input samples are paired with corresponding output labels or annotations. In the context of animal detection, supervised learning involves training deep learning models using annotated images or video frames, where the animal presence and location are known.

Training: The process of optimizing the parameters of a deep learning model by iteratively adjusting them to minimize the difference between predicted outputs and ground-truth labels. During training, the model learns to recognize and differentiate animals from other objects or backgrounds, improving its accuracy and performance.

1.4 LITERATURE SURVEY

1.4.1 INTRODUCTION

The literature survey on animal detection using deep learning provides an overview of the research and advancements in the field. It encompasses studies that have explored the application of deep learning techniques, such as convolutional neural networks (CNNs), for the detection and localization of animals in images or video data.

The primary focus of the literature survey is to examine the use of deep learning models in various contexts related to animal detection, including camera trap images, UAV imagery, thermal images, and autonomous vehicle settings. The surveyed studies aim to address the challenges associated with accurately detecting and identifying animal species, mitigating false positives and false negatives, achieving real-time processing capabilities, and improving overall performance and efficiency.

1.4.2 LITERATURE

"Deep learning for wildlife species detection and localization in camera trap images" by Beery et al. (2018)
This study focuses on using deep learning techniques, specifically convolutional neural networks (CNNs), for wildlife species detection and localization in camera trap images. The authors propose an approach that combines multiple CNN architectures and demonstrates its effectiveness in accurately identifying and localizing various animal species.

"Automatic animal detection in UAV images using deep learning" by Norouzzadeh et al. (2018)
The researchers present a deep learning-based method for automatic animal detection in aerial images captured by unmanned aerial vehicles (UAVs). They employ a Faster R-CNN architecture to detect and classify animals, achieving high accuracy in identifying species such as elephants, giraffes, and cattle.

"Deep multi-modal animal detection and species classification in camera trap images" by Tabak et al. (2018)

This paper proposes a deep learning framework that combines both visual and acoustic information for animal detection and species classification in camera trap images. The authors leverage CNNs and convolutional recurrent neural networks (CRNNs) to process both visual and audio features, achieving improved performance compared to using only visual data.

1.4.3 SUMMARY OF LITERATURE SURVEY

The literature survey on animal detection using deep learning explores the application of deep learning techniques, primarily convolutional neural networks (CNNs), in detecting and localizing animals in images and video data. The survey encompasses studies that focus on various contexts, including camera trap images, UAV imagery, thermal images, and autonomous vehicle settings.

The surveyed literature highlights the potential of deep learning techniques in revolutionizing animal detection by providing accurate, efficient, and scalable solutions. By automating the detection process, deep learning models can contribute to wildlife conservation efforts, enhance ecological research, support precision agriculture practices, and aid in mitigating human-wildlife conflicts.

The surveyed studies also emphasize the importance of addressing challenges such as false positives and false negatives, adaptability to different environments and species, and the integration of deep learning models into existing systems and platforms.

Overall, the literature survey underscores the significance of deep learning in animal detection and its potential to advance scientific understanding, promote conservation, and improve the management of animal populations.

CHAPTER 2

SYSTEM METHODOLOGY

2.1 PROBLEM STATEMENT

The problem statement for Animal Detection Using Deep Learning is to develop an accurate and efficient system capable of automatically identifying and classifying animals in images or videos. Existing methods for animal detection often struggle with complex backgrounds, varying lighting conditions, and diverse animal species. The proposed solution aims to leverage deep learning techniques, such as convolutional neural networks (CNNs), to train a model capable of detecting and recognizing animals with high precision and recall. The system should be scalable, adaptable to different environments, and capable of handling real-time processing to support applications like wildlife conservation, monitoring, and surveillance.

The primary objective is to design a robust model that can accurately identify and localize animals in images or video streams, enabling automated and real-time monitoring. The system should be able to handle diverse environmental conditions, various animal species, and different camera perspectives.

Key challenges to address include the limited availability of labeled animal datasets, the potential occlusion of animals by vegetation or other objects, and the variability in animal appearance due to factors such as lighting conditions and camouflage.

By overcoming these challenges, the proposed deep learning-based animal detection system will provide a valuable tool for wildlife conservation, improved transportation safety, and enhanced agricultural protection, facilitating timely and efficient decision-making processes.

2.2 EXISTING SYSTEM

Animal detection using Convolutional Neural Networks (CNNs) has emerged as a promising technology for automated and accurate identification of animals in images or video streams. There exist several systems for animal detection using CNN have been proposed by researchers. Some example are

- AnimalVision

  AnimalVision is an existing CNN-based system for automated animal detection. It leverages deep learning techniques to accurately identify animals in images or video streams. AnimalVision aims to assist researchers and conservationists in wildlife monitoring and ecological research, providing valuable insights into animal populations and behaviors.

- ZooGuard

  ZooGuard is an intelligent surveillance system that employs Convolutional Neural Networks (CNNs) for real-time animal detection and monitoring within zoo environments. The system aims to enhance the security and well-being of animals by automatically detecting their presence and behavior.

- EcoTrack

  EcoTrack is an intelligent wildlife monitoring system that utilizes Convolutional Neural Networks (CNNs) for animal detection and tracking in natural habitats. The system aims to assist ecologists, researchers, and conservationists in studying and monitoring wildlife populations and behaviors.

- WildWatch

  WildWatch is an existing system that leverages Convolutional Neural Networks (CNNs) for animal detection and tracking in wildlife conservation and research. The system combines computer vision techniques with machine learning algorithms to automatically identify and monitor animal species in their natural habitats.

2.3 PROPOSED SYSTEM

The aim of the proposed system is to develop a system of improved facilities. The proposed system can overcome the limitations of existing systems. The proposed systems could incorporate

- Data Acquisition

  The system supports various data acquisition methods, including static images or real-time video feeds from cameras or drones deployed in the target area.

- Preprocessing

  The system applies preprocessing techniques such as resizing, normalization, and data augmentation to enhance the quality and diversity of the dataset, improving the CNN model's performance.

- CNN Model Development

  The system develops a customized CNN model architecture suitable for animal detection. The model undergoes training on a labeled dataset, utilizing optimization algorithms and appropriate loss functions to learn meaningful features for accurate classification.

- Training and Fine-tuning

  The system trains the CNN model using the labeled dataset, iteratively updating the model's parameters to minimize the training loss. Fine-tuning techniques may be applied to adapt the model to specific animal species or environmental conditions.

- Inference and Detection

  Once trained, the CNN model is utilized for inference on new images or video frames. The system processes the data through the model to detect animals present in the scene. Bounding boxes are drawn around each detected animal to indicate their locations.

- Tracking and Behavior Analysis

The system employs tracking algorithms to link animal detections across frames, enabling the tracking of individual animals over time. The system also incorporates behavior analysis techniques to recognize specific behaviors and gather insights into animal interactions and habitat usage.

- User Interface and Visualization

  The system provides a user-friendly interface for users to interact with the system. It allows users to input images or video streams, visualize the detected animals, adjust parameters if necessary, and generate reports or visual outputs for further analysis.

2.4 TECHNOLOGIES USED

2.4.1 YOLOv8:

YOLOv8 is the latest iteration in the YOLO series of real-time object detectors, offering cutting-edge performance in terms of accuracy and speed. Building upon the advancements of previous YOLO versions, YOLOv8 introduces new features and optimizations that make it an ideal choice for various object detection tasks in a wide range of applications.

Key Features of YOLOv8:

1. **Improved Accuracy:** YOLOv8 improves object detection accuracy compared to its predecessors by incorporating new techniques and optimizations.
2. **Enhanced Speed:** YOLOv8 achieves faster inference speeds than other object detection models while maintaining high accuracy.
3. **Multiple Backbones:** YOLOv8 supports various backbones, such as EfficientNet, ResNet, and CSPDarknet, giving users the flexibility to choose the best model for their specific use case.
4. **Adaptive Training:** YOLOv8 uses adaptive training to optimize the learning rate and balance the loss function during training, leading to better model performance.
5. **Advanced-Data Augmentation:** YOLOv8 employs advanced data augmentation techniques such as MixUp and CutMix to improve the robustness and generalization of the model.
6. **Customizable Architecture**: YOLOv8's architecture is highly customizable, allowing users to easily modify the model's structure and parameters to suit their needs.
7. **Pre-Trained Models:** YOLOv8 provides pre-trained models for easy use and transfer learning on various datasets.

2.4.2 TOOLS USED:

Google Colab:

Google Colab is a hosted Jupyter Notebook service that requires no setup to use and provides free access to computing resources, including GPUs and TPUs. Colab is especially well suited to machine learning, data science, and education[1]. You can import an image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code. You can choose the Colab plan that's right for you. Whether you're a student, a hobbyist, or a ML researcher, Colab has you covered. Colab is always free of charge to use.

Ultralytics:

Install YOLOv8 via the ultralytics pip package for the latest stable release or by cloning the https://github.com/ultralytics/ultralytics repository for the most up-to-date version.

YAML File:

YAML is a data serialization language that allows you to store complex data in a compact and readable format[1]. YAML files use a .yml or .yaml extension[1]. The structure of a YAML file is a map or a list[1]. Maps allow you to associate key-value pairs. Each key must be unique, and the order doesn't matter[1]. A map in YAML needs to be resolved before it can be closed, and a new map is created. Its syntax is independent of a specific programming language.

LabelImg:

It is an open-source graphical image annotation tool that is written in Python and uses Qt for its graphical interface. It is used to label images and generate the annotations in the format of Pascal VOC or YOLO. You can use it to draw bounding boxes around objects of interest in images and save the labels as XML files. It is available on Windows, Linux, and macOS.

2.5 ALGORITHM:

INPUT: Image or a Video frame containing an animal.

OUTPUT: Animal detection and Alert message in telegram.

IMPLEMENTATION:

Step 1: Installing the required libraries and dependencies.

Step 2: Download the YOLOv8 model. In this as three model type , there ` areYOLOn8,YOLOm8andYOLOl8.

Step 3: Load the YOLOv8 model.

Step 4: Load the image or video frame containing an animal.

Step 5: Run the image or video frame through the YOLOv8 model.

Step 6: Get the output of the YOLOv8 model alert message of the animal in the image or video frame and the class of the animal.

2.6 MODULES

2.6.1 Dataset Collection:

Kaggle is a platform for data science and machine learning enthusiasts. It provides a community of data scientists and machine learning engineers to share datasets, code, and ideas. Kaggle hosts many datasets that can be used for machine learning and deep learning projects such as animal detection.

2.6.2 Annotation Module:

To use YOLOv8 for animal detection, you need to annotate the images in the dataset with bounding boxes around the animals. You can use tools such as LabelImg or RectLabel for image annotation. Once you have the annotated images, you can train the YOLOv8 model on the dataset using a deep learning framework such as TensorFlow or PyTorch.

2.6.3 Training and Testing:

To train and test the YOLOv8 model on your annotated animal detection dataset, you can use the train mode of the YOLOv8 model. In this mode, you can specify the dataset and hyperparameters for training the model. Once you have trained the model, you can use it for testing on new images, so In this process

Training data taken 75% and reminding 25% used for the validation process.

CHAPTER 3

UML DIAGRAM

3.1 USE CASE DIAGRAM

A use case diagram is a visual representation of the interactions between actors (users or systems) and the system itself. It depicts the functional requirements of a system by illustrating the different use cases or functionalities it provides.

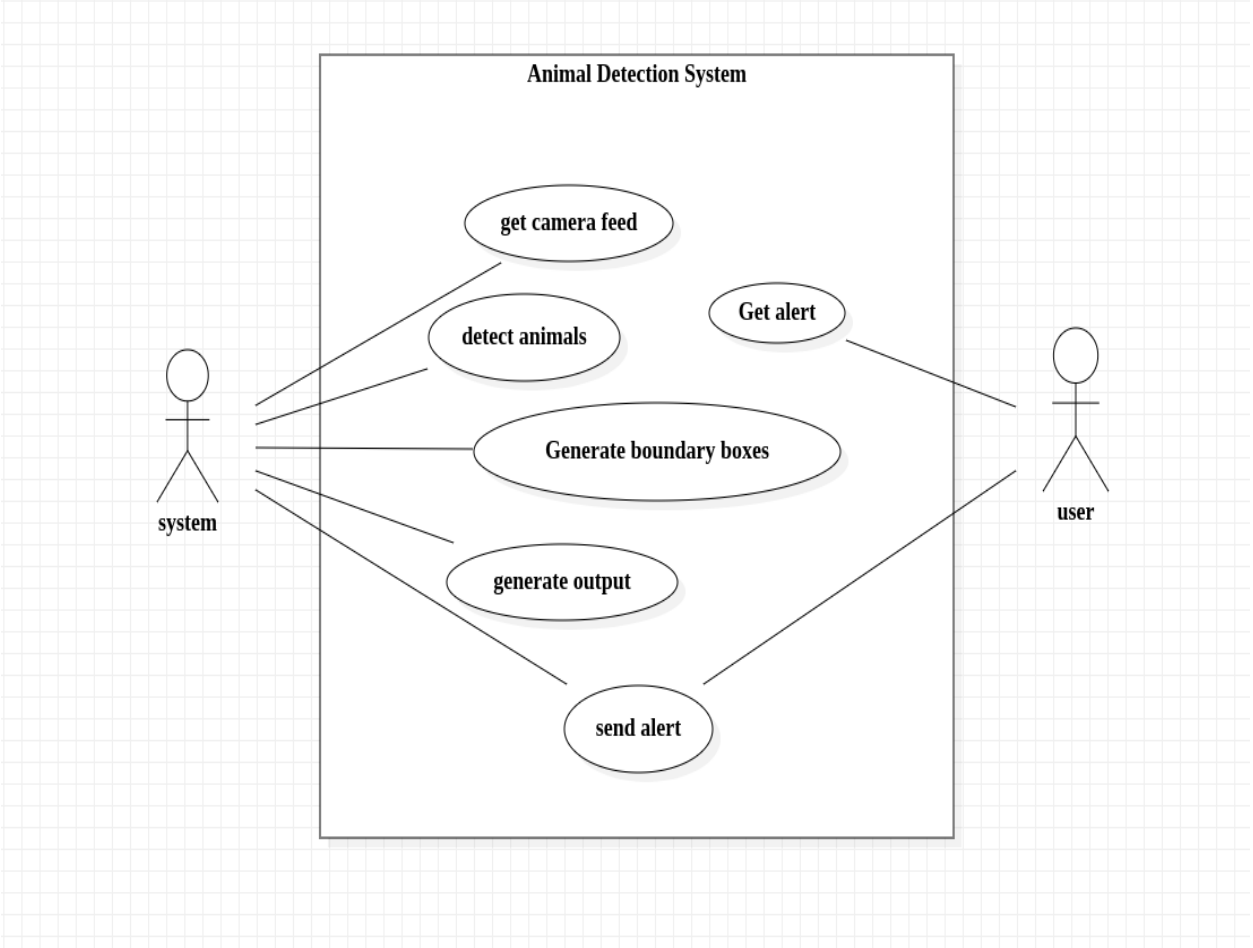

Fig 3.1 Use Case Diagram

3.2 CLASS DIAGRAM

A class diagram is a type of UML (Unified Modeling Language) diagram that represents the structure and relationships of classes in an object-oriented system. It provides a visual representation of the classes, their attributes, methods, and associations, allowing developers to understand the overall design of a system.

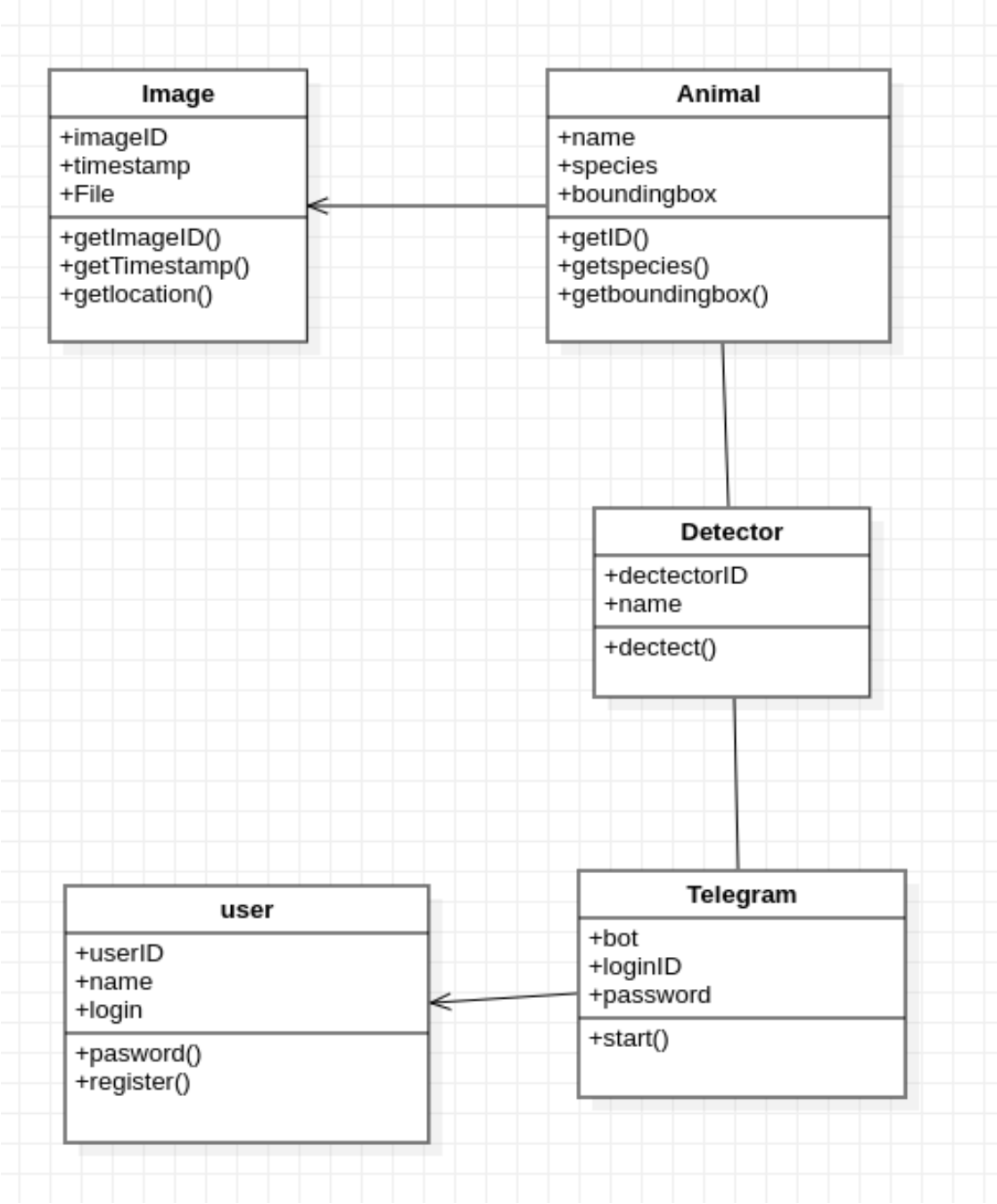

Fig 3.2 Class Diagram

3.3 DATA FLOW DIAGRAM

A dataflow diagram is a graphical representation that illustrates the flow of information within a system or process. It depicts how data moves through different stages or components, showing the inputs, outputs, and transformations that occur along the way.

3.3.1 LEVEL 0

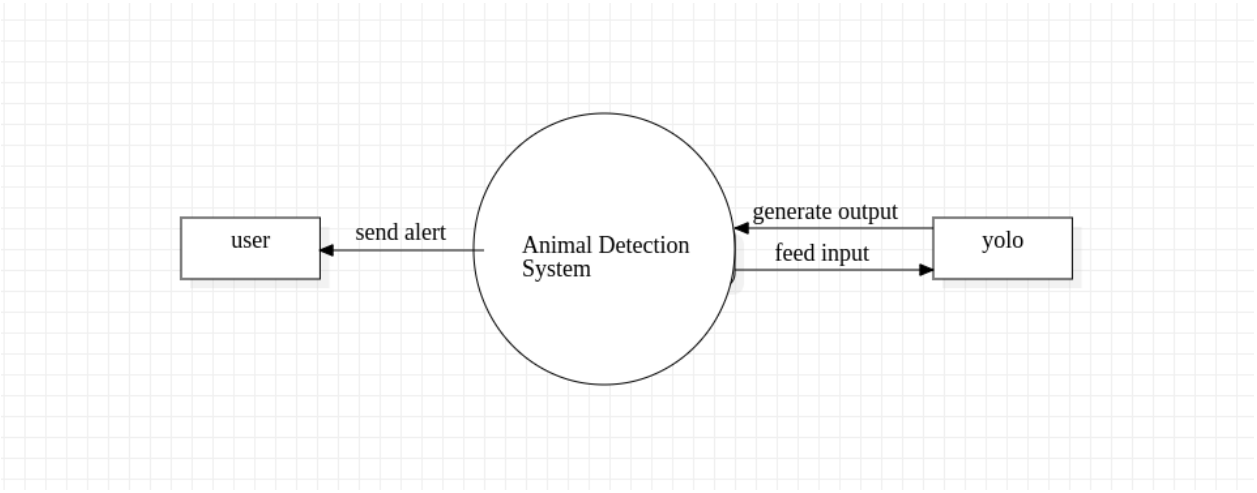

Fig 3.3.1 DFD Level 0

3.3.2 LEVEL 1

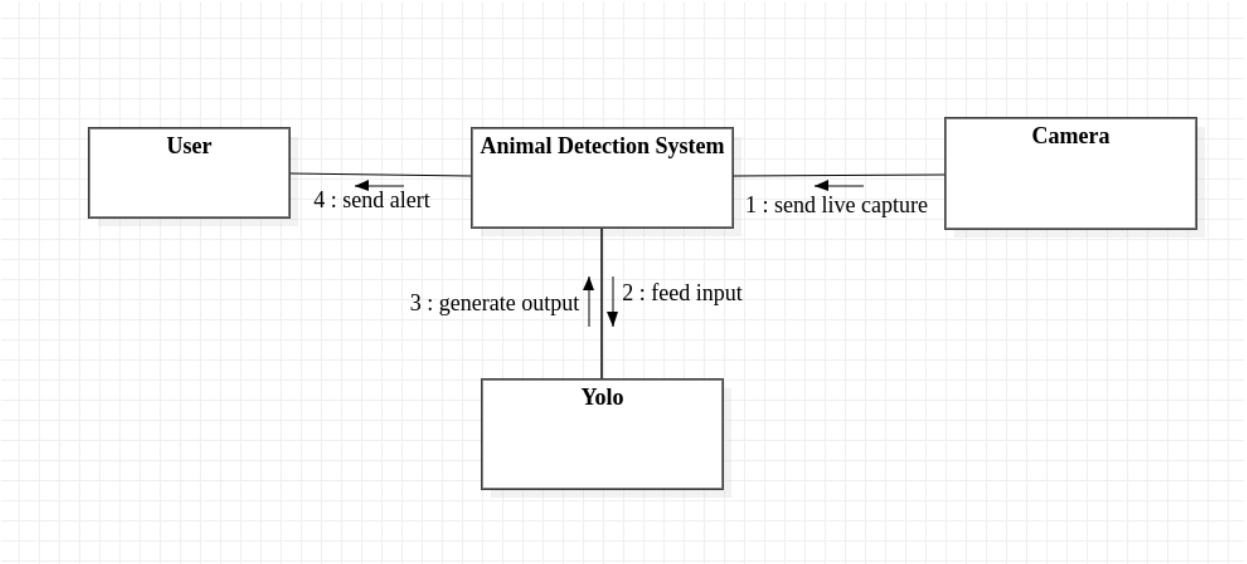

Fig 3.3.2 DFD Level 1

3.4 ARCHITECTURE DIAGRAM

Architecture refers to the design and organization of structures, buildings, and spaces that serve various purposes. It encompasses the art and science of creating functional and aesthetically pleasing environments for human use. An architectural diagram is a visual representation of the design and composition of a structure or system.

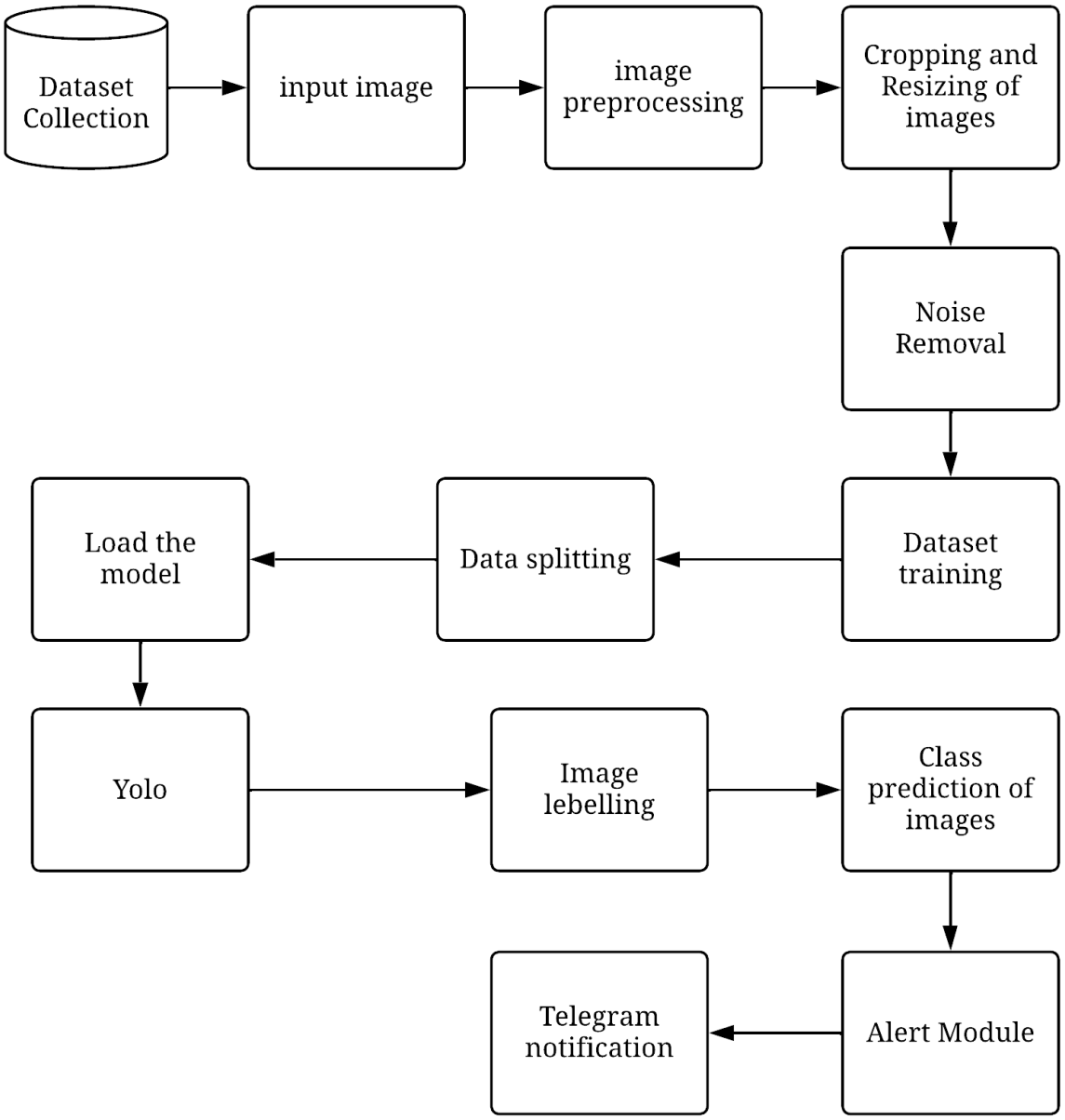

Fig 3.4 Architecture Diagram

3.5 SEQUENCE DIAGRAM

A sequence diagram is a type of interaction diagram in UML (Unified Modeling Language) that represents the flow of messages or interactions between objects or components within a system. It provides a visual representation of the dynamic behavior of a system, showing how objects collaborate and exchange information over time.

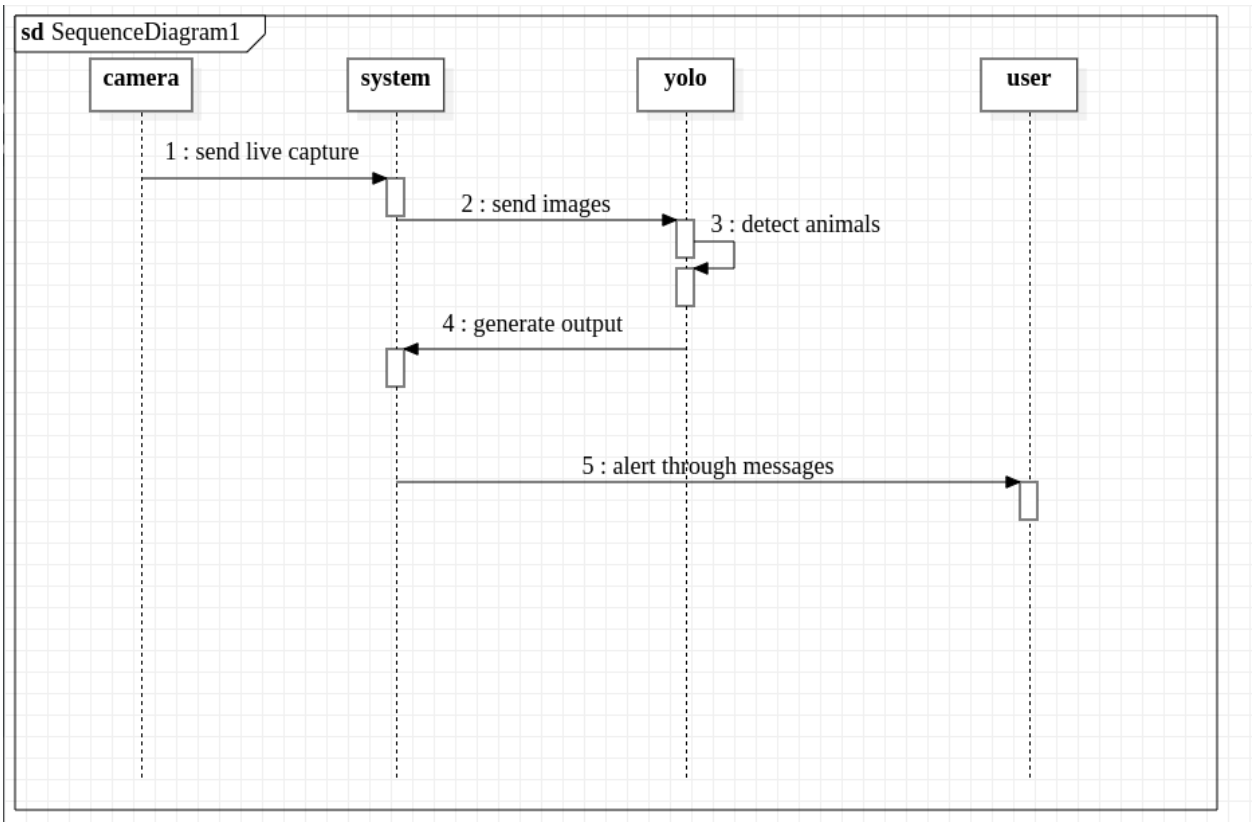

Fig 3.5 Sequence Diagram

3.6 ACTIVITY DIAGRAM

An activity diagram is a graphical representation of the flow of activities or processes within a system, software application, or business process. It is a part of the Unified Modeling Language (UML) and is used to model the behavior of a system.

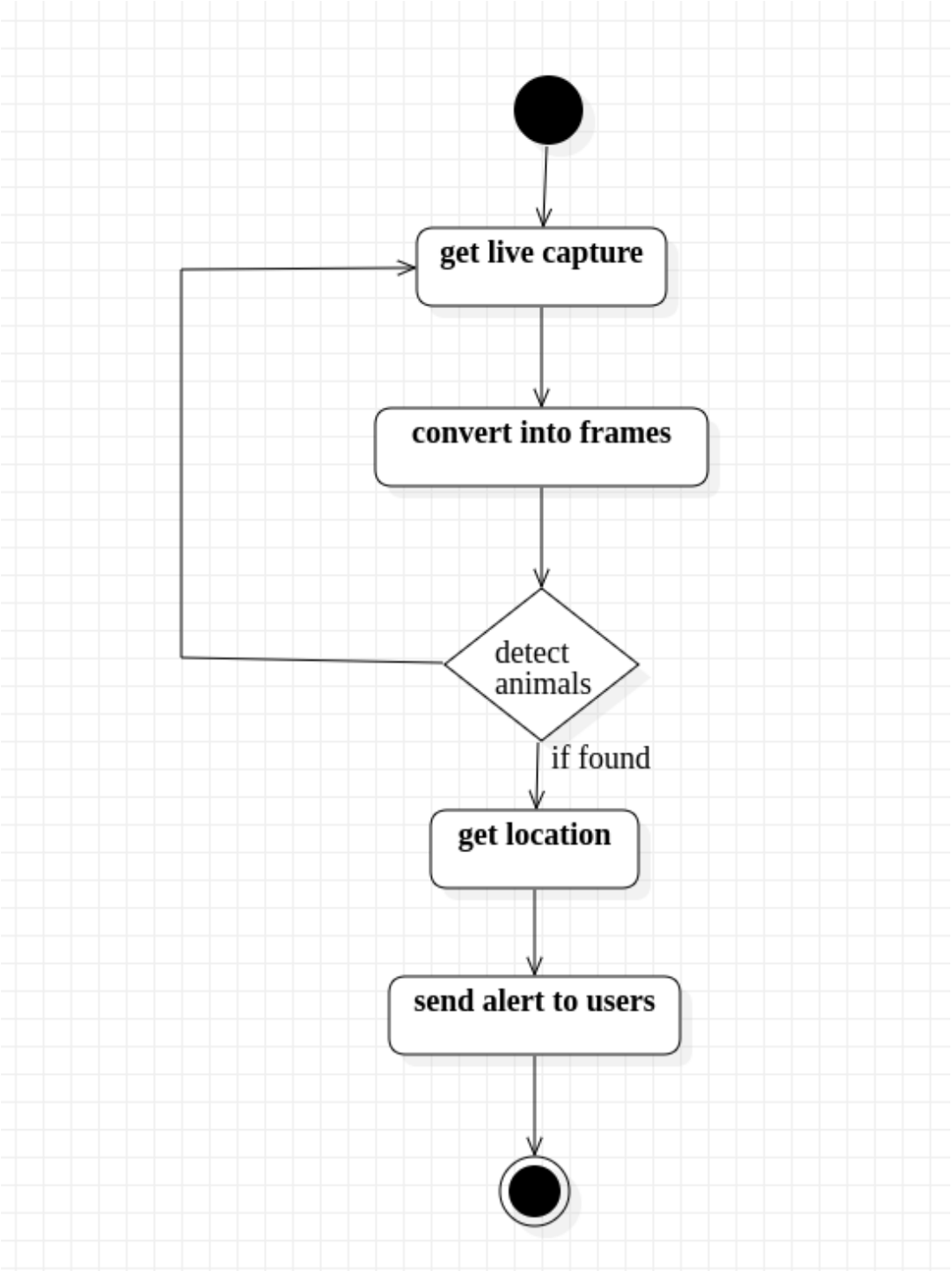

Fig 3.6 Activity Diagram

3.7 COLLABORATION DIAGRAM

A collaboration diagram, also known as a communication diagram, is a type of behavioral diagram in UML (Unified Modeling Language) that illustrates the interactions and relationships between objects or components within a system. It provides a visual representation of how different objects or components collaborate and communicate with each other to achieve a specific functionality or behavior.

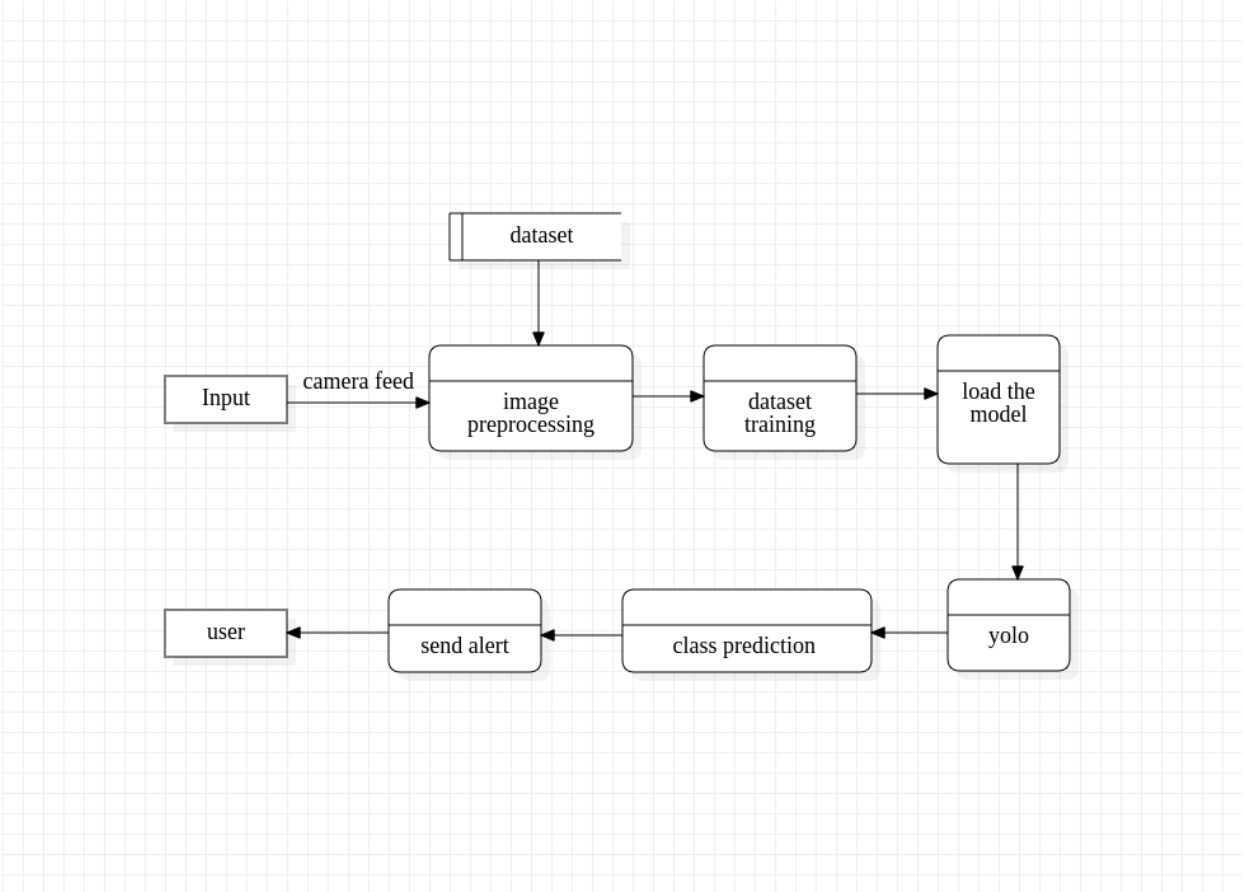

Fig 3.7 Collaboration Diagram

CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1 IMPLEMENTATION:

path.yaml file:

path: ../drive/MyDrive/ Yo8/New U

train: ../train/images

val: ../valid/images

nc: 12
names:
['Boar','Cheetah','Deer','Elephant','Hyena','Jaguar','Leopard','Lion','Panda','Snake'
,'Tiger','Wolf']

Animal_Detection.ipynb file:

```
!pip install -q kaggle

import os

import shutil

import zipfile

import numpy as np

import pandas as pd

import tensorflow as tf
```

```
from tensorflow import keras

from keras.models import load_model

from tqdm import tqdm



# Create a directory named .kaggle in the home directory
os.makedirs(os.path.expanduser('~/.kaggle'), exist_ok=True)

# Copy the kaggle.json file to the .kaggle directory in the home directory
shutil.copyfile('kaggle.json', os.path.expanduser('~/.kaggle/kaggle.json'))
kaggle datasets download -d lokeshvloki/animal-dataset-image-with-annotation

# Specify the path of the zip file
zip_path = ' animal-dataset-image-with-annotation.zip'

# Specify the directory where you want to extract the files
extract_path = ' animal-dataset-image-with-annotation '

# Open the zip file
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    # Extract all the files to the specified directory
    zip_ref.extractall(extract_path)



 !nvidia-smi

!pip install ultralytics

from ultralytics import YOLO

        !yolo mode=predict model=yolov8m.pt conf=0.25
source="../content/drive/MyDrive/yolov8animal/Animal/photo_2023-03-27_16-48-26.jpg"
```

!yolo task=detect mode=train model=yolov8l.pt data="../content/drive/MyDrive/ Yo8/New U/path.yaml" epochs=140 imgsz=736

Animal_detection-yolo.py file:

```
from ultralytics import YOLO
import cv2
import cvzone
import math
import time
import requests
import base64


import time

start_time=time.time()

cap = cv2.VideoCapture("ven/video/The_friendships_of_elephant_and_dogs_-_ElephantNews(360p).mp4")
cap.set(3, 1280)
cap.set(4, 720)
model = YOLO("ven/finalYolo/best_L-model2.pt")
classnames = ['Boar','Cheetah','Deer','Elephant','Hyena','Jaguar','Leopard','Lion','Panda','Snake','Tiger','Wolf']
import requests

flag =True
def send_telegram_msg(message):
    TOKEN = "6263063793:AAGo22xawSSiz9ciYnhFGPRa0c1MGPPrKfE"
    chat_id = "5341914866"
    # message = "Hello,World!"
```

```
url=f"https://api.telegram.org/bot{TOKEN}/sendMessage?chat_id={chat_id}&text={message}"

    response = requests.get(url)

    return response.json()

while True:
    success, img = cap.read()
    results = model(img, stream=True)
    for r in results:
        boxes = r.boxes
        for box in boxes:
            x1, y1, x2, y2 = box.xyxy[0]
            x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
            w, h = x2 - x1, y2 - y1
            cvzone.cornerRect(img, (x1, y1, w, h))
            conf = math.ceil((box.conf[0] * 100)) / 100
            cls = int(box.cls[0])

            if conf >0.87:
                    cvzone.putTextRect(img, f'{classnames[cls]} {conf}', (max(0, x1), max(35, y1)),scale=1,thickness=1)


              if flag:
                send_telegram_msg(f'{classnames[cls]}' + " Please, Go to safe place")
                flag = False

              cu_time = time.time()
              e_time = cu_time - start_time
              if e_time >= 30:
                        send_telegram_msg(f'{classnames[cls]}' + " Please, Go to safe place")
                    start_time = cu_time
```

```
cv2.imshow('Animal', img)
cv2.waitKey(1)
```

4.2 RESULTS

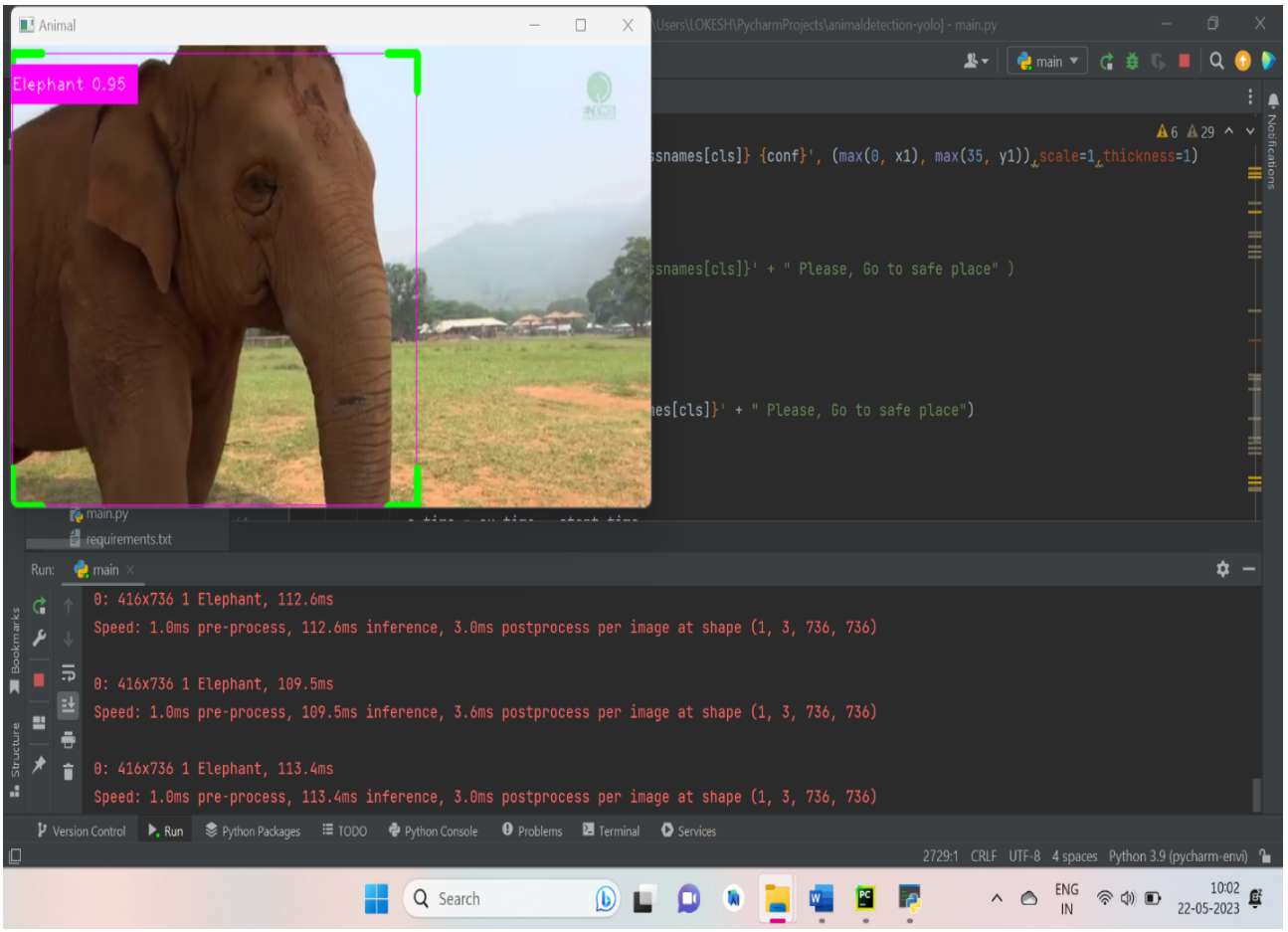

Fig 4.1 Detected Output 1

Fig 4.1 depicts the output for the animal detection using Deep Learning.In this output,Elephant is detected with prediction of 0.95.This system was able to correctly deduce that the image contains elephant using yolo v8 model.

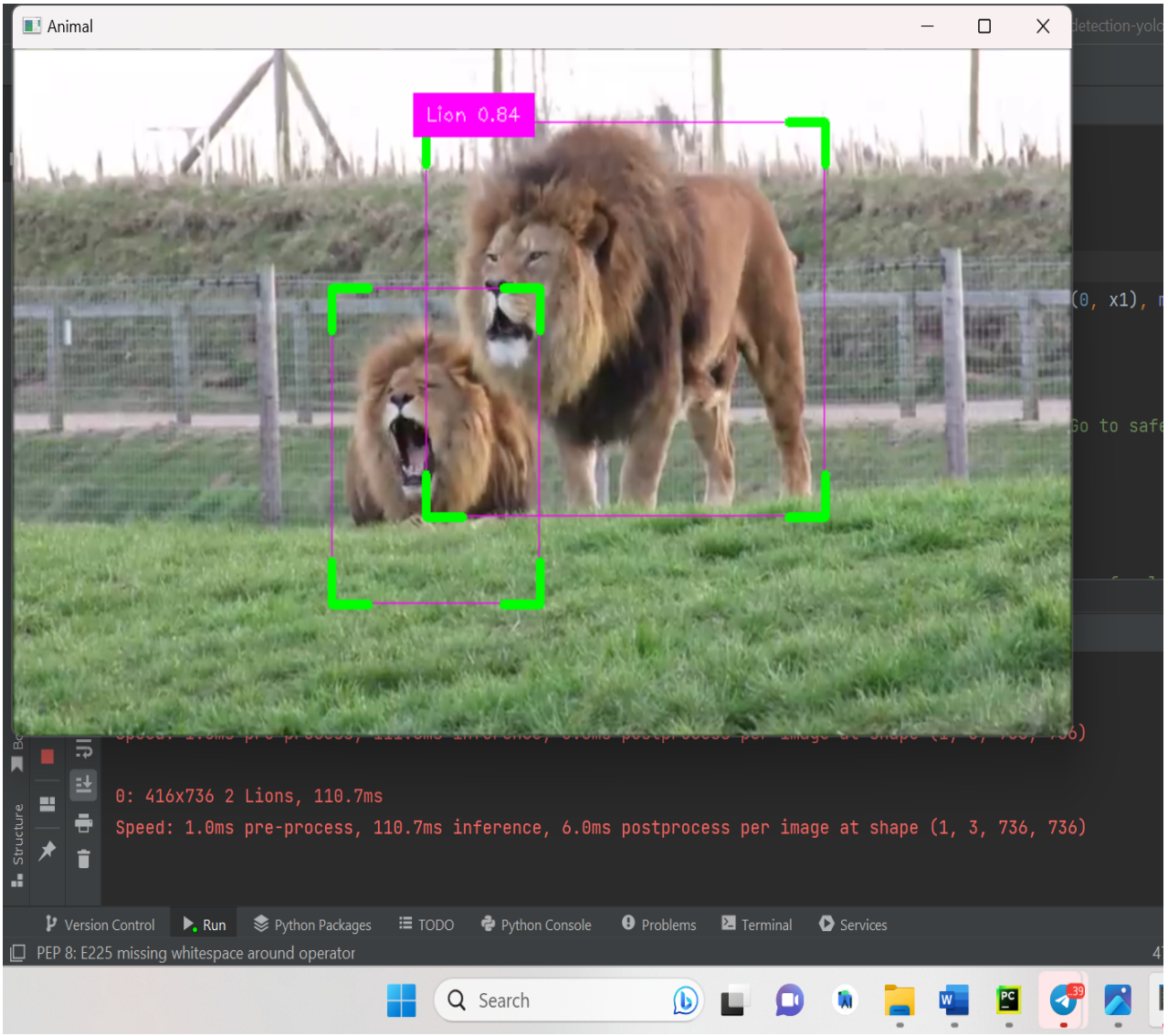

Fig 4.2 Detected Output 2

Fig 4.2 depicts the output for the animal detection using Deep Learning.In this output,Lion is detected with prediction of 0.84.This system was able to correctly deduce that the image contains two lions using yolo v8 model.

Fig 4.3 Alert Message Output

Fig 4.3 depicts alert message through telegram after the detecting animals within 2-3 seconds .

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

Animal detection using the YOLO (You Only Look Once) algorithm has proven to be highly effective. YOLO's unique approach of performing object localization and classification simultaneously in real-time sets it apart from traditional methods. By dividing the image into grid-based regions, YOLO accurately predicts bounding boxes and class probabilities for each region, enabling it to detect a wide range of animal species in various environmental conditions. Its ability to handle multiple animals in a single image further enhances its applicability.

The real-time capabilities of YOLO make it valuable for applications such as wildlife monitoring, animal tracking, and surveillance systems. It has the potential to aid researchers in wildlife conservation efforts by enabling them to track animal populations, study behavior, and identify endangered species. Additionally, in agricultural settings, YOLO can enhance safety measures by preventing wildlife-related accidents and minimizing crop damage. With continuous development and improvements, YOLO holds great promise for advancing animal detection technologies and contributing to various fields related to animal welfare, research, and environmental conservation.

5.2 FUTURE WORK:

We are aspiring to develop android app with some key features such as tracking,classifying,locating animals in rural areas.This will helps to find the habitual place and behaviour of animals using the deep learning technology.

REFERENCES

1. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 779-788).

2. Zhu, Y., Zhou, X., Ye, Q., Qiu, Q., & Jiao, J. (2019). Feature selective anchor-free module for single-shot object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 4783-4792).

3. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In the European Conference on Computer Vision (ECCV) (pp. 21-37).

4. Zhang, S., Wen, L., Bian, X., & Lei, Z. (2018). Single-Shot Refinement Neural Network for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 4203-4212).

5. Choi, W., Chao, Y. W., Pantofaru, C., & Savarese, S. (2018). Context-based ensemble learning for object detection. International Journal of Computer Vision, 126(12), 1296-1312.

6. Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Advances in Neural Information Processing Systems (NIPS) (pp. 91-99).

7. Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal Loss for Dense Object Detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV) (pp. 2980-2988).

8. Zhou, Y., Zhu, X., & Bai, X. (2019). Objects as Points. arXiv preprint arXiv:1904.07850.

9. Law, H., & Deng, J. (2018). Cornernet: Detecting objects as paired keypoints. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 734-750).

10. Dollár, P., Appel, R., Belongie, S., & Perona, P. (2014). Fast feature pyramids for object detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(8), 1532-1545.