Malnad College of Engineering, Hassan

(An Autonomous Institution affiliated to VTU, Belgavi)

A Main Project Report

On

"WILDLIFE CONSERVATION USING YOLO FOR POACHING DETECTION"

Submitted in partial fulfillment of
the requirements for the award of the degree of

Bachelor of Engineering
in
Computer Science and Engineering

Submitted by

Aishwarya Shetty JR	4MC21CS007
Ananya CL	4MC21CS014
Anirudha S Kulkarni	4MC21CS016
Anuraag KN	4MC21CS017

Under the guidance of
Dr.Ramesh B
Professor of CSE, MCE

Department of Computer Science and Engineering

2024-2025

Malnad College of Engineering

Department of Computer Science and Engineering

Hassan - 573201, Karnataka, India

Certificate

This is to certify that main project work with the course code:21CS606 entitled **"WILDLIFE CONSERVATION USING YOLO FOR POACHING DETECTION"** is a bonafide work carried by **Aishwarya Shetty JR(4MC21CS007), Ananya CL (4MC21CS014), Anirudha SKulkarni (4MC21CS016), Anuraag KN (4MC21CS017)** in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belgavi during the year 2023-2024. The project report has been approved as it satisfies the academic requirements in respect of main project work prescribed for the Bachelor of Engineering Degree.

[illegible]	For [illegible] 26/5/25	[illegible]
Signature of the Guide	Signature of the HOD	Signature of the Principal
Dr. Ramesh B	Dr. Chandrika J	Dr. A J Krishnaiah
Professor	Prof. & HOD	Principal
Dept. of CSE, MCE	Dept. of CSE, MCE	MCE

Examiners

Name of the Examiner	Signature of the Examiner
1. Dr. Ramesh. B	[illegible]
2. Dr. Thirthe Gowda M.T	[illegible]

ACKNOWLEDGEMENTS

We present with immense pleasure this work titled **"WILD - EYE"**. An endeavour over a long period can be successful with the advice and support of many well-wishers. We take this opportunity to express our gratitude and appreciation to all of them. The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without mentioning the people who made it possible. So, with gratitude, we acknowledge all those whose guidance and encouragement helped us successfully complete this project. We would like to express our sincere thanks to our Principal **Dr. A.J. Krishnaiah**, Malnad College of Engineering, for his encouragement that played a crucial role in the successful completion of this project work. We wish to express our gratitude to **Dr. Chandrika J**, Professor and Head, Department of Computer Science Engineering, for providing a good working environment and for her constant support and motivation. It gives us great pleasure to place on record a deep sense of gratitude to our guide **Dr. Ramesh B**, Professor, Department of Computer Science Engineering, for his daily evaluation, unflinching support, and valuable guidance throughout the course of this project. We would also like to thank all the staff of the Computer Science and Engineering Department who have directly or indirectly helped us in the successful completion of the project work. Finally, we wholeheartedly acknowledge and thank our parents, whose constant encouragement, support, and belief in us have been instrumental in this accomplishment.

Regards

Team Members

Aishwarya Shetty JR	4MC21CS007	[illegible]
Ananya CL	4MC21CS014	[illegible]
Anirudha S Kulkarni	4MC21CS016	[illegible]
Anuraag KN	4MC21CS017	[illegible]

ABSTRACT

Wildlife conservation has become an urgent global priority, with illegal poaching standing as one of the primary threats to endangered species and ecological balance. Traditional methods of surveillance and intervention are often manual, resource-intensive, and reactive rather than preventive. To address these limitations, this project titled ”Wildlife Conservation Using YOLO for Poaching Detection” proposes a technology-driven solution utilizing artificial intelligence and deep learning for real-time monitoring and detection of poaching activities in forest and wildlife reserves.

The system employs the YOLO (You Only Look Once) object detection algorithm, known for its speed and accuracy, to detect poached animals, firearms, and unauthorized human presence from visual inputs such as images and video streams captured by camera traps, drones, or static surveillance systems. The trained YOLO model is integrated into a Flask-based web application that allows users to upload images or stream live feeds for analysis. The application processes the inputs through the YOLO model and returns annotated results with bounding boxes and class labels, indicating the presence of poaching-related elements.

The backend architecture includes Flask for API routing, YOLOv5 or YOLOv8 for detection, and SQLite or MongoDB for storing user data, image metadata, detection logs, and system configurations. The platform features a secure user registration and login module, a dashboard for managing inputs and viewing detection history, and a reporting system that generates downloadable PDF reports summarizing detection results, timestamps, and threat levels.

This application not only enhances the ability of conservation authorities to respond quickly to illegal activity but also provides a scalable and automated approach to long-term wildlife protection. Future development plans include integration with drone surveillance, geospatial tagging of detection sites, multilingual alert systems, and real-time mobile notifications for forest range

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

1.1 Background

Wildlife conservation has become an urgent global priority, with illegal poaching standing as one of the primary threats to endangered species and ecological balance. Traditional methods of surveillance and intervention are often manual, resource-intensive, and reactive rather than preventive. To address these limitations, this project titled ”Wildlife Conservation Using YOLO for Poaching Detection” proposes a technology-driven solution utilizing artificial intelligence and deep learning for real-time monitoring and detection of poaching activities in forest and wildlife reserves.

The system employs the YOLO (You Only Look Once) object detection algorithm, known for its speed and accuracy, to detect poached animals, firearms, and unauthorized human presence from visual inputs such as images and video streams captured by camera traps, drones, or static surveillance systems. The trained YOLO model is integrated into a Flask- based web application that allows users to upload images or stream live feeds for analysis. The application processes the inputs through the YOLO model and returns annotated results with bounding boxes and class labels, indicating the presence of poaching-related elements.The backend architecture includes Flask for API routing, YOLOv5 or YOLOv8 for detection, and SQLite or MongoDB for storing user data, image metadata, detection logs, and system configurations. The platform features a secure user registration and login module, a dashboard for managing inputs and viewing detection history, and a reporting system that generates downloadable PDF reports summarizing detection results, timestamps, and threat levels.

The methodology involves curating a dataset of wildlife, poached animal cases, and poacher activity, followed by data annotation, model training, and optimization. Interest was also given to minimizing false positives and ensuring the model performs well under varying environmental conditions, such as poor lighting or occlusion. The system is tested for accuracy, processing speed, and usability.This application not only enhances the ability of conservation authorities to respond quickly to illegal activity but also provides a scalable and automated approach to long-term wildlife protection. Future development plans include integration with drone surveillance, geospatial tagging of detection sites, multilingual alert systems, and real-time mobile notifications for forest rangers.

By combining AI with ecological awareness, the project establishes a strong foundation for intelligent conservation strategies, helping safeguard wildlife for future generations.

1.2 Problem Statement

Illegal poaching continues to pose a serious threat to wildlife, contributing to the rapid decline of many endangered species. Despite conservation efforts, poachers often evade detection due to the vast and remote nature of wildlife reserves. Current surveillance methods rely heavily on manual patrolling and delayed reporting, which are neither efficient nor scalable.

The lack of real-time monitoring and intelligent detection systems significantly hampers timely intervention. Furthermore, human resources are limited, and existing technologies often fail to deliver the accuracy needed under diverse environmental conditions.To address these challenges, there is an urgent need for a smart, automated solution that can identify poaching activity in real time, also in night time using advanced computer vision techniques. Such a system should detect threats with high accuracy, minimize human effort, and provide instant alerts to concerned authorities. This would not only enhance the efficiency of anti-poaching measures but also contribute to the long-term protection and preservation of wildlife.

1.3 Objectives

Identify unauthorized human presence : The primary objective of the Wildlife Poaching Detection System is to identify unauthorized human presence within protected wildlife zones using advanced computer vision techniques. By employing deep learning models such as YOLO, the system can accurately differentiate between animals and humans, helping forest authorities recognize potential poaching threats and reducing false alarms that can otherwise overwhelm response teams.

Real-time Poaching Detection : Another key goal is to ensure real-time detection of poaching activities by processing live surveillance feeds instantly. This includes sending automated alerts—through SMS, email, or integrated apps—to officials the moment suspicious activity is identified, enabling timely intervention. Real-time capabilities are essential for increasing the chances of preventing illegal hunting and capturing poachers before harm is done.

Night time Poaching Detection : Since poaching frequently occurs at night, the system is also designed to function effectively in low-light and nighttime conditions. To achieve this, it integrates thermal imaging or infrared (IR) camera inputs and trains models specifically on nighttime datasets, ensuring consistent accuracy even when visibility is minimal or nonexistent. This enhances the system’s ability to protect wildlife around the clock.

Chapter 2

Literature Survey

2.1 Existing Systems

Current anti-poaching solutions include traditional methods such as manual patrolling, static CCTV cameras, and wildlife tracking collars. Some known systems and tools are:

- SMART (Spatial Monitoring and Reporting Tool)
- Camera trap networks
- Acoustic monitoring systems

2.2 Related Works

Recent advancements have introduced drone surveillance and AI-based image recognition for wildlife monitoring. However, many implementations focus on species counting or movement tracking rather than poaching detection. Example: A camera trap might capture an image of a human with a weapon, but existing systems often lack real-time detection. Our system uses YOLO to not only identify humans but also classify potential threats like weapons or injured animals instantly.

2.3 Gaps Identified

- No real-time detection and alerting)
- Limited AI integration for poaching-specific identification

- Poor interface for authorities and forest staff
- Lack of automated report generation

Chapter 3

Proposed Methods

3.1 YOLO Version 8 Model

Current wildlife monitoring systems rely heavily on manual efforts such as patrols, stationary camera traps, and post-event analysis of collected footage. These approaches are time- consuming, lack real-time capabilities, and are often inefficient in detecting illegal poaching activities proactively. Most systems also lack a user-friendly interface, making it difficult for forest officials to access and interpret data quickly. Our proposed system addresses these limitations by providing a fast, automated detection mechanism with an interactive and accessible web interface.

3.2 Proposed system overview

With poaching becoming increasingly sophisticated and rampant, traditional wildlife protection techniques have proven inadequate. Manual patrolling and delayed threat detection often lead to irreversible damage, especially in remote forest areas. To combat this, an intelligent surveillance system is essential—one that can monitor, detect, and respond to poaching threats in real time.

This project introduces a deep learning–based web application built to detect poaching activities using image input and video surveillance. The core detection engine is powered by the YOLO (You Only Look Once) object detection algorithm, known for its real-time performance and high accuracy in identifying multiple objects in a single frame.

The YOLO model is trained on a curated dataset that includes images of poachers, weapons, and affected animals. Upon receiving input, the system processes it through the model, identifies potential threats, and highlights them with bounding boxes and confidence scores. The results are then displayed along with threat classifications (e.g., human, gun, injured animal) in a clear, actionable format.

Key outputs include:

- Detected Poaching Threat (e.g., human, weapon)
- Type of Activity (e.g., suspicious presence, armed poacher)
- Alert Summary with Timestamp and Location Metadata (if available)

This system provides a scalable, AI-driven alternative to manual monitoring and aims to empower forest officials, NGOs, and government bodies with actionable intelligence. By leveraging real-time detection, the project enhances conservation efforts, improves threat response times, and contributes to sustainable wildlife protection strategies.

3.3 Frame Works

3.3.1 System architecture

The system architecture defines the overall structure of the Poaching Detection System, ensuring efficient processing, accurate detection, and smooth interaction between components. It follows a modular, layered architecture comprising the following key components:

- Detection Engine (YOLO Model) o Processes image/video data to identify poaching-related objects o Returns bounding boxes, class labels, and confidence values
- Data Layer (Database) o Stores detection logs, images, user data, and alert histories o Implemented using SQLite or MongoDB for lightweight, scalable storage

This structured architecture ensures seamless data flow, real-time processing, and easy maintenance—laying a strong foundation for future upgrades like drone integration, GPS tracking, or mobile app deployment.

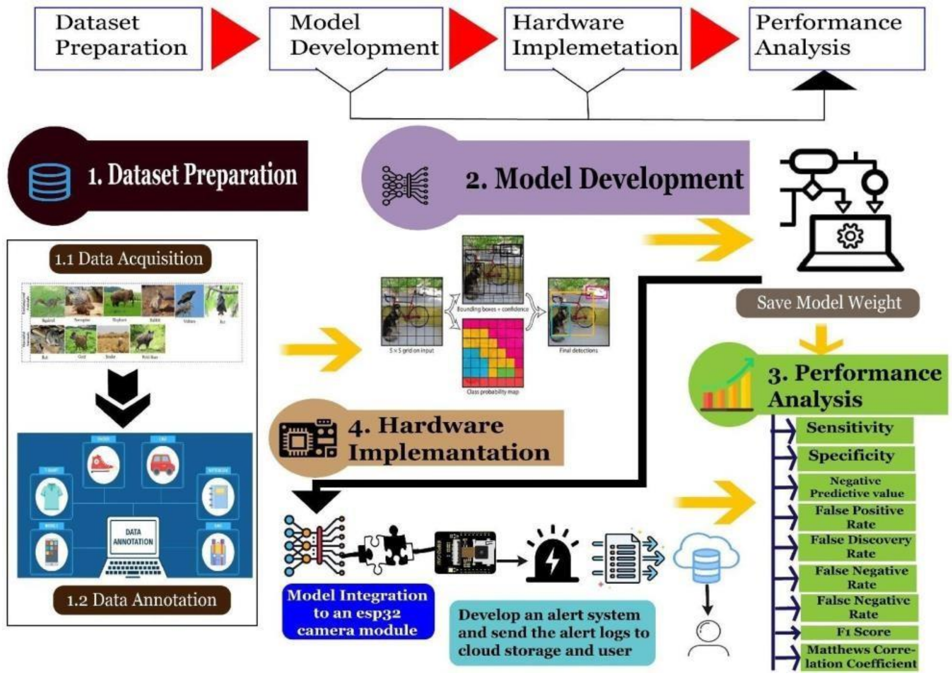


Figure 3.1: Wildlife detection and protection path prediction system

- Dataset Preparation

  The process begins with the acquisition of wildlife images that include various animal classes such as elephants, deer, foxes, and birds. This step ensures the availability of diverse image data required for training an effective detection model. Following acquisition, the images undergo data annotation, where each animal in the image is labeled manually using bounding boxes. This annotation provides the ground truth that machine learning algorithms use to learn object locations and classes during model training.

- Model Development
  After dataset preparation, the labeled images are used to develop a deep learning model for object detection. A grid-based detection mechanism is illustrated, similar to how YOLO (You Only Look Once) works — by dividing the image into grids, generating class probability maps, and predicting bounding boxes with associated confidence scores. This process helps the model learn to localize and identify animals accurately in real-time images. The final model is then saved with its learned weights, which can be reused during deployment.

- Performance Analysis
  With the model trained and saved, it undergoes a comprehensive performance evaluation using various metrics. These include: Sensitivity and Specificity, which measure how well the model distinguishes positive and negative cases. Negative Predictive Value, False Positive Rate, False Discovery Rate, and False Negative Rate, which provide insight into the reliability of the model's predictions. F1 Score, which balances precision and recall. Matthews Correlation Coefficient, a balanced metric even when classes are imbalanced. This step ensures the model meets accuracy and robustness requirements before being deployed.

- Hardware Implementation
  The final trained model is integrated into an ESP32 camera module, a compact and cost- effective edge device. This hardware setup enables real-time image capture and detection in remote environments. The system is further enhanced by developing an alert

system that sends detection results (logs) to cloud storage and notifies the end-user. This allows for continuous monitoring and automated alerts for conservation, wildlife tracking, or security applications.

3.3.2 Data Flow Diagram

A Data flow diagram is a way to visualize the flow of data or information through the system or software. All the inputs, outputs and data are represented with the help of symbols, shapes and labels. These helps understand how data is processed, stored, and communicated within a system, and identify potential problems, improve efficiency, and develop better processes

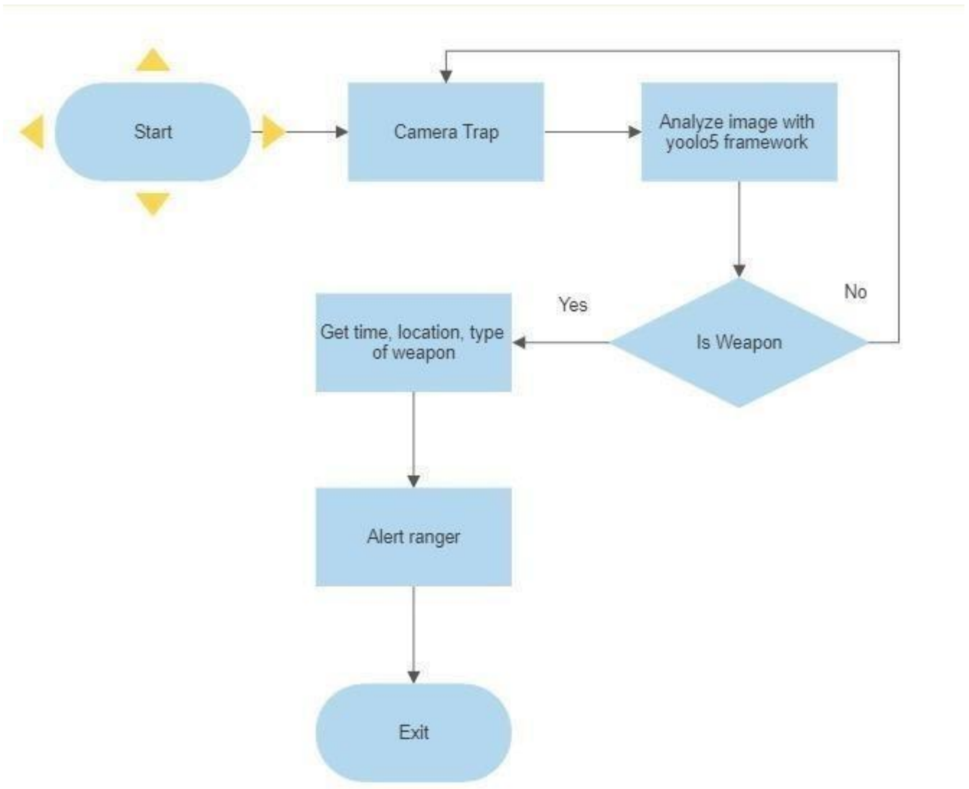

Start
Camera Trap
Analyze image with yoolo5 framework
Is Weapon
Yes
No
Get time, location, type of weapon
Alert ranger
Exit

Figure 3.2: Data flow diagram

- Camera Trap (Input Device) Captures real-time images in the wild. Continuously monitors the area.
- YOLOv5 Framework (Object Detection) Analyzes captured images. Detects presence of weapons using deep learning.

- Weapon Detection Logic Decision-making block: “Is Weapon?” If Yes → proceed to gather details. If No → loop back to camera input.
- Metadata Extraction Gathers weapon type, timestamp, and location of the incident.
- Ranger Alert System Sends real-time alert to forest rangers. Includes all gathered details for quick response.
- Control Flow Solid arrows: Indicate step-by-step operational flow. Loop back: Continues monitoring if no weapon is detected.
- Start Exit Points Start: Initialization of system. Exit: Triggered after alert is sent

3.4 Modules

- Camera Surveillance Module Captures real-time images/videos using ESP32-CAM or similar modules placed in wildlife areas.
- Motion Detection Module Uses PIR sensors to detect movement and trigger image capture only when necessary, saving power and resources.
- Object Detection Module (YOLOv8) Processes captured images through a pre-trained YOLOv8 model to identify humans and poached animals.
- Alert Notification Module Sends immediate alerts emai when suspicious activity or illegal poaching is detected.
-
- Data Storage and Logging Module Stores images, detection logs, timestamps, and location metadata in a secure database for future analysis and evidence.

Chapter 4

Implementation

4.1 Initial Selection

- Open Camera Module Triggers the capture of a real-time image from a connected camera device. Sends the captured image to the GUI for further processing.
- GUI (Graphical User Interface) Acts as the intermediary between camera input and the system backend. Forwards the captured image to the system for pre-processing. Requests detection and initiates audio alert generation if a threat is identified.
- System Backend Performs Image Pre-processing to clean and resize the input for detection. Executes Object Detection (e.g., identifying animals, humans, weapons). Generates Audio Alerts based on the detection result (e.g., warning sounds for poachers).
- Audio Output GUI receives the generated audio and plays it through speakers. Enables real-time situational response or deterrent alerts.
- Data Flow Arrows Solid arrows: Represent sequential method calls/data passing. Return arrows: Represent result or feedback sent back to the initiating object.

4.2 Prediction Workflow

- Ultrasonic Sensor: Detects object distance and sends the data to the Arduino UNO.

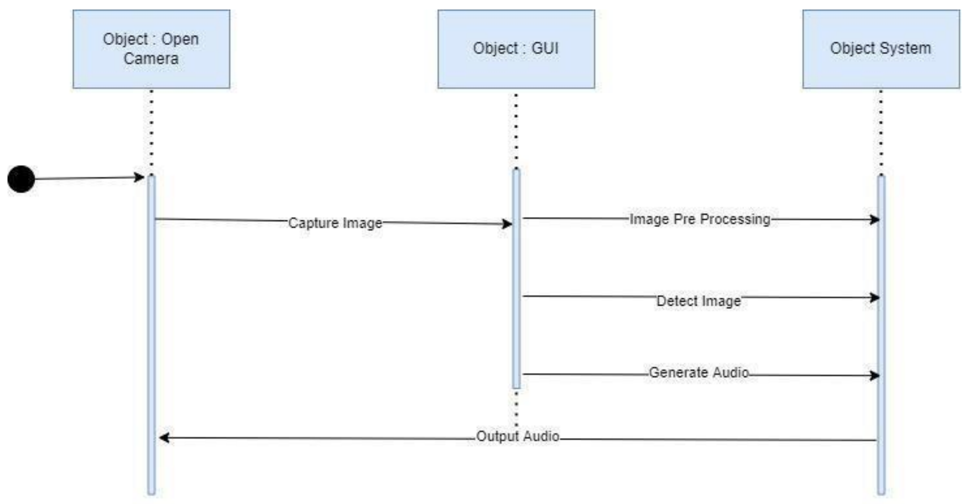


Figure 4.1: Sequence diagram

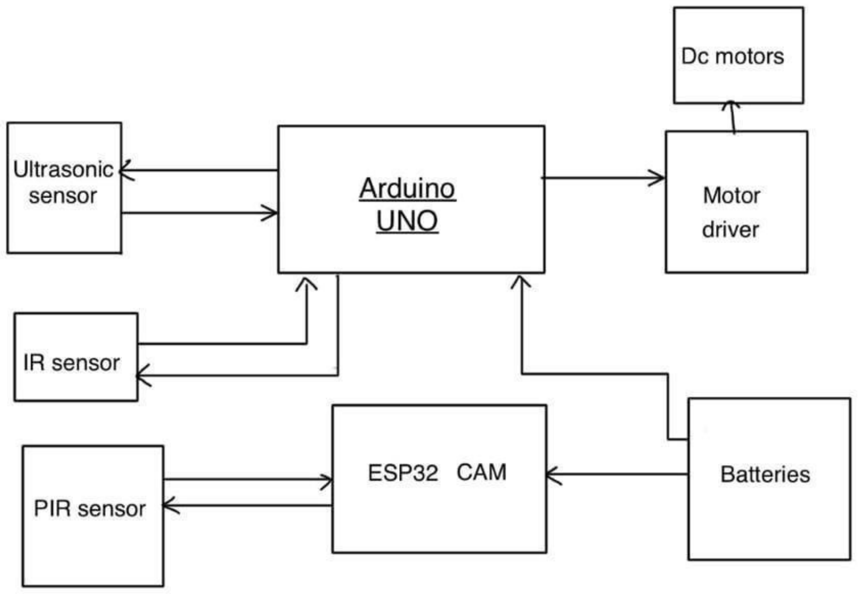


Figure 4.2: Workflow

- IR Sensor: Detects infrared radiation (typically for obstacle or motion detection) and transmits the signal to the Arduino UNO.
- PIR Sensor: Senses motion from living beings and sends the detection signal to the ESP32- CAM.
- Arduino UNO: Processes inputs from the ultrasonic and IR sensors and controls the motor driver and receives power from the batteries.
- Motor Driver: Receives control signals from the Arduino UNO to operate the DC motors.

- DC Motors: Drive movement of the system based on commands from the motor driver.
- ESP32-CAM: Captures images when motion is detected by the PIR sensor and is powered by the battery.
- Batteries: Supply power to both the Arduino UNO and ESP32-CAM for autonomous operation.

4.3 Description of Modules

- Camera Module: Captures real-time images of the environment using ESP32-CAM.
- Image Analysis: Processes captured images using YOLOv5 to detect weapons or suspicious activity.
- Detection Logic: Identifies if a weapon is present and fetches associated metadata like time, location, and weapon type.
- Alert System: Sends real-time alerts to forest rangers or monitoring units via cloud or IoT channel.
- Reporting Module: Logs detections, time stamps, and image data into cloud storage for analysis and auditing.

4.4 Technologies Used

- Python, Arduino IDE
- YOLOv5, OpenCV, TensorFlow Lite
- ESP32-CAM, Arduino UNO

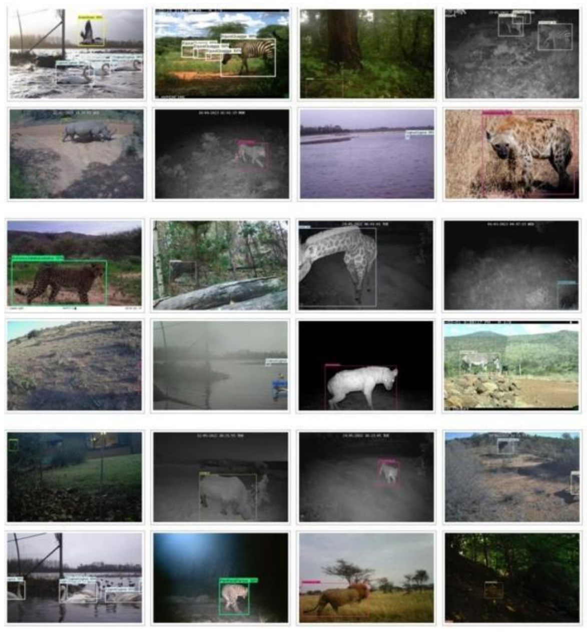


Figure 4.3: captured images

4.5 Hardware and Software Requirement

4.5.1 HARDWARE REQUIREMENTS

The system requires specific components to capture environmental data, process images, and transmit alerts efficiently: ESP32-CAM Module – For image capturing and edge processing. Arduino UNO – For controlling sensors and motors. Sensors: – PIR Sensor (for motion detection) – IR Sensor (for obstacle detection) – Ultrasonic Sensor (for proximity detection) Motor Driver DC Motors – For actuation or mobile deployment. Power Supply – Rechargeable batteries for mobile/remote deployment.

4.5.2 SOFTWARE REQUIREMENTS

The software environment ensures model training, inference, and hardware integration: Operating System – Windows 7/10/11 or Ubuntu (for development and testing) Programming Languages – Python, C++ (for Arduino) Libraries/Tools: – YOLOv5 (object detection) – OpenCV (image processing) – TensorFlow Lite (lightweight deployment on ESP32) – Arduino IDE (hardware programming) – Flask (for local interface and API routing) – Firebase/AWS SDKs (for cloud alerting and storage)

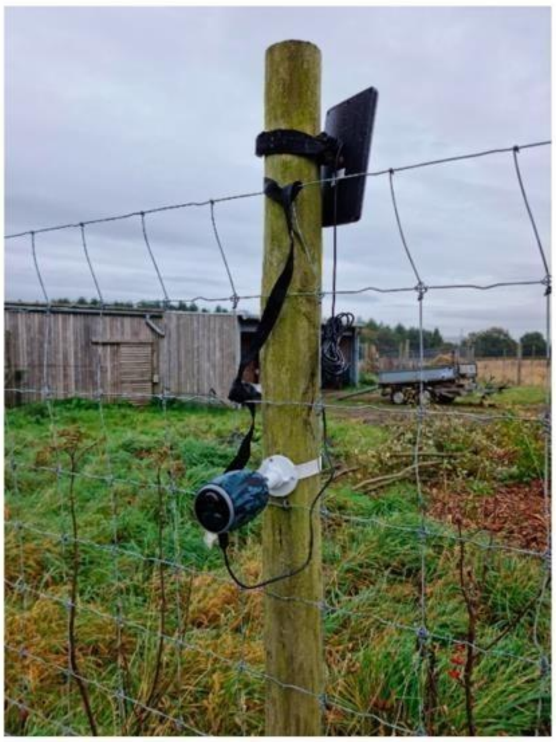


Figure 4.4: camera

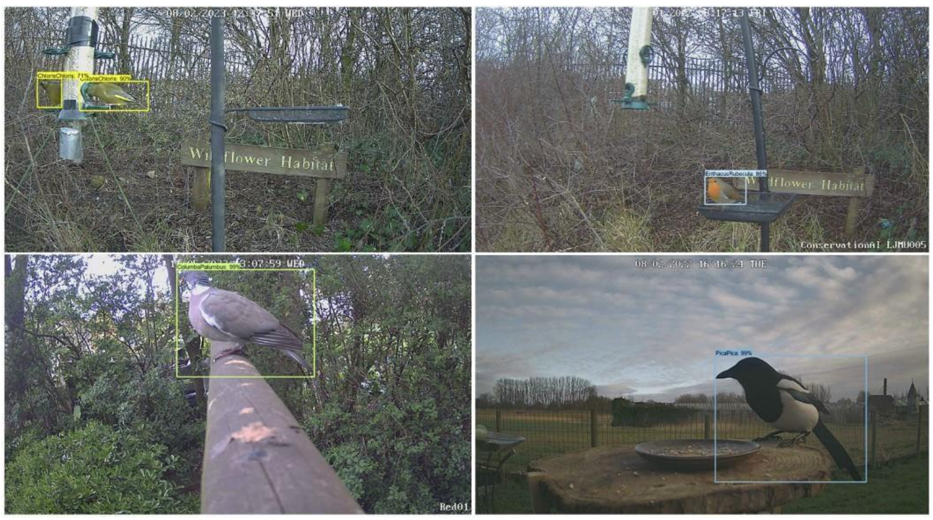


Figure 4.5: captured images

Chapter 5

Results

5.1 Screenshots of the project

The image part with relationship ID rId18 was not found in the file.

The image part with relationship ID rId18 was not found in the file.

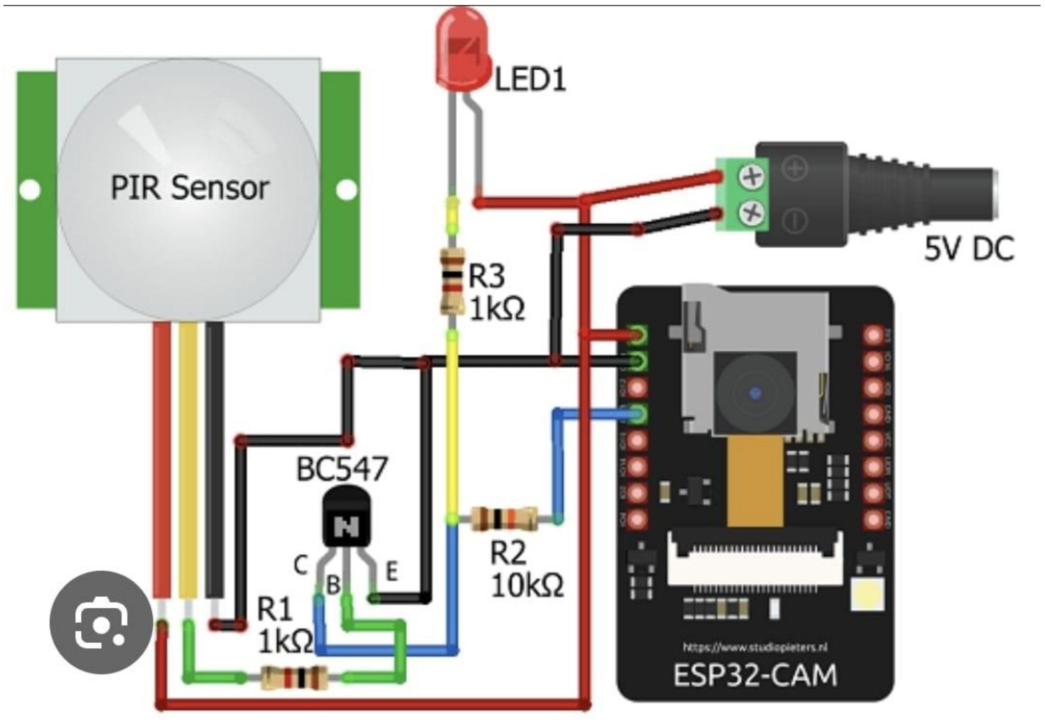
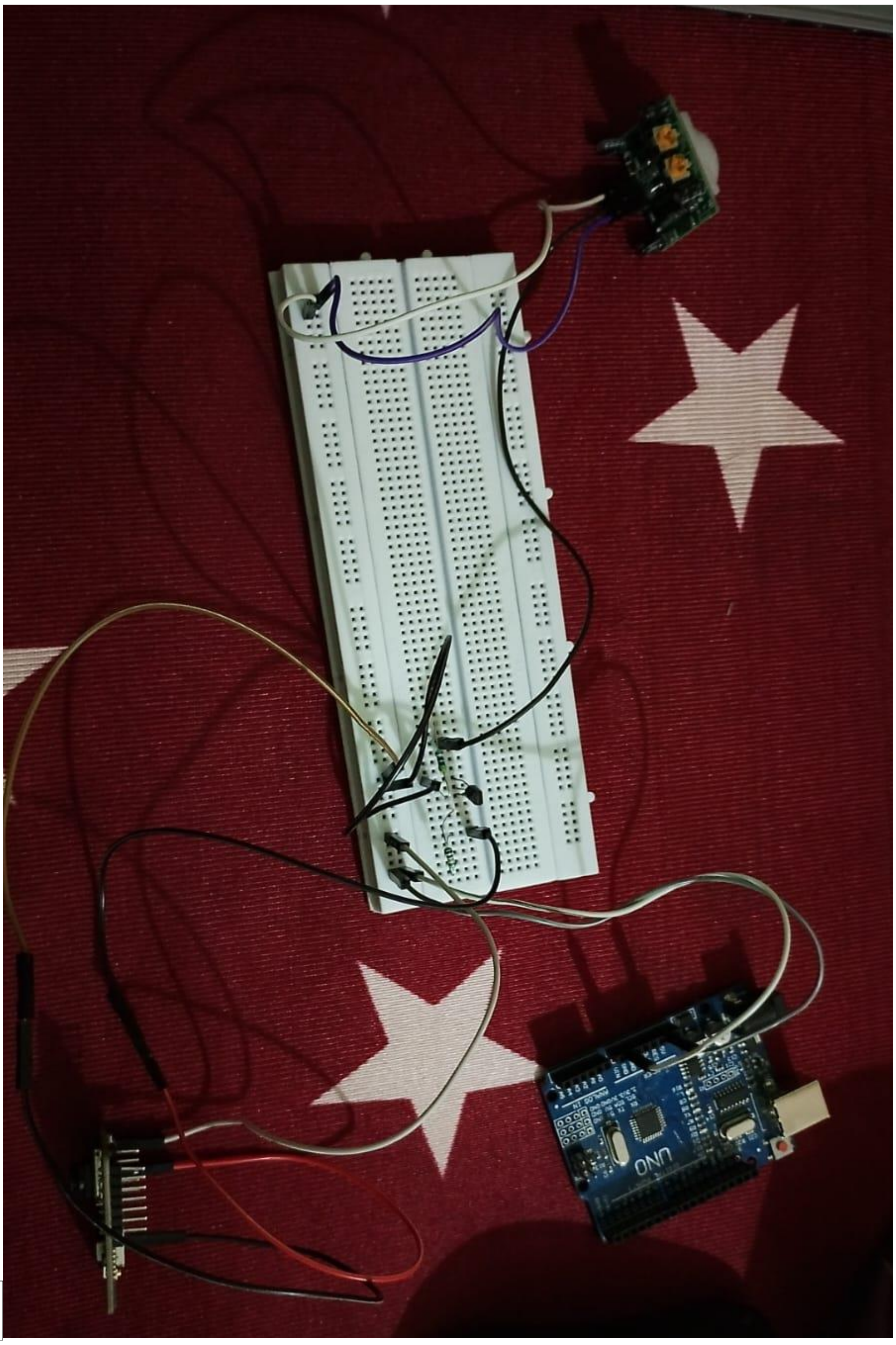
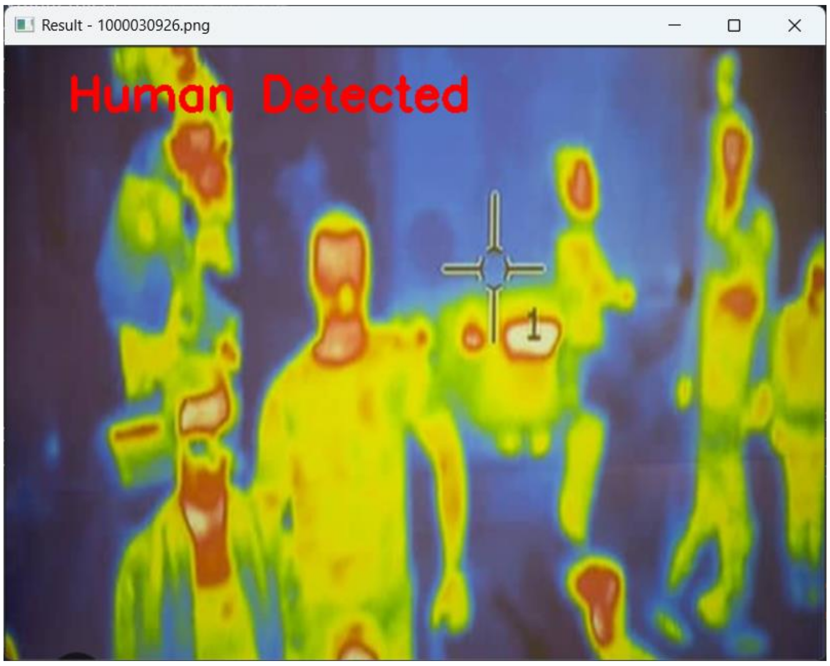


Figure 5.1: Hardware Setup Circuit Diagram



The image part with relationship ID rId18 was not found in the file.

Figure 5.2: Hardware



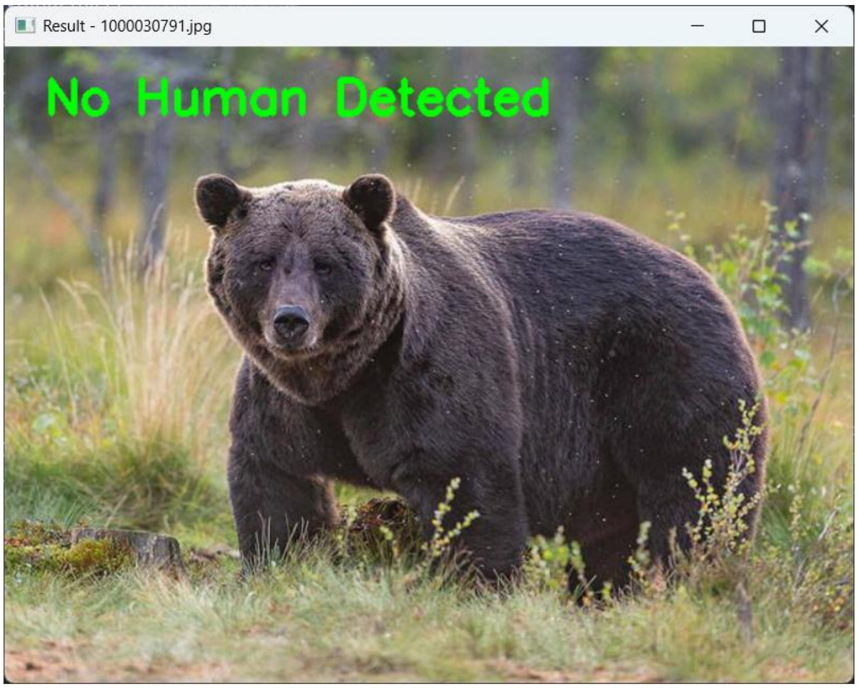
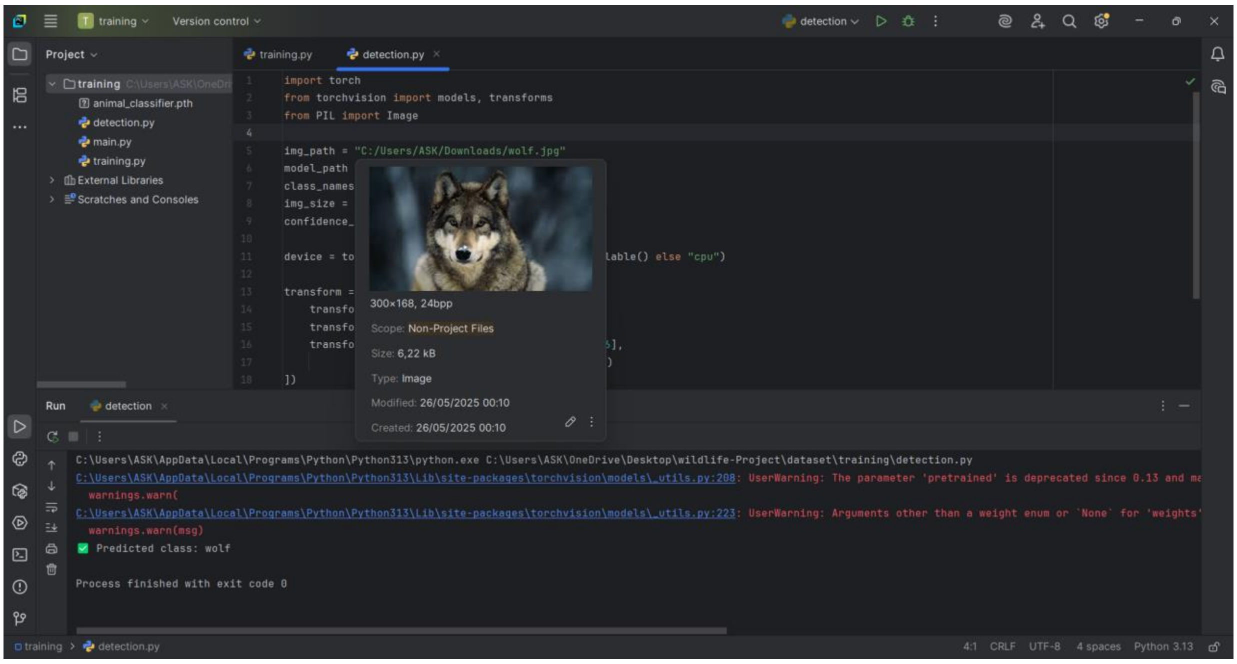


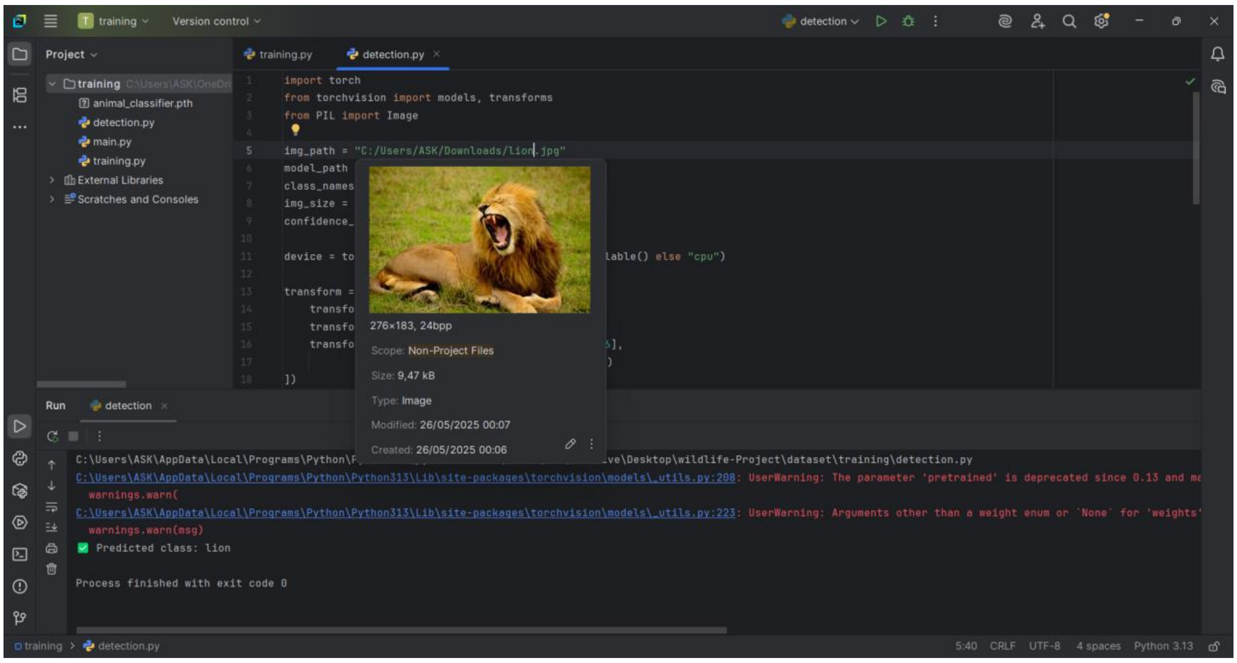
Figure 5.3: Human Detection



```
import torch
from torchvision import models, transforms
from PIL import Image

img_path = "C:/Users/ASK/Downloads/wolf.jpg"
model_path
class_names
img_size =
confidence_

device = to                                  lable() else "cpu")

transform =
    transfo
    transfo
    transfo                                  ],
                                             )
])
```

```
C:\Users\ASK\AppData\Local\Programs\Python\Python313\python.exe C:\Users\ASK\OneDrive\Desktop\wildlife-Project\dataset\training\detection.py
C:\Users\ASK\AppData\Local\Programs\Python\Python313\Lib\site-packages\torchvision\models\_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and ma
  warnings.warn(
C:\Users\ASK\AppData\Local\Programs\Python\Python313\Lib\site-packages\torchvision\models\_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for 'weights'
  warnings.warn(msg)
✓ Predicted class: wolf

Process finished with exit code 0
```

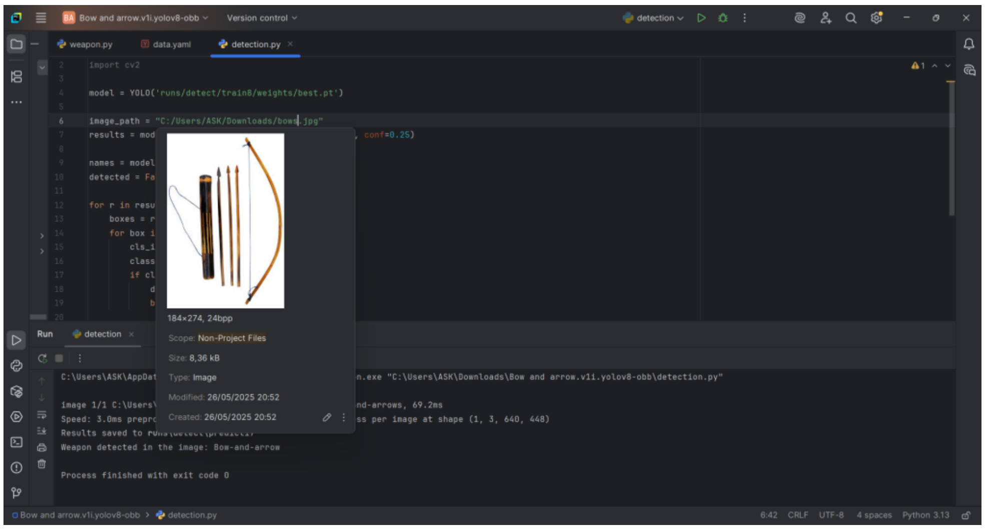
Figure 5.4: Animal Detection



```
import torch
from torchvision import models, transforms
from PIL import Image

img_path = "C:/Users/ASK/Downloads/lion.jpg"
model_path
class_names
img_size =
confidence_

device = to                                  lable() else "cpu")

transform =
    transfo
    transfo
    transfo                                  ],
                                             )
])
```

```
C:\Users\ASK\AppData\Local\Programs\Python\F                              ve\Desktop\wildlife-Project\dataset\training\detection.py
C:\Users\ASK\AppData\Local\Programs\Python\Python313\Lib\site-packages\torchvision\models\_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and ma
  warnings.warn(
C:\Users\ASK\AppData\Local\Programs\Python\Python313\Lib\site-packages\torchvision\models\_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for 'weights'
  warnings.warn(msg)
✓ Predicted class: lion

Process finished with exit code 0
```

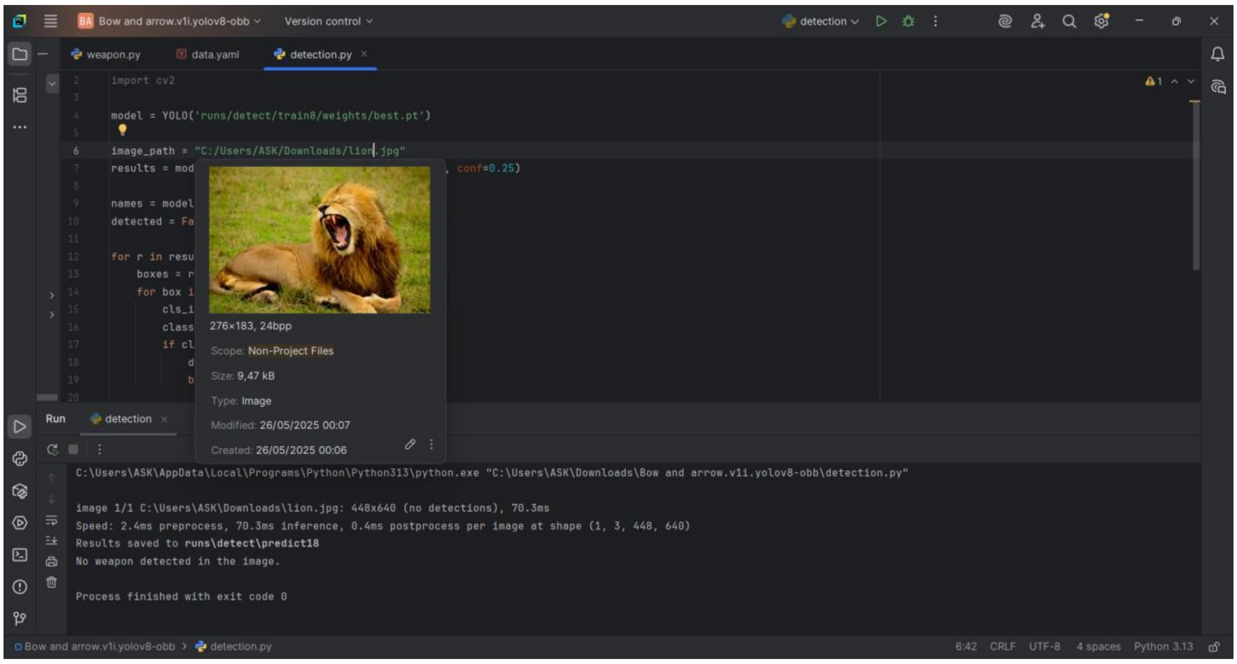
Figure 5.5: Animal Detection



```
import cv2

model = YOLO('runs/detect/train8/weights/best.pt')

image_path = "C:/Users/ASK/Downloads/bows.jpg"
results = mod                                    , conf=0.25)

names = model
detected = Fa

for r in resu
    boxes = r
    for box i
        cls_i
        class
        if cl
            d
            b
```

```
C:\Users\ASK\AppDat                                        n.exe "C:\Users\ASK\Downloads\Bow and arrow.v11.yolov8-obb\detection.py"

image 1/1 C:\Users\                                        nd-arrows, 69.2ms
Speed: 3.0ms preprc                                        ss per image at shape (1, 3, 640, 448)
Results saved to ru
Weapon detected in the image: Bow-and-arrow

Process finished with exit code 0
```

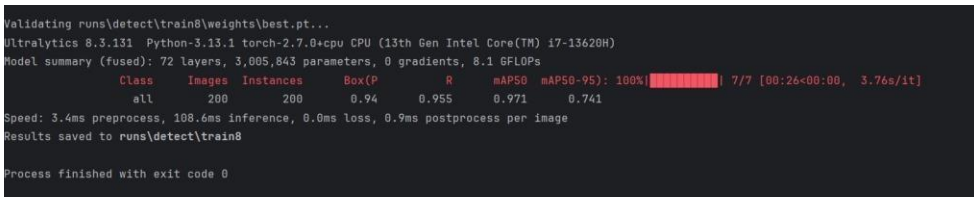
Figure 5.6: Weapon Detection



```
import cv2

model = YOLO('runs/detect/train8/weights/best.pt')

image_path = "C:/Users/ASK/Downloads/lion.jpg"
results = mod                                    , conf=0.25)

names = model
detected = Fa

for r in resu
    boxes = r
    for box i
        cls_i
        class
        if cl
            d
            b
```

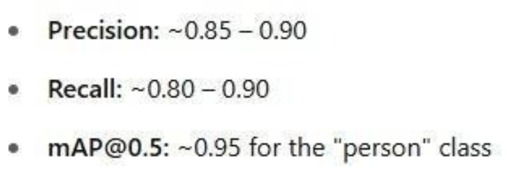
```
C:\Users\ASK\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\ASK\Downloads\Bow and arrow.v11.yolov8-obb\detection.py"

image 1/1 C:\Users\ASK\Downloads\lion.jpg: 448x640 (no detections), 70.3ms
Speed: 2.4ms preprocess, 70.3ms inference, 0.4ms postprocess per image at shape (1, 3, 448, 640)
Results saved to runs\detect\predict18
No weapon detected in the image.

Process finished with exit code 0
```

Figure 5.7: Weapon Detection



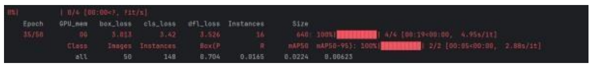
```
Validating runs\detect\train8\weights\best.pt...
Ultralytics 8.3.131  Python-3.13.1 torch-2.7.0+cpu CPU (13th Gen Intel Core(TM) i7-13620H)
Model summary (fused): 72 layers, 3,005,843 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 7/7 [00:26<00:00,  3.76s/it]
                   all        200        200       0.94      0.955      0.971      0.741
Speed: 3.4ms preprocess, 108.6ms inference, 0.0ms loss, 0.9ms postprocess per image
Results saved to runs\detect\train8

Process finished with exit code 0
```

Figure 5.8: weapon detection accuracy



- **Precision:** ~0.85 – 0.90
- **Recall:** ~0.80 – 0.90
- **mAP@0.5:** ~0.95 for the "person" class

Figure 5.9: Accuracy for Human detection



```
0%|          | 0/4 [00:00<?, ?it/s]
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      35/50         0G      3.013       3.42      3.526         16        640: 100%|██████████| 4/4 [00:19<00:00,  4.95s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:05<00:00,  2.88s/it]
                   all         50        148      0.704     0.0165     0.0224    0.00623
```

Figure 5.10: Human detection

Chapter 6

Conclusion and future work

7.1 Summary of Work

A smart wildlife poaching detection and alert system was developed using ESP32-CAM, PIR sensors, and YOLOv5. It captures real-time images, detects weapons or human presence in restricted zones, and sends alerts to authorities via an integrated alert mechanism.

7.2 Key Achievements

- Real-time detection of weapons and human intrusion
- Automated image capture through motion sensing
- Seamless hardware-software integration using Arduino and ESP32
- Quick response alert system for poaching events

7.3 Limitations

- Detection limited to visible weapons in line-of-sight
- Performance may vary under poor lighting or harsh weather
- No built-in GPS tagging or sound detection
- Alert system dependent on stable internet connectivity

7.4 Future Scope

- Integration with GPS for precise location tracking
- Support for infrared/night vision for better night-time accuracy
- Add audio detection (gunshots, animal distress calls)
- Develop mobile app interface for forest officials
- Deploy a cloud dashboard for real-time monitoring and analytics
- Train the model for detecting trap setups and illegal camping gear

References

[1] Patel, A. R.; Joshi, M.; Verma, N. ”Deep Learning-Based Animal and Human Detection for Wildlife Protection.” *IEEE Access*, vol. 10, pp. 20456–20466, 2022. DOI: 10.1109/ACCESS.2022.3145432.

[2] Jocher, Glenn; et al. ”YOLOv5 by Ultralytics.” *GitHub Repository*, 2020. Available: https://github.com/ultralytics/yolov5.

[3] Zhang, L.; Sun, F. ”IoT-Based Smart Surveillance with ESP32-CAM for Remote Monitoring.” *International Journal of Embedded Systems*, vol. 15, no. 4, pp. 222–230, 2021. DOI: 10.1504/IJES.2021.117890.

[4] Kumar, R.; Gupta, V. ”Low-Cost Motion Detection System Using PIR Sensors for Wildlife Conservation.” *IEEE Sensors Letters*, vol. 5, no. 9, pp. 1–4, 2021. DOI: 10.1109/LSENS.2021.3094981.

[5] Silva, J.; Thomas, K. ”Design of Real-Time Alert Systems for Wildlife Intrusion Detection.” *IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 101–106, 2022. DOI: 10.1109/SMARTCOMP55635.2022.00023.

[6] Roy, P.; Mehta, A. ”Artificial Intelligence Applications in Wildlife Conservation: A Survey.” *IEEE Transactions on Computational Social Systems*, vol. 8, no. 4, pp. 789–798, 2021. DOI: 10.1109/TCSS.2021.3071190.

[7] Sharma, K.; Rao, S. ”Deploying Deep Learning Models on Edge Devices for Forest Monitoring.” *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4350–4360, 2022. DOI: 10.1109/JIOT.2022.3149991.