A
Project Report
On

ANIMAL DETECTION AND DETERRENCE SYSTEM USING AI

Submitted in partial fulfilment of the award of the degree of
UNDER GRADUATION
In
COMPUTER APPLICATION

Submitted by
S R LEKHANA [U10UD22S0002]

UNDER THE GUIDANCE OF
S R MEGHANA
DEPARTMENT OF COMPUTER SCIENCE
KSAWUV, MANDYA

SUBMITTED TO
UNDER GRADUATE DEPARTMENT OF COMPUTER SCIENCE,
KARNATAKA STATE AKKAMAHADEVI WOMEN UNIVERSITY, VIJAYAPURA
Extension Centre for UG, PG Studies & Research, B.H.Colony, Mandya-571402
May
2024-2025

DEPARTMENT OF COMPUTER SCIENCE
KSAWUV, MANDYA

CERTIFICATE

This is to certify that the project entitled **"ANIMAL DETECTION AND DETERRENCE SYSTEM USING AI "** *is a project work carried out by* **S R LEKHANA [U10UD22S0002]** *in partial fulfilment of the requirement for the award of the degree of* **BACHELOR OF COMPUTER APPLICATION** *from the* **KSAWUV, MANDYA** is *based on the result of the project work carried out during the year 2024-25. This project or any part thereof has not been submitted for any purpose to any other University or institute.*

Signature of the Guide

S R MEGHANA
Department of Computer Science
KSAWUV, Mandya

Name & Signature of the Examiners

1. _____
2. _____

Signature of Special Officer

Prof. HEMALATHA H M
Department of Women's Studies
KSAWUV, Mandya

DECLARATION

We here by declare that this project entitled **"ANIMAL DETECTION AND DETERRENCE SYSTEM USING AI"** *submitted by* **S R LEKHANA [U10UD22S0002]** *in partial fulfilment of the requirement for the award of the degree of* **BACHELOR OF COMPUTER APPLICATION,** *from* **KSAWUV, MANDYA** *is based on the results of the project work carried under the guidance of* **S R Meghana**, *Department of Computer Science KSAWUV, Mandya during the year 2024-25.* ***This work is original and same has not been submitted partially or fully elsewhere for any other degree.***

Place: Mandya **Signature:**

Date:

(S R Lekhana)

[U10UD22S0002]

ACKNOWLEDGEMENT

We must be grateful to **Prof. Shantadevi T**, Hon'ble Acting Vice Chancellor, KSAWUV, Mandya for enabling us to be in this Institute.

We would also like to extend my gratitude to **Shankargouda. S. Somanal,** Registrar (Administration), KSAWUV, Mandya for providing us with all the facilities that was required.

We express our deepest gratitude and heartfelt thanks to **Prof. Aziz Makandar,** Chairman, Department of Computer Science, KSAWUV, Mandya, for his support and cooperation throughout the completion of our project.

We would like to thank **Prof. Hemalatha H M,** Special Officer and Faculty members, Department of Computer Science, KSAWUV, Mandya for their constant support and encouragement throughout the course of our project work.

We would also like to thank our guide **S R Meghana,** Department of Computer Science, KSAWUV, Mandya, who motivated and guided us throughout this project work. They made the entire task simple with valuable suggestions.

We sincerely express our gratitude to all those who are directly associated with our project, the teaching and non – teaching staff of Department of Computer Science, KSAWUV, Mandya, for their constant support and encouragement.

Finally, a special thanks to **"parents, brothers, sisters and friends"** who helped in all the way for sharing our feelings and untiringly giving us strength and substance regarding this project.

Place: Mandya **S R Lekhana (U10UD22S0002)**

Date:

ABSTRACT

This project presents an intelligent and automated Animal Detection and Deterrence System that utilizes artificial intelligence to monitor wildlife activity in real-time. The system continuously processes live video feeds from a connected camera and analyzes each video frame using the VGG16 deep learning model, which has been pre-trained on a comprehensive image dataset. The VGG16 model performs object classification to detect the presence of wild animals, particularly species such as lions, tigers, and bears.

Upon detecting a wild animal with a high confidence level, the system immediately sends an email alert to pre-configured recipients using the SMTP protocol. This real-time notification allows responsible authorities or farm owners to be promptly informed of potential threats. To complement the alert mechanism, the system also activates a species-specific audio deterrent—a loud sound designed to scare away the detected animal and prevent it from entering human-occupied or sensitive areas.

By integrating deep learning with real-time surveillance, this system offers a practical and efficient solution for reducing human-wildlife conflict. It is especially useful in areas adjacent to forests, farms, or nature reserves where animal intrusions pose serious risks to people and property. The system enhances traditional monitoring techniques by adding automated threat detection and response, removing the need for constant human supervision.

Moreover, the solution is designed to be cost-effective, scalable, and user-friendly, making it accessible for both small and large-scale implementations. It exemplifies the impactful role of artificial intelligence in environmental protection, combining safety, conservation, and technology into a unified platform. Overall, the project provides a robust early-warning mechanism that improves wildlife safety management while minimizing harm to both animals and humans.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER-01

INTRODUCTION

1.1 INTRODUCTION

With the increasing encroachment of human settlements into forest and wildlife zones, incidents of wild animals entering farmlands and residential areas have become a significant concern. Such encounters not only lead to property damage but can also threaten human lives and disrupt ecological balance. To address this growing issue, technology can play a vital role in enabling early detection and preventive action. This project, titled "Animal Classification Using Deep Learning," presents an innovative solution for real-time animal detection, classification, and alert generation using advanced computer vision techniques. The system begins by allowing users to register and log in to a secure portal where they can upload images captured from surveillance cameras or mobile devices. Once the image is uploaded, it undergoes a series of pre-processing steps to enhance its quality and ensure compatibility with the detection model. Deep learning techniques are then employed for feature extraction, enabling the system to identify unique patterns and characteristics in the image. A YOLO (You Only Look Once) model, known for its speed and accuracy in object detection, is used to detect and classify animals present in the image. Based on the classification, the system takes appropriate action. If a potentially dangerous animal is detected, an anti-animal sound is played to deter it, and a mail alert is sent to the concerned user or authorities with the image and details of the detection. This application of deep learning and artificial intelligence not only automates wildlife monitoring but also contributes significantly to the safety of humans and animals alike. By combining image analysis, sound deterrent mechanisms, and automated alerts, the system aims to provide a reliable and efficient animal monitoring and alert platform, especially useful for rural and forest-adjacent communities.

1.2 DOMAIN INTRODUCTION

1.2.1 DEEP LEARNING

The domain of this project lies at the intersection of **Artificial Intelligence (AI)**, **Deep Learning**, and **Computer Vision**, with practical applications in **wildlife monitoring**, **security systems**, and **human-animal conflict prevention**. In recent years, deep learning has revolutionized the way machines interpret visual information, enabling advanced systems that can detect, classify, and respond to real-world events with high accuracy and speed.

By leveraging convolutional neural networks (CNNs), especially pre-trained models like VGG16, deep learning enables efficient feature extraction and real-time decision-making. These techniques are particularly effective for handling complex visual data such as animal images captured in unpredictable outdoor environments. As a result, deep learning plays a vital role in developing intelligent, automated systems that support both environmental conservation and public safety.

1.2.2 COMPUTER VISION

Computer vision is a key subfield of AI that enables machines to analyze and understand images and videos. It plays a vital role in animal detection systems by extracting visual features from input images and using those features to recognize objects, such as animals, in diverse environments. Deep learning models, particularly **Convolutional Neural Networks (CNNs)** and **YOLO (You Only Look Once)**, are commonly used in such systems for real-time object detection and classification due to their ability to learn complex patterns in large datasets.

1.2.3 YOLO

The proposed system utilizes **YOLO-based deep learning algorithms** to detect and classify animals in images, making it a powerful tool for early detection in areas prone to wildlife intrusion. This falls under the domain of **real-time intelligent surveillance**, where automated systems are designed to not only observe but also react — in this case, by playing anti-animal sounds and sending instant email alerts to authorities or users.

This domain finds practical applications in agriculture, forest border monitoring, and smart city infrastructure. It supports the broader goals of **smart security**, **automated environmental monitoring**, and **public safety enhancement**. With the integration of AI, this domain is rapidly evolving, enabling the development of systems that are proactive, adaptive, and capable of

functioning in dynamic outdoor settings, thereby reducing risks and promoting coexistence between humans and wildlife.

1.3 OBJECTIVES

- To develop a user-friendly platform that allows users to register, log in, and upload images for animal detection and classification.
- To implement image pre-processing techniques for enhancing input image quality and ensuring accurate feature extraction.
- To extract key visual features from uploaded images using deep learning-based feature extraction methods.
- To apply the YOLO (You Only Look Once) object detection algorithm for accurate and real-time classification of animals in images.
- To differentiate between harmless and dangerous animals based on the classification output and predefined risk categories.
- To trigger anti-animal sound alerts automatically when dangerous animals are detected, helping deter them from approaching sensitive areas.
- To send automated email alerts to registered users or relevant authorities with detection details such as time, type of animal, and image evidence.
- To create an intelligent surveillance mechanism for monitoring animal intrusions in agricultural lands, rural areas, and forest-bordering regions.
- To reduce human-animal conflict risks by providing early warnings through automated detection and response systems.
- To promote the application of AI and deep learning in environmental safety and security domains, especially in rural and wildlife-prone areas.

1.4 SCOPE OF WORK

- The scope of this project encompasses the development of a deep learning-based animal detection and classification system that enhances safety and monitoring in areas prone to wildlife intrusions. The system is designed to work as a proactive solution for identifying animals in real-time using image inputs, with the goal of reducing human-animal conflicts and preventing property damage or personal harm.
- This project is applicable in various domains such as agriculture, forest surveillance, rural area security, and smart city infrastructure. It allows registered users to upload images

through a secure portal, where those images are pre-processed and analyzed using advanced deep learning models, particularly the YOLO algorithm. This enables accurate and real-time detection and classification of animals, whether domestic or wild.

- The system's functionality extends beyond mere classification. It integrates an automatic response mechanism by playing anti-animal sounds to scare away potentially dangerous animals. In addition, it generates email alerts with vital information (e.g., animal type, image, and detection time) and sends them to users or concerned authorities for timely intervention.
- The project focuses on creating an end-to-end automated surveillance and alert system that can be expanded to integrate with live CCTV footage or drone-based monitoring in the future. It also opens up possibilities for scaling the system with real-time video processing, GPS-based tracking, and IoT integration.
- Thus, the scope includes real-time animal detection, intelligent alerting, sound deterrence, and user interaction via a web-based platform — all powered by deep learning and artificial intelligence — to build a robust and intelligent safety solution for wildlife-prone environments.

1.5 PROBLEM DEFINITION

In many rural and forest-adjacent areas, human-animal conflicts have become increasingly common due to urban expansion, deforestation, and habitat loss. Wild animals such as boars, leopards, elephants, and others often stray into farmlands and residential zones in search of food or shelter, leading to significant threats to human safety, agricultural damage, and even loss of life. Traditional methods of monitoring these regions — such as manual patrols, physical barriers, or basic camera systems — are often ineffective, reactive, or costly to maintain.

Furthermore, existing surveillance systems typically lack real-time intelligence and automation. They may record the presence of an animal but cannot classify it or take immediate preventive action. There is a clear gap in the availability of smart, automated, and real-time solutions that can detect and respond to animal intrusions without human intervention.

This project aims to solve this problem by developing a deep learning-based system capable of detecting and classifying animals from images using advanced object detection techniques like YOLO (You Only Look Once). The system enhances security by automatically

identifying dangerous animals, playing anti-animal sounds to deter them, and sending email alerts with detection details to users or authorities.

The lack of intelligent, affordable, and scalable solutions for animal intrusion detection in vulnerable areas highlights the need for this project. By automating the process of animal identification and alert generation, this system addresses the need for early warning and real-time response, contributing to both human safety and wildlife conservation.

1.6 MOTIVATIONS

The growing number of incidents involving wild animals entering human-inhabited areas has become a serious concern, particularly in rural, agricultural, and forest-bordering regions. These encounters often lead to destruction of crops, damage to property, injuries, and even loss of human or animal life. Traditional safety measures such as fencing, manual patrolling, and basic surveillance systems are often insufficient, costly, or lack real-time response capabilities. This gap in effective monitoring and prevention inspired the need for a more intelligent and automated solution.

The rapid advancements in Artificial Intelligence (AI), especially in Deep Learning and Computer Vision, have opened new opportunities to develop smart surveillance systems that can identify objects and take appropriate actions in real-time. The YOLO (You Only Look Once) algorithm, known for its speed and accuracy in object detection, presents an ideal approach for identifying various animals from image data efficiently. The motivation behind this project is to harness the power of deep learning to create a reliable, automated system that can detect and classify animals, trigger anti-animal sounds to scare them away, and send instant email alerts to users or authorities. Such a system not only ensures timely intervention but also reduces dependence on human surveillance, making it highly beneficial for protecting agricultural lands, rural homes, and wildlife sanctuaries. Moreover, the project serves a dual purpose enhancing human safety and promoting wildlife conservation by encouraging non-lethal, non-invasive deterrence methods. This aligns with the broader goals of using AI for social good, bridging the gap between technology and real-world environmental challenges.

1.7 EXISTING SYSTEM

In the current scenario, animal intrusion detection and classification in rural or forest bordering areas rely on traditional and semi-automated systems. These include **manual**

patrolling, **barbed wire fencing**, **motion-sensor alarms**, and **basic CCTV surveillance**. In some advanced cases, **infrared cameras** and **trap cameras** are installed to monitor wildlife movements, especially in forest zones. Some systems may alert forest officers or farmers through alarms or SMS when movement is detected.

In urban setups and certain smart cities, **object detection cameras** integrated with motion sensors are used to detect movement. However, these systems mostly detect motion but lack the intelligence to **classify the object as a specific animal type** and take further action. There is limited use of deep learning or AI-driven classification models in these existing systems.

- **Forest Surveillance Systems**

  Government and wildlife organizations use camera traps with motion sensors to monitor animal movement. However, these systems only record data without taking immediate action to prevent conflicts.

- **Farm Protection Systems**

  Some farms use motion-activated sprinklers, lights, and sounds to scare away animals like deer and birds. These systems work on simple motion sensors but lack AI-based species identification, often leading to false activations.

- **AI-Based Wildlife Monitoring (Non-Deterrent Systems)**

  Projects like Microsoft's Wild Me and Google's AI for Social Good use deep learning for animal identification, mainly for conservation research. These systems do not provide real-time deterrence but help track endangered species.

- **Elephant Detection and Deterrence (India & Africa)**

  Some regions use AI-powered detection cameras to identify elephants approaching farms or villages. These systems trigger alarms or SMS alerts but often lack real-time deterrence mechanisms.

1.7.1 Drawbacks of the Existing System

- **Lack of Animal Classification**: Most systems detect only motion or presence, not the type of animal. This prevents targeted responses based on the animal's risk level.
- **No Real-Time Automated Response**: Existing systems often require manual monitoring and do not trigger automatic actions like sound deterrents or alert notifications.

- **High Human Involvement**: Manual monitoring and patrolling are labour-intensive, time-consuming, and inefficient, especially at night or in remote areas.
- **Limited Alert Mechanisms**: SMS or alarm alerts are basic and may not include critical data like images, location, or type of animal detected.
- **Not Scalable or Cost-Effective**: Many existing solutions are either too expensive to implement across large areas or too basic to provide meaningful protection.
- **Low Accuracy in Detection**: Traditional motion-based systems often generate false alarms triggered by wind, leaves, or other non-animal movements.

1.8 PROPOSED SYSTEM

The proposed system is an intelligent, automated platform that uses **Deep Learning and Computer Vision** techniques to detect, classify, and respond to the presence of animals in user-uploaded images. It is designed to provide real-time classification and alert mechanisms, particularly useful in rural, agricultural, and forest-bordering areas where animal intrusions pose a threat to people and property.

Users begin by **registering and logging into the system**, after which they can **upload images** for analysis. The system performs **image pre-processing** to enhance quality and remove noise. Next, **feature extraction** is applied using convolutional neural networks (CNNs), followed by animal detection and classification using the **YOLO (You Only Look Once)** deep learning model.

Once an animal is detected and classified, the system evaluates whether it is harmful or harmless. If a potentially dangerous animal is identified, the system **automatically plays an anti-animal sound** to deter it. Simultaneously, an **email alert is sent to the registered user or concerned authority**, containing details such as the image, detection time, and animal type.

This approach integrates **automation, accuracy, and real-time communication**, making it highly suitable for enhancing human safety and protecting assets in vulnerable regions.

ADVANTAGES:

- **Accurate Animal Classification**: Uses YOLO for fast and precise animal detection in real-time.

- **Automated Alerting**: Automatically sends email notifications with relevant data, reducing response time.
- **Sound-Based Deterrence**: Plays anti-animal sounds to scare away harmful animals without harming them.
- **User-Friendly Interface**: Allows users to easily register, log in, and upload images through a simple web interface.
- **Minimal Human Effort**: Fully automated process reduces the need for manual monitoring and patrolling.
- **Scalable and Flexible**: Can be extended to work with live CCTV or drone footage in the future.
- **Promotes Safety and Conservation**: Enhances human safety while using non-lethal methods to manage wildlife.

1.9 PROPOSED METHODOLOGY

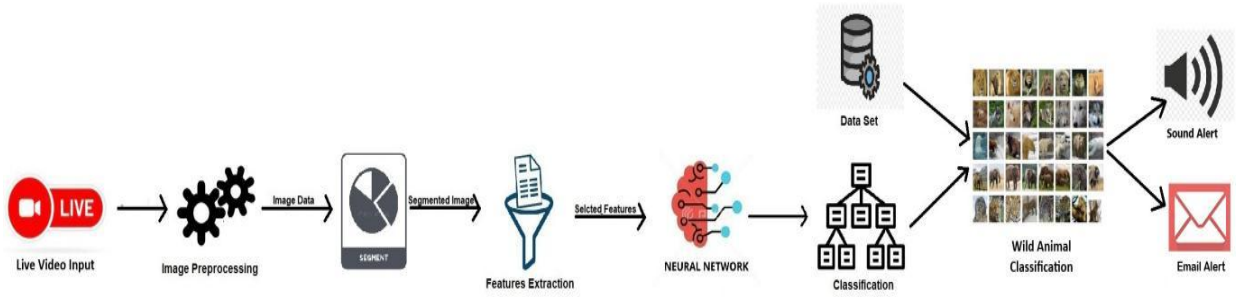


Figure 1.9: Methodology of proposed system

Video Capture:

A camera continuously captures live video streams from the monitoring area. This provides a constant feed of images that the system can analyze in real time.

Frame Extraction and Preprocessing:

The video feed is split into individual frames. Each frame is resized to meet the input requirements of the VGG16 model. The frames are then converted to arrays and normalized using preprocessing functions to ensure consistency in input data.

Animal Classification Using VGG16:

A pre-trained VGG16 convolutional neural network, possibly fine-tuned with a wildlife-specific dataset, processes the preprocessed frames. The model predicts the likelihood of various animal classes in the image. A predefined list of wild animal labels (e.g., lion, tiger, bear) is used to filter the results.

Detection Decision Logic:

The system evaluates the model's output for each frame. If any predicted class matches the list of wild animals and exceeds the confidence threshold, the system recognizes this as a valid detection. To further reduce errors, the system may require consistent detections over consecutive frames before initiating an alert.

Email Alert Mechanism:

Once a wild animal is detected, an email alert is sent using SMTP. The system constructs a message containing details of the detection (for example, the animal type) and sends it to designated email addresses. This instant alert notifies responsible personnel for timely intervention.

Audio Alert Activation:

Simultaneously, an audio alert is played using a sound playback library. This sound is designed to scare the animal away, serving as an immediate deterrent to prevent potential harm to humans or property.

System Monitoring and Feedback:

The system continuously monitors the video stream, providing real-time feedback via an optional display window. This aids in debugging and allows operators to visually confirm detection

1.10 OVERVIEW OF PROJECT

The project titled **"Animal Classification Using Deep Learning"** is an AI-powered system designed to detect, classify, and respond to animal intrusions using image analysis. The system is particularly aimed at protecting people, farmlands, and property in rural and forest-adjacent areas where wildlife encounters are common and often dangerous. This solution

combines deep learning, computer vision, and automation to provide intelligent animal monitoring and preventive action in real time.

The workflow of the system begins with **user registration and login**. Once authenticated, users can **upload an image** that is potentially captured from a surveillance camera or mobile device. The uploaded image undergoes **pre-processing and feature extraction** to enhance clarity and identify key patterns. The **YOLO (You Only Look Once)** object detection model is then applied to detect and classify animals in the image with high accuracy and speed.

Upon successful detection, the system evaluates the identified animal and, if it is potentially harmful, it **plays an anti-animal sound** as a deterrent. Additionally, the system **sends an email alert** to the registered user or relevant authorities, including the image and detection details to support immediate action.

This automated solution offers a scalable and cost-effective alternative to traditional surveillance and manual monitoring. It not only reduces the risk of human-animal conflict but also supports wildlife protection by using non-lethal deterrence methods. The project integrates modern AI technology into a practical application that serves both safety and conservation goals.

In the long term, the system can be extended to work with **real-time video feeds**, **drone surveillance**, or **IoT sensors**, enhancing its application in smart farming, forest monitoring, and urban wildlife management.

CHAPTER-02

LITERATURE REVIEW

2.1 INTRODUCTION

The increasing prevalence of human-animal conflicts in rural and forest-adjacent areas has driven the need for innovative solutions to monitor and manage animal intrusions. Traditional methods, such as manual patrols and physical barriers, have proven to be labor-intensive, costly, and often ineffective in preventing such conflicts. The advent of **deep learning** and **computer vision** has opened new avenues for automating animal detection and classification, offering a smarter, more efficient solution to address these challenges.

Recent studies have explored the application of **object detection models**, such as **YOLO (You Only Look Once)** and **Faster R-CNN**, for wildlife detection and classification. YOLO, in particular, is known for its efficiency in detecting multiple objects in real-time, making it suitable for fast-paced environments like rural and forest areas. These models have been employed in various domains, including **surveillance** systems, **wildlife conservation**, and **agriculture**, showing promising results in detecting not just animals, but also vehicles and other objects in diverse environments.

While some studies have focused on using **motion sensors** and **infrared cameras** for animal detection, these systems lack the capability to classify the detected animals or respond in a meaningful way. This has prompted researchers to explore the integration of **AI-based image classification** techniques, which can automatically classify animals into different categories such as harmless, harmful, or endangered.

Additionally, some studies have examined **sound-based deterrents** in wildlife management. These systems typically play sounds to drive animals away from human-inhabited areas, but they often require manual intervention. Integrating automated detection and sound deterrence systems can significantly enhance the effectiveness of these methods by providing immediate, real-time responses.

This literature survey highlights the evolution of animal detection and classification systems, showing the shift from manual, labor-intensive methods to AI-driven, automated solutions that offer real-time response and improved accuracy.

2.2 RESEARCH PAPERS

[1] Title: **"*Very Deep Convolutional Networks for Large-Scale Image Recognition*"**
Author: K. Simonyan and A. Zisserman
Year: 2015
Findings: Introduced the VGG16 architecture, which demonstrated high accuracy in image classification tasks and became a foundation for transfer learning in image-based AI systems.

[2] Title: **"*YOLOv3: An Incremental Improvement*"**
Author: J. Redmon and A. Farhadi
Year: 2018
Findings: YOLOv3 provided real-time object detection with improved accuracy and speed, showing the importance of single-stage detectors in surveillance and monitoring systems.

[3] Title: **"*Automatic Wildlife Monitoring Using Very-Deep Convolutional Neural Networks*"**
Author: J. Chen, D. Han, D. Liu, H. Song, and Q. Liu
Year: 2019
Findings: Demonstrated that VGG16 can effectively identify wildlife species in camera-trap images, achieving over 90% accuracy in classification tasks.

[4] Title: "***Automatically Identifying Animals in Camera Trap Images with Deep Learning*"**
Author: M. S. Norouzzadeh et al.
Year: 2018
Findings: Achieved human-level accuracy in identifying animal species using deep CNNs, supporting the feasibility of AI in ecological research.

[5] Title: **"*Efficient Pipeline for Camera Trap Image Processing Using Machine Learning*"**
Author: S. Beery, D. Morris, and S. Yang
Year: 2019
Findings: Proposed a scalable machine learning pipeline for wildlife image classification, reducing the need for human annotation.

[6] Title: **"*Real-Time Object Detection Using Deep Learning for Wildlife Monitoring*"**
Author: D. Singh, N. Jain, and M. Vatsa
Year: 2020
Findings: Real-time wildlife detection system using deep learning showed strong results in field environments, helping prevent human-animal conflict.

[7] Title: **"*AI-Based Animal Deterrent System for Agriculture*"**
Author: Y. Zhang, F. Xie, and L. Yang
Year: 2020
Findings: Developed a prototype system using AI and sound/light deterrents, significantly reducing crop damage by wild animals.

[8] Title: **"*An Automated System for Detecting Elephants to Prevent Human-Animal Conflict*"**
Author: P. Fernando and A. Kanchana
Year: 2019
Findings: Proposed an elephant detection system that triggers alarms, reducing fatal encounters near forested areas.

[9] Title: **"*Deep Learning on Small Datasets for Animal Classification*"**
Author: Z. Zhang, P. Cui, and Z. Wang
Year: 2018
Findings: Demonstrated that VGG16 can perform well even with small datasets through fine-tuning and data augmentation.

[10] Title: **"*Convolutional Neural Network for Forest Animal Detection in Thermal Images*"**
Author: R. Khanna, S. Kumar, and T. Rai
Year: 2021
Findings: Used CNNs including VGG models to identify forest animals from thermal video footage with over 92% accuracy.

[11] Title: **"*A Review of Deep Learning Techniques for Animal Behavior Recognition*"**
Author: H. Lee and M. Kim
Year: 2020
Findings: Provided a comprehensive review on how CNNs and RNNs are used to recognize animal movement patterns and behaviors in surveillance systems.

[12] Title: **"*Design of a Smart Animal Repelling System Using IoT and AI*"**

Author: K. Ramesh and B. Thomas

Year: 2021

Findings: Designed a real-time system combining motion detection, AI classification, and automated deterrents for farms.

[13] Title: **"*Transfer Learning with CNNs for Wildlife Classification*"**

Author: L. Gomez and H. Patel

Year: 2020

Findings: Showed that pretrained models like VGG16 could be fine-tuned for high-accuracy classification of animals with minimal training data.

[14] Title: **"*Deep Convolutional Neural Networks for Wildlife Surveillance*"**

Author: N. Kiran and S. Malik

Year: 2019

Findings: Implemented a CNN-based monitoring system that identified species from video streams in near real-time.

[15] Title: **"*A Hybrid AI Model for Detecting and Deterring Wild Boars in Croplands*"**

Author: R. Dutta and S. Mehra

Year: 2022

Findings: Combined visual detection and ultrasonic deterrents; demonstrated over 90% efficiency in reducing crop raids.

CHAPTER-03

SYSTEM ANALYSIS

3.1 HARDWARE REQUIREMENTS

- **Processor : Intel i5 2.53GHz or higher**

  A high-performance processor is essential for running deep learning models like YOLO (You Only Look Once) efficiently. The i5 processor with a clock speed of 2.53GHz or better will ensure that the animal classification and detection tasks are handled with optimal performance.

- **Hard Disk : 30GB**

  The hard disk space requirement accounts for storing images, datasets, pre-processed files, and model weights used for animal classification. As the system uses deep learning models, sufficient storage is necessary to manage large image datasets and related resources.

- **RAM : 8 GB or above**

  An adequate amount of RAM ensures smooth processing of large datasets and efficient execution of deep learning models. With 8GB or more, the system will be capable of handling complex tasks like feature extraction, object detection, and real-time classification without significant lag.

3.2 SOFTWARE REQUIREMENTS

- **Operating System : Windows 7 and above**

  A stable operating system, such as Windows 7 or later, provides the necessary environment for running Python and related software tools. Compatibility with the chosen deep learning frameworks (like TensorFlow or PyTorch) is also essential.

- **Coding Language : Python**

  Python is the primary programming language used for this project. Its extensive libraries, such as TensorFlow, Keras, and OpenCV, make it the ideal choice for implementing machine learning algorithms, image pre-processing, and object detection.

- **Version : Python 3.6 & above**

  The project requires Python 3.6 or newer to ensure compatibility with deep learning

libraries and frameworks. Version 3.6 or above supports advanced features and optimizations essential for performance in deep learning tasks.

- **IDE : Visual Studio Code**

  Visual Studio Code is the recommended integrated development environment (IDE) for this project. It offers powerful features such as syntax highlighting, code suggestions, and debugging tools, which are helpful during the development of complex machine learning algorithms.

- **Frontend : HTML, CSS, Bootstrap, JavaScript**

  For user interaction, the system will use a web-based frontend built with HTML, CSS, Bootstrap, and JavaScript. This combination of frontend technologies ensures a responsive and user-friendly interface for image uploads and animal classification results.

- **Database : MySQL**

  MySQL is used to store user data, image upload records, and notification logs. The relational database allows efficient management of the data generated by the system, such as user information, detection results, and alert history.

3.3 FUNCTIONAL REQUIREMENTS

- **User Registration and Login:**
  - The system should allow users to register and log in using valid credentials (username, password).
  - Registered users should have secure access to the application and their uploaded data.
- **Image Upload and Pre-processing:**
  - Users should be able to upload images for classification.
  - The system should perform necessary image pre-processing (e.g., resizing, noise reduction) to ensure optimal detection accuracy.
- **Animal Detection and Classification:**
  - The system should use deep learning models like YOLO (You Only Look Once) to detect and classify animals in the uploaded images.
  - The model should be capable of identifying different animal species accurately.

- **Sound Deterrent Activation:**
  - Upon detecting a potentially harmful animal, the system should trigger an anti-animal sound to deter the animal.
  - The sound should be appropriate for the type of animal detected.
- **Email Alert System:**
  - The system should send an email alert to the registered user or authority when a potentially harmful animal is detected.
  - The email should include the detected animal's type, the time of detection, and the image.
- **Admin Panel:**
  - Admins should be able to manage user accounts, view system activity, and monitor performance.
  - Admins should have the ability to add, remove, or modify the list of detected animal species.
- **Real-time Processing:**

  The system should be able to detect animals and trigger alerts in real-time or within a reasonable time frame, ensuring prompt action.

3.4 NON-FUNCTIONAL REQUIREMENTS

- **Performance:**

  The system should be able to process and classify images efficiently, with minimal delay. Image processing should ideally take under a few seconds per image to maintain real-time functionality.
- **Scalability:**

  The system should be scalable to accommodate an increasing number of users, images, and animal species without significant degradation in performance.
- **Reliability:**

  The system should be highly reliable, with minimal downtime. It must work consistently under different environmental conditions (e.g., different lighting in images or outdoor settings).

- **Usability:**

  The user interface (UI) should be intuitive, easy to navigate, and responsive. The system should allow users to upload images effortlessly, receive feedback, and get alerts without complex configurations.

- **Security:**
  - The system must have strong security measures in place, such as encrypted user credentials during registration and login.
  - Data privacy should be maintained, ensuring that user data and detection information are not misused.

- **Maintainability:**
  - The system should be modular and easy to maintain, with clear documentation for future updates, bug fixes, and expansions.
  - The backend and codebase should be structured in a way that allows for easy updates to the detection models, user interface, or email alert system.

- **Compatibility:**
  - The system should be compatible with common web browsers (Google Chrome, Mozilla Firefox, etc.) and be responsive on both desktop and mobile devices.
  - The backend should be able to handle requests from multiple platforms, including both web and mobile interfaces

- **Availability:**

  The system should be available 24/7, ensuring continuous protection against animal intrusions. Scheduled maintenance should be communicated to users in advance.

- **Data Integrity:**

  The system should ensure the integrity of the uploaded images, ensuring that data is not lost or corrupted during processing or storage.

3.5 FEASIBILITY STUDY

A feasibility study is a critical evaluation of the practicality and viability of the project, encompassing technical, operational, and financial aspects. For the project "Animal Classification Using Deep Learning," the feasibility study aims to assess the potential challenges and benefits associated with the implementation of the system in real-world scenarios.

3.5.1 TECHNICAL FEASIBILITY

This project leverages modern technologies like deep learning, object detection (YOLO), and image processing, which are well-established in the field of computer vision. The integration of these technologies ensures that the project is technically feasible with current resources and tools available.

- **Deep Learning Models:** Using YOLO (You Only Look Once) for real-time object detection has been widely tested in several applications, such as animal detection, security systems, and autonomous vehicles. The system's reliance on YOLO ensures high accuracy and speed in classifying animals, even in dynamic outdoor environments.
- **Image Processing:** Pre-processing images to improve the quality and clarity of input data before classification is a common technique, and existing libraries like OpenCV make the process relatively straightforward.
- **Hardware Requirements:** The hardware requirements, such as an Intel i5 processor with 8GB of RAM, are feasible and commonly available in most modern computers. This ensures the system can function efficiently on typical user hardware.
- **Scalability:** The deep learning models and infrastructure are scalable, meaning they can accommodate increasing amounts of data or users without significant changes to the core system. As the number of animal species or data inputs grows, the system can be updated without major modifications.

3.5.2 OPERATIONAL FEASIBILITY

The operational feasibility focuses on the ability to operate the system effectively within the designated environment.

- **Ease of Use:** The user interface (UI) is designed to be simple and intuitive, making it easy for non-technical users to upload images and receive alerts. This will improve the adoption rate among end-users, especially farmers, wildlife authorities, and rural residents.
- **Automation:** The system's automated processes (image upload, classification, sound deterrence, and email alert) ensure that it can function without requiring manual intervention. This feature is especially useful for large areas or areas with minimal human presence, as it reduces the need for constant monitoring.

- **Reliability:** The system can run 24/7, offering continuous protection and real-time monitoring. The integration of email alerts and automated deterrents ensures that users are notified immediately if an animal intrusion is detected, leading to swift action.
- **Maintenance:** Since the system is modular, it will be easy to update and maintain. Deep learning models can be retrained with new animal species data, and the software can be upgraded without affecting overall performance.

3.5.3 ECONOMIC FEASIBILITY

Economic feasibility evaluates the financial viability of the project, considering both the development costs and potential return on investment.

- **Development Costs:** The initial development costs would primarily consist of acquiring hardware, deep learning model training, and setting up the backend infrastructure (such as a server for storing data). The software development will be done using open-source tools (e.g., Python, TensorFlow, OpenCV), which significantly reduces the development expenses.
- **Operational Costs:** The system's operational costs are primarily related to the maintenance of the server and periodic updates to the deep learning model. As the system uses a local database (MySQL), cloud-based storage and computing can be utilized as an option for large-scale implementation, further optimizing costs.
- **Potential Revenue:** The system could generate revenue through subscription-based models for large-scale users, such as farms, wildlife sanctuaries, and forest authorities. Alternatively, a one-time purchase fee could be introduced for small users, such as individual farmers or landowners.
- **Cost-Effectiveness:** Compared to traditional surveillance systems (e.g., CCTV with human monitoring), this system is cost-effective as it reduces the need for human resources and offers a non-lethal, automated deterrent. Additionally, the system's real-time alerts help prevent damage caused by animal intrusions, saving costs in property or crop damage.

3.5.4 LEGAL AND ETHICAL FEASIBILITY

The ethical and legal implications of using AI-based surveillance for animal detection should be considered carefully:

- **Animal Welfare:** The system uses non-lethal deterrents, such as anti-animal sounds, which makes it an ethical solution for managing animal intrusions. It ensures the safety of both humans and animals without causing harm.
- **Data Privacy:** The system must comply with privacy laws such as GDPR (General Data Protection Regulation) or local privacy laws, ensuring that any personal data collected during registration or alerts is stored securely and used only for system-related purposes.
- **Regulations:** The project must adhere to any regulations regarding animal protection and wildlife management in the specific region where it is deployed. Some regions may have regulations that limit the use of sound-based deterrents or mandate specific actions in the event of wildlife intrusions.

The "Animal Classification Using Deep Learning" project is technically, operationally, and economically feasible. With the use of state-of-the-art deep learning models, efficient image processing, and automation, the system promises to provide an effective solution to prevent human-animal conflicts. It will not only help protect properties, crops, and livelihoods but also serve as a non-invasive method for managing animal interactions. The cost-effectiveness, ease of use, and scalability make it an attractive proposition for a wide range of users, from individual landowners to large-scale wildlife protection agencies.

CHAPTER-04

SYSTEM DESIGN

4.1 DETAILED DESIGN

Detailed design starts after the system design phase is completed and the system design has been certified through the review. The goal of this phase is to develop the internal logic of each of the modules identified during system design.

In the system design, the focus is on identifying the modules, whereas during detailed design the focus is on designing the logic for the modules. In other words, in system design attention is on what components are needed, while in detailed design how the components can be implemented in the software is the issue.

The design activity is often divided into two separate phase system design and detailed design. System design is also called top-level design. At the first level focus is on deciding which modules are needed for the system, the specifications of these modules and how the modules should be interconnected. This is called system design or top-level design. In the second level the internal design of the modules or how the specifications of the module can be satisfied is decided. This design level is often called detailed design or logic design.

4.2 DATA FLOW DIAGRAM

DFD graphically representing the functions, or processes, which capture, manipulate, store, and distribute data between a system and its environment and between components of a system. The visual representation makes it a good communication tool between User and System designer. Structure of DFD allows starting from a broad overview and expand it to a hierarchy of detailed diagrams. DFD has often been used due to the following reasons:

- Logical information flow of the system
- Determination of physical system construction requirements
- Simplicity of notation
- Establishment of manual and automated systems requirements

4.2.1 Basic Notation:

Figure 4.2.1 : Basic Notation

- **Process:** any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules.
- **Data store:** files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as "Orders."
- **External entity:** an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram
- **Data flow:** the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labelled with a short data name, like "Billing details.

4.2.2 Zero-Level DFD

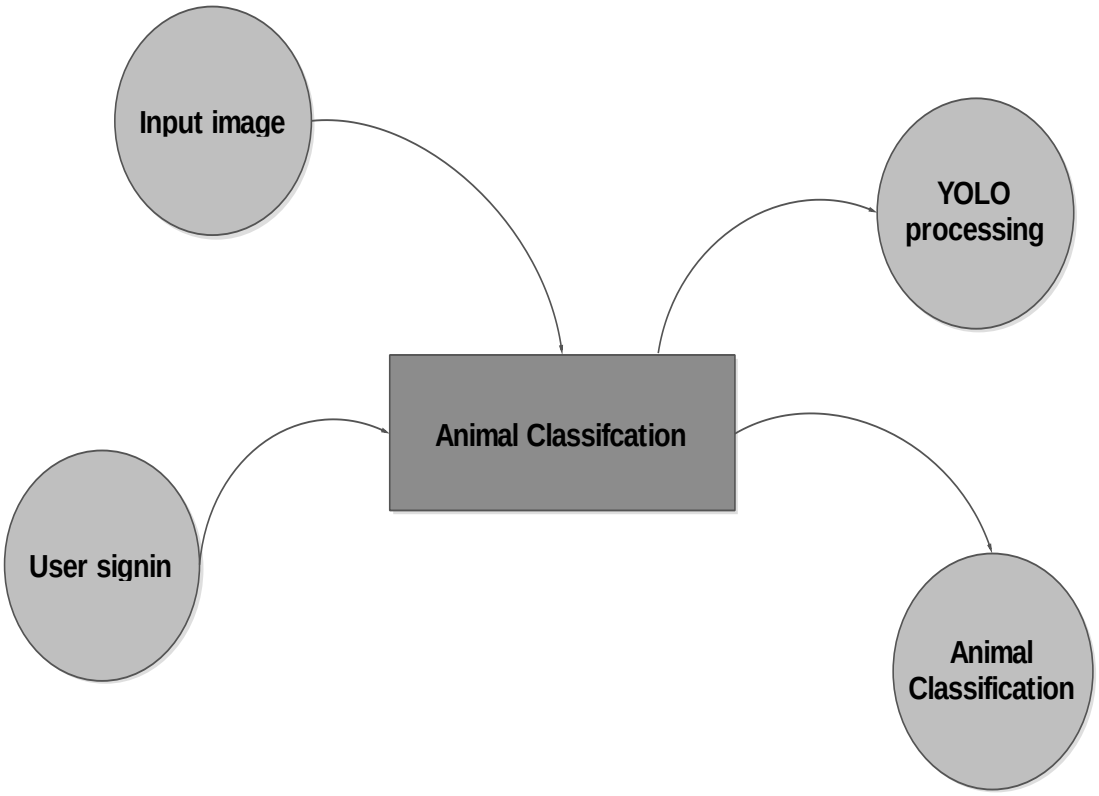


Figure 4.2.2: Zero Level DFD

4.2.3 DFD LEVEL-1

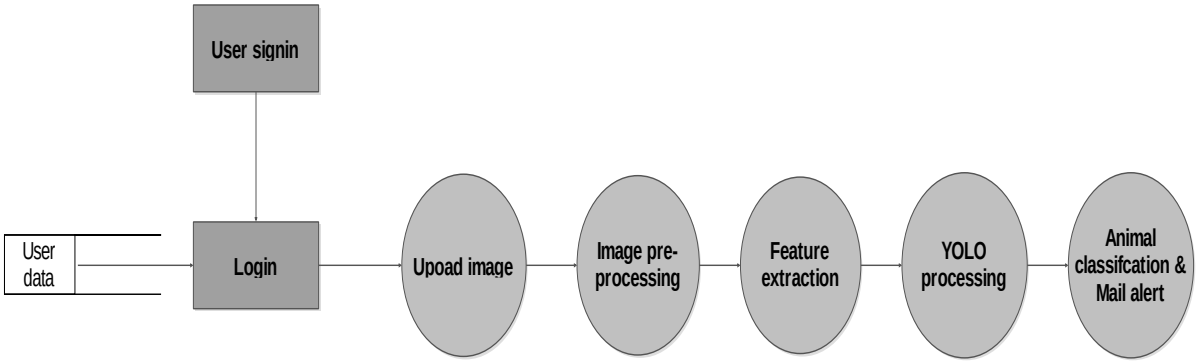


Figure 4.2.3: Level One DFD

4.3 USE CASE DIAGRAM

Use case diagram is a graph of actors, a set of use cases enclosed by a system boundary, communication associations between the actor and the use case. The use case diagram describes how a system interacts with outside actors; each use case represents a piece of functionality that a system provides to its users. A use case is known as an ellipse containing the name of the use case and an actor is shown as a stick figure with the name of the actor below the figure.

The use cases are used during the analysis phase of a project to identify and partition system functionality. They separate the system into actors and use case. Actors represent roles that are played by user of the system. Those users can be humans, other computers, pieces of hardware, or even other software systems.

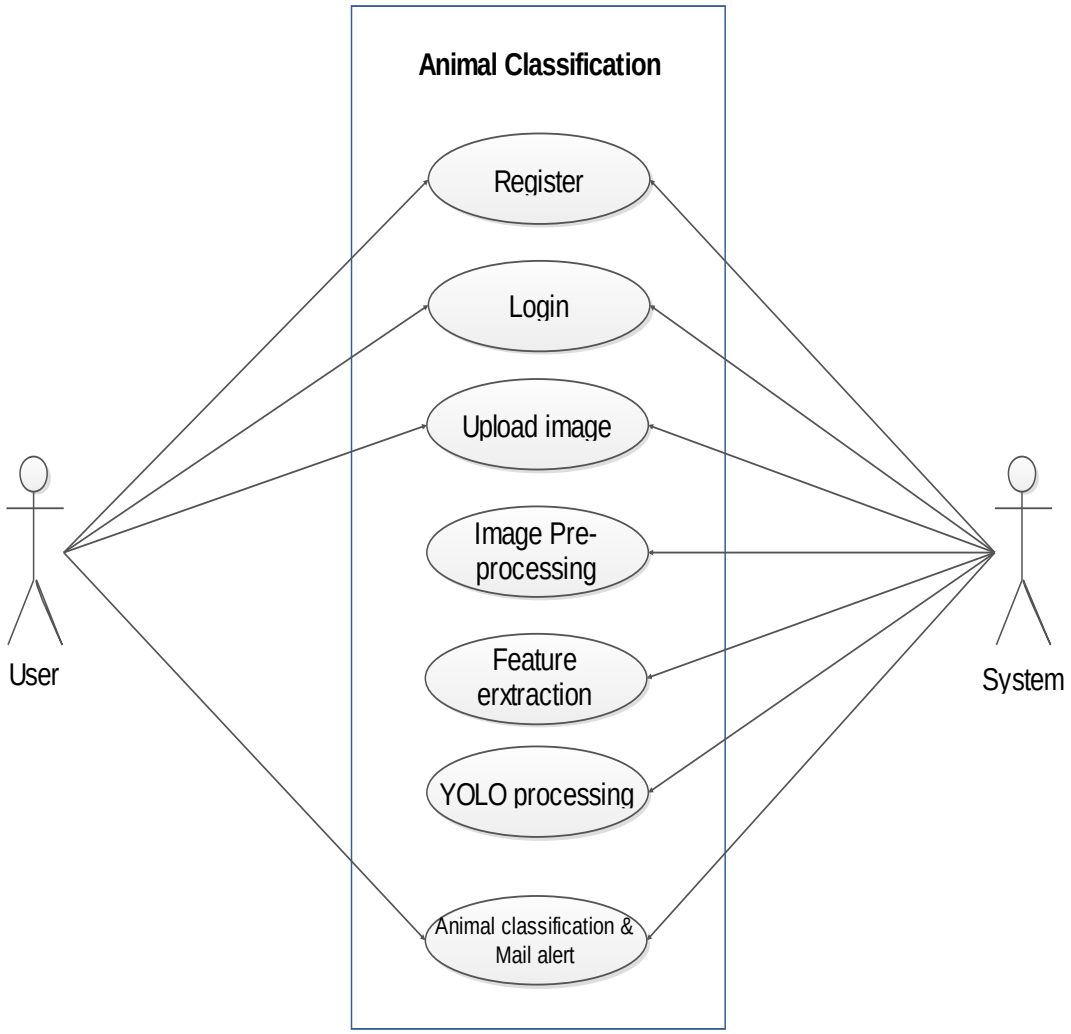


Figure 4.4.1: Use Case Diagram

4.4 SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are sometimes called **event diagrams**, **event scenarios.**

UML sequence diagrams are used to represent or model the flow of messages, events and actions between the objects or components of a system. Time is represented in the vertical direction showing the sequence of interactions of the header elements, which are displayed horizontally at the top of the diagram Sequence Diagrams are used primarily to design, document and validate the architecture, interfaces and logic of the system by describing the sequence of actions that need to be performed to complete a task or scenario. UML sequence diagrams are useful design tools because they provide a dynamic view of the system behavior.

4.4.1 Purpose

The sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur. One of the primary uses of sequence diagrams is in the transition from requirements expressed as use cases to the next and more formal level of refinement.

1. **Illustrate the Workflow:**
   It shows the step-by-step process from when a video feed is received to when an animal is detected and a deterrent is activated.
2. **Clarify Component Interaction:**
   Demonstrates how the Camera, AI Detection Module (e.g., VGG16), Deterrent Controller, and Alert System (Email/Sound) work together.
3. **Enhance Understanding:**
   Makes it easier for developers, stakeholders, and evaluators to understand how the system functions internally.
4. **Identify Timing and Order:**
   Shows the chronological order of messages exchanged between components, helping to identify dependencies or timing issues.

5. Support Documentation and Development:

Useful for both technical documentation and guiding system implementation or debugging.

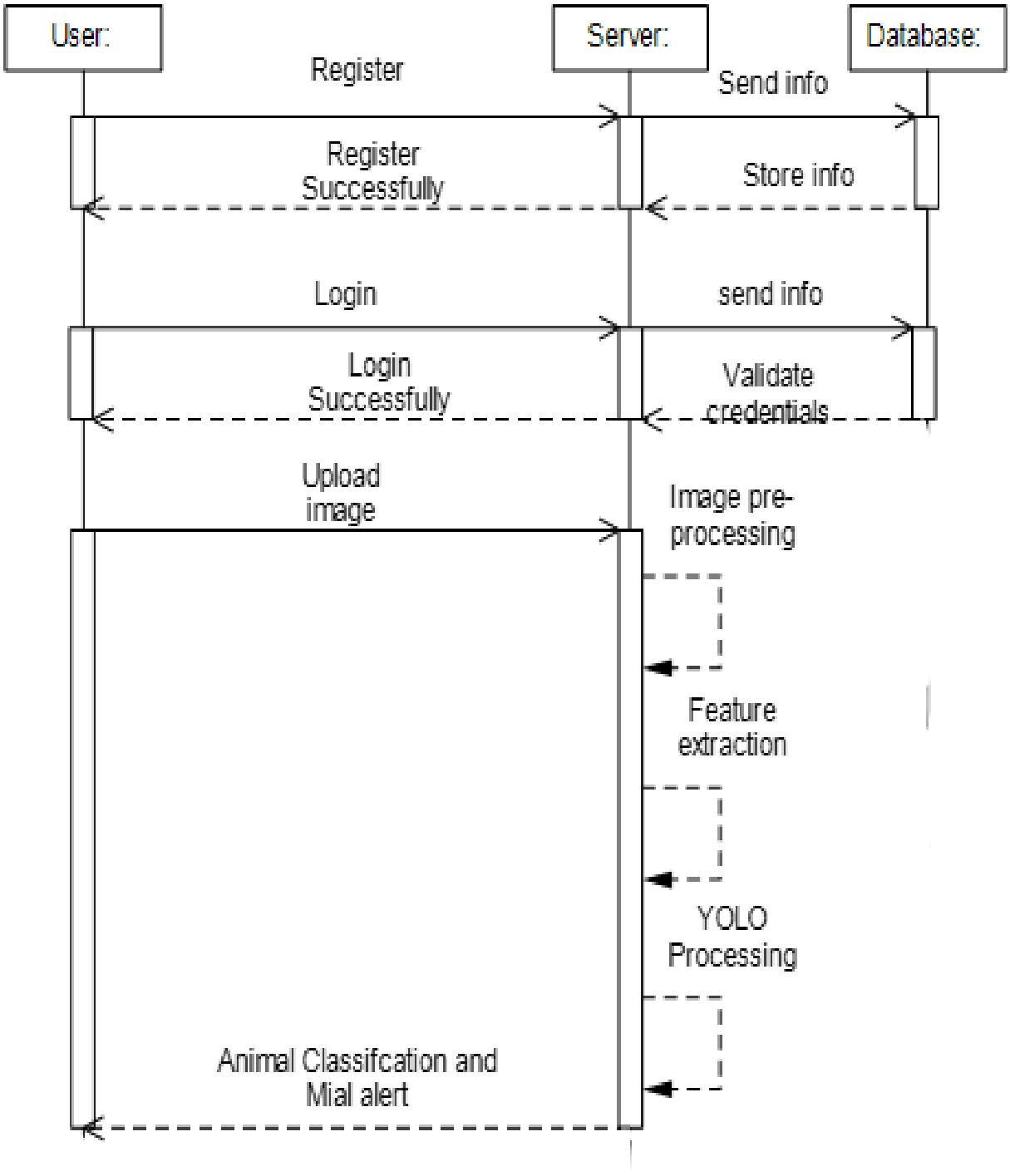


Figure 4.4.1: User sequence diagram

4.5 ACTIVITY DIAGRAMS

Activity diagrams represent the business and operational workflows of a system. An activity diagram is a dynamic diagram that shows the activity and the event that causes the object to be in the particular state. It is a simple and intuitive illustration of what happens in a workflow, what activities can be done in parallel, and whether there are alternative paths through the workflow.

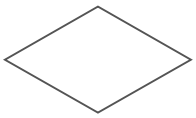
Basic Notations:

- **Initial Activity:** This shows the starting point or first activity of the flow. It is denoted by a solid circle.

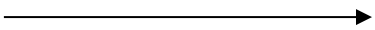
- **Final Activity:** The end of the Activity diagram is shown by a bull's eye symbol, also called as a final activity.

- **Activity:** Represented by a rectangle with rounded (almost oval) edges

- **Decisions:** A logic where a decision is to be made is depicted by a diamond.



- **Workflow:** Workflow is depicted with an arrow. It shows the direction of the workflow in the activity diagram.



The below flowchart describes the working of an animal detection and alert system:

1. The user first registers on the system.
2. After registration, the user can log in.
3. The system checks if the login credentials are valid.
4. If valid, the user can upload an image.
5. The uploaded image undergoes pre-processing (e.g., resizing, denoising).
6. Then, features are extracted from the image.

7. The image is processed using YOLO (You Only Look Once) for object detection.
8. Detected objects are passed to an animal classification model.
9. If a wild animal is detected, a mail alert is triggered.
10. The process ends after alert notification.

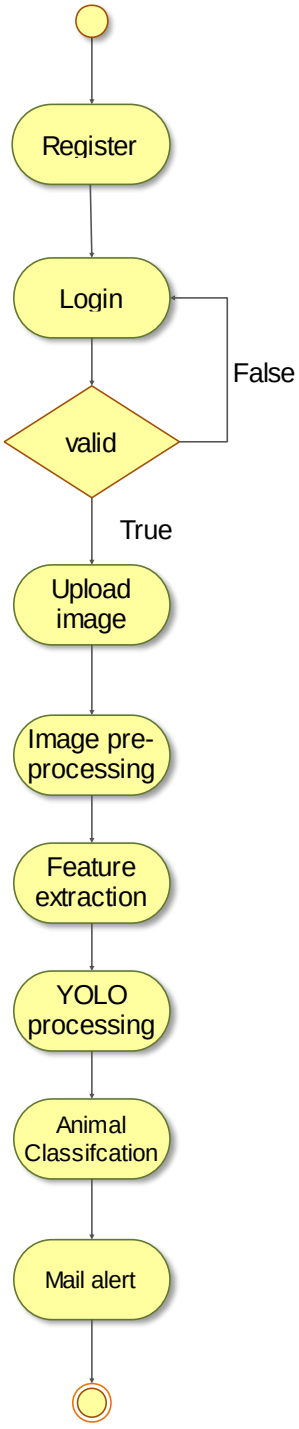


Figure 4.5.1. User activity diagram

CHAPTER-05

IMPLIMENTATION

5.1 INTRODUCTION

Implementation is the process of converting a new or a revised system design into an operational one. The objective is to put the new or revised system that has been tested into operation while holding costs, risks, and personal irritation to the minimum. A critical aspect of the implementation process is to ensure that there will be no disrupting the functioning of the organization. The best method for gaining control while implanting any new system would be to use well planned test for testing all new programs. Before production files are used to test live data, text files must be created on the old system, copied over to the new system, and used for the initial test of each program.

Another factor to be considered in the implementation phase is the acquisition of the hardware and software. Once the software is developed for the system and testing is carried out, it is then the process of making the newly designed system fully operational and consistent in performance.

Implementation is the most crucial stage in achieving a successful system and giving the user's confidence that the new system is workable and effective. Implementation of a modified application to replace an existing one. This type of conversation is relatively easy to handle, provide there are no major changes in the system.

1. **System Implementation:**
   There are three major types of implementations are there but the following are proposed for the project.
2. **Parallel Conversion type of Implementation:**
   In this type of implementation both the current system and the proposed system run in parallel. This happens till the user gets the complete confidence on the proposed system and hence cuts of the current system.
3. **Phase - in method of implementation:**
   In this type of implementation, the proposed system is introduced phase-by-phase. This reduces the risk of uncertainty of proposed system.

Each program is tested individually at the time of development using the data and has verified that this program linked together in the way specified in the programs specification, the computer system and its environment is tested to the satisfaction of the user. The system that has been developed is accepted and proved to be satisfactory for the user. And so, the system is going to be implemented very soon. A simple operating procedure is included so that the user can understand the different functions clearly and quickly.

Initially as a first step the executable form of the application is to be created and loaded in the common server machine which is accessible to the entire user and the server is to be connected to a network. The final stage is to document the entire system which provides components and the operating procedures of the system.

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

Implementation is the process of converting a new system design into operation. It is the phase that focuses on user training, site preparation and file conversion for installing a candidate system. The important factor that should be considered here is that the conversion should not disrupt the functioning of the organization.

4. Implementation Methodology of the Project:

The project is implemented in modular approach. Each module is coded as per the requirements and tested and this process is iterated till the all the modules have been thoroughly implemented.

5. System Security:

Although our method is mainly thought Here in proposed methodology RNN & LSTM classifier whether to be known as effected or not all the features are extracted using feature acquisition and feature extraction.

5.2 SYSTEM IMPLEMENTATION

This application is executed utilizing python programming language. Article situated in writing a stepwise program to implement the application. Each program needs to store inside the memory and the capacities can be in any formats. Highlights of the object-oriented worldview:

- Emphasis is based on the given information may opposes to methods.
- Computer programs are divided into different items.
- Product is partitioned according to the data structures.
- Methods will work according to the design of the system.
- Objects of the system will relate to one another techniques.
- Different strategies are used to improve the product or application.
- According to the plan, application or product will be generated or designed.
- Datasets are used within the given capacities.

5.3 ALGORITHMS USED

5.3.1 YOLO:

YOLO is a state-of-the-art object detection model that can detect multiple objects in an image in a single pass. Unlike traditional object detection models that apply a sliding window to an image multiple times to predict the presence of objects, YOLO views the entire image during the prediction process. This global understanding of the image allows YOLO to detect objects quickly and accurately, making it ideal for real-time applications such as animal detection in this project.

YOLO (You Only Look Once) divides the image into a grid and predicts bounding boxes and class probabilities for each cell. This enables it to detect multiple objects of different categories in a single frame efficiently. Its architecture is based on a single convolutional neural network (CNN) that performs both classification and localization tasks simultaneously. YOLO reduces the number of false positives compared to traditional methods, improving reliability in critical applications. Due to its high speed and accuracy, it is widely used in autonomous driving, surveillance, and smart farming systems.

YOLO's ability to generalize well from training data allows it to detect unseen objects with reasonable precision.

In animal detection, YOLO helps differentiate between species based on shape, size, and other visual features. The real-time processing capability makes it suitable for live video feed

monitoring and instant alerts. YOLO also supports transfer learning, allowing developers to fine-tune the model with specific animal datasets.

Overall, YOLO brings a balance between performance and speed, making it a powerful tool for intelligent detection systems.

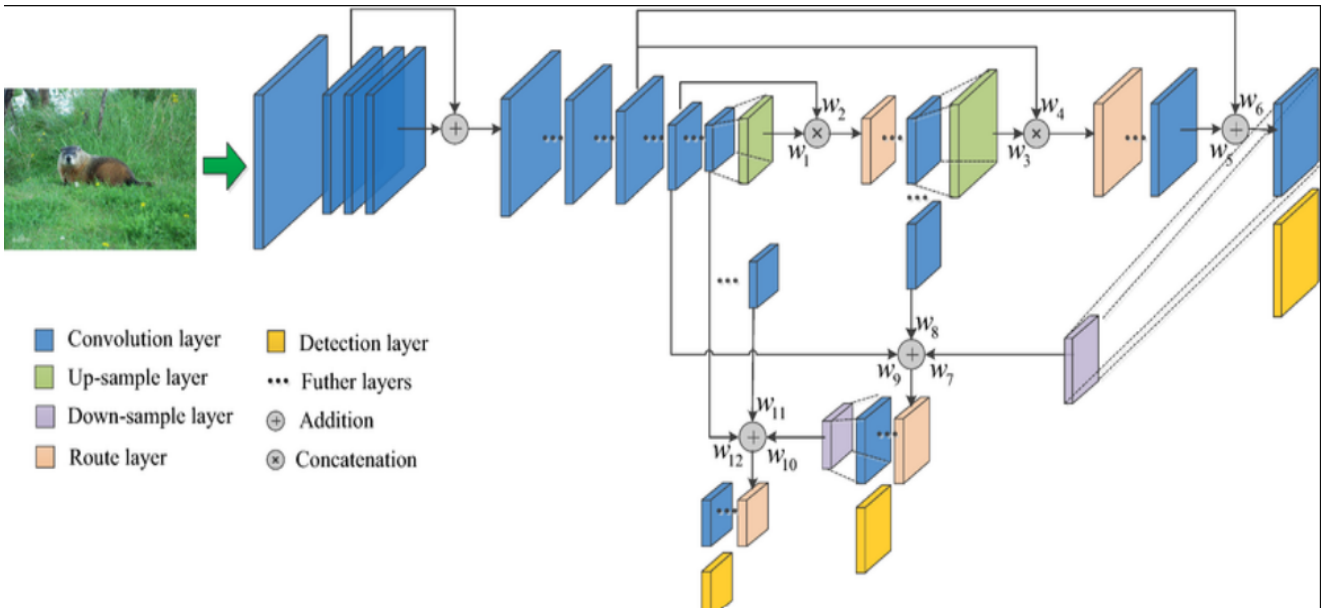


Figure 5.3.1: YOLO Architecture

YOLO divides an image into a grid of cells, and each cell is responsible for predicting a certain number of bounding boxes along with the associated class probabilities. Here's how it works step-by-step:

1. **Grid Division:**

   The input image is divided into a grid (e.g., 13x13 for small images or 19x19 for larger images). Each grid cell is responsible for detecting an object within that portion of the image.

2. **Bounding Boxes:**

   Each grid cell predicts a fixed number of bounding boxes (usually 2-3), with each box having:

   - Coordinates (x, y) for the box's center relative to the grid cell.
   - Width (w) and height (h) of the bounding box.
   - Confidence score indicating how confident the model is that an object exists within that box.

3. **Class Prediction:**
   In addition to the bounding box, each grid cell predicts the class of the object (e.g., dog, cat, lion). YOLO assigns a probability to each class, indicating how likely the object is to belong to that category.
4. **Class Probability:**
   YOLO assigns class probabilities to each detected object. For instance, if an animal detected in an image is classified as a dog, YOLO outputs a high probability for the "dog" class and low probabilities for others like "cat" or "bird."
5. **Non-Maximum Suppression (NMS):**
   YOLO generates multiple bounding boxes for the same object. Non-Maximum Suppression is applied to remove duplicate bounding boxes by selecting the one with the highest confidence score. This ensures that each object is detected once with the most accurate bounding box.

5.3.2 YOLO in Animal Classification

For the animal classification task in this project, the YOLO model performs the following steps:

1. **Input Image:**
   The user uploads an image containing an animal (e.g., a dog, cat, elephant). The system pre-processes the image (resizing, normalization) before feeding it into the YOLO model.
2. **Object Detection:**
   YOLO detects any object within the image and assigns it to a specific class (e.g., "dog", "cat", "deer"). The grid cells predict bounding boxes that highlight the position of each detected animal.
3. **Class Prediction:**
   YOLO outputs the class probabilities for each object in the image. The animal detected will be classified according to the highest probability. For instance, if the image contains a cat, YOLO will output a "cat" class with a high confidence score.
4. **Detection and Sound Activation:**
   Once the animal is detected and classified, the system triggers an appropriate anti-animal sound to deter the animal, which can be heard through connected speakers.

5. **Email Alert:**
   Simultaneously, an email alert is sent to the registered user or admin, informing them about the detected animal along with the image and detection details.

Advantages of Using YOLO for This Project

- Real-time Detection: YOLO is incredibly fast and can process an image in milliseconds, which is critical for applications requiring real-time object detection.
- High Accuracy: YOLO's unified architecture and grid-based predictions make it accurate in detecting multiple objects within an image, including animals in various poses and conditions.
- Efficiency: YOLO can handle large-scale image datasets efficiently, which is ideal for dealing with real-world images from various environmental settings (e.g., farms, forests, and rural areas).
- Single-Pass Detection: Unlike other models that require multiple passes for object detection, YOLO performs detection in a single pass, significantly reducing processing time and enabling quick animal classification.

5.3.3 YOLO Versions Used

The project can leverage any of the modern YOLO versions based on the requirement for detection accuracy and computational resources. Some versions are:

- YOLOv3: Known for balance between accuracy and speed.
- YOLOv4: Offers improved performance, better accuracy, and is suitable for industrial use cases.
- YOLOv5: The latest version, with enhancements in accuracy and speed, and easier to use.
- Depending on the resources and deployment requirements, one of these versions can be selected.

5.4 TOOLS AND TECHNOLOGIES USED

5.4.1 OVERVIEW OF PYTHON

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released in 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released in 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3.

The Python 2 language, i.e. Python 2.7.x, was officially discontinued on January 1, 2020 (first planned for 2015) after which security patches and other improvements will not be released for it. With Python 2's end-of-life, only Python 3.5.x and later are supported.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open-source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL),capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until July 12, 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator for Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. He now shares his leadership as a member of a five-person steering council. In January, 2019, active Python core developers elected Brett Cannon, Nick Coghlan, Barry Warsaw, Carol Willing and Van Rossum to a five-member "Steering Council" to lead the project.

Python 2.0 was released on 16 October 2000 with many major new features, including a cycle-detecting garbage collector and support for Unicode.

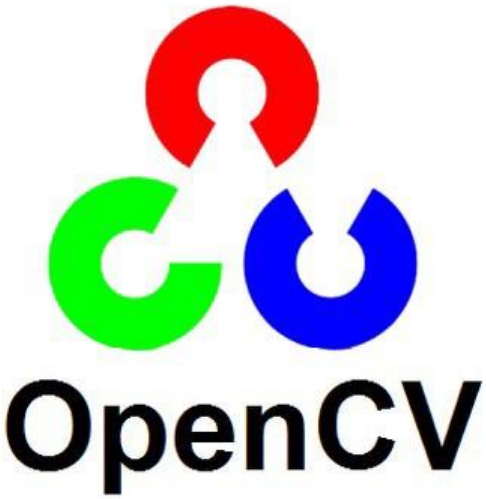
Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2to3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3.

5.4.2 OPENCV

OpenCV (Open-source computer vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license.

OpenCV supports some models from deep learning frameworks like TensorFlow, Torch, PyTorch (after converting to an ONNX model) and Caffe according to a defined list of supported layers. It promotes OpenVisionCapsules, which is a portable format, compatible with all other formats.



Officially launched in 1999 the OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time ray tracing and 3D display walls. The main contributors to the project included a number of optimization experts in Intel Russia, as well as Intel's Performance Library Team. In the early days of OpenCV, the goals of the project were described as:

Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.

Advance vision-based commercial applications by making portable, performance-optimized code available for free – with a license that did not require code to be open or free itself.

The first alpha version of OpenCV was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five betas were released between 2001 and 2005. The first 1.0 version was released in 2006. A version 1.1 "pre-release" was released in October 2008.

The second major release of the OpenCV was in October 2009. OpenCV 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi-core systems). Official releases now occur every six months and development are now done by an independent Russian team supported by commercial corporations.

In August 2012, support for OpenCV was taken over by a non-profit foundation OpenCV.org, which maintains a developer and user site. On May 2016, Intel signed an agreement to acquire Itseez, a leading developer of OpenCV.

5.4.3 TENSORFLOW

TensorFlow is a free and open-source software library 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. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 on November 9, 2015.

TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017.[10] 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. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.

In December 2017, developers from Google, Cisco, RedHat, CoreOS, and CaiCloud introduced Kubeflow at a conference. Kubeflow allows operation and deployment of TensorFlow on Kubernetes. In March 2018, Google announced TensorFlow.js version 1.0 for machine learning in JavaScript. In Jan 2019, Google announced TensorFlow 2.0. It became officially available in Sep 2019. In May 2019, Google announced TensorFlow Graphics for deep learning in computer graphics.

TensorFlow is an end-to-end open-source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models. See the TensorFlow documentation for complete details on the broader TensorFlow system.

TensorFlow APIs are arranged hierarchically, with the high-level APIs built on the low-level APIs. Machine learning researchers use the low-level APIs to create and explore new machine learning algorithms. In this class, you will use a high-level API named tf.keras to define and train machine learning models and to make predictions. tf.keras is the TensorFlow variant of the open-source Keras API.

The following figure shows the hierarchy of TensorFlow toolkits:

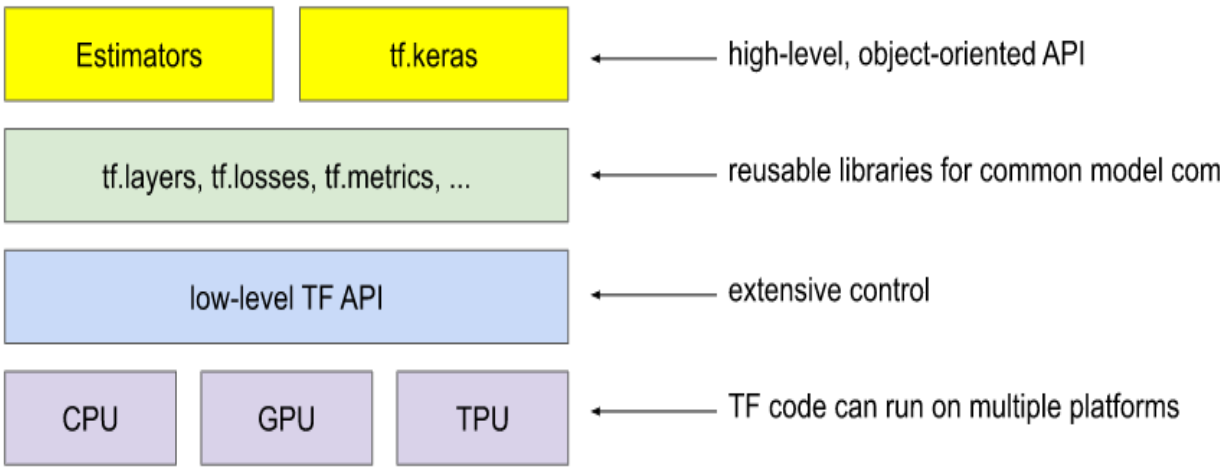


Figure 5.4.3: Hierarchy of TensorFlow toolkits

CHAPTER -06

TESTING PHASE

6.1 INTRODUCTION

The **Testing Phase** is a critical component in the software development life cycle, aimed at evaluating and validating the functionality, performance, and reliability of the system before its deployment. In the context of the **Bird Classification Using Deep Learning** project, testing plays an essential role in ensuring the accuracy of the bird detection and classification models, as well as verifying the usability and stability of the web-based application. The primary objective of this phase is to identify and rectify defects or inconsistencies in the system to ensure it performs according to specified requirements. This includes verifying image uploads, model predictions, user authentication, system responses, and overall user interaction with the application. Additionally, the deep learning models (YOLO for detection and CNN for classification) are rigorously tested for precision, recall, and overall classification accuracy using standard datasets and real-world images.

The testing process involves multiple levels of testing such as **unit testing**, **integration testing**, **system testing**, and **user acceptance testing (UAT)**. These tests are performed systematically to validate the correctness of individual components, the interaction between modules, and the behavior of the entire application in a real-world environment. Furthermore, special attention is given to performance testing to ensure the system can handle concurrent image uploads, real-time classification, and multiple user sessions without compromising on speed or accuracy. Security testing is also conducted to ensure secure authentication, protection against common web vulnerabilities, and the confidentiality of user-uploaded data.

By thoroughly testing all aspects of the application, from backend processing to frontend interaction, the system is validated to be robust, reliable, and user-friendly. This phase ultimately builds confidence in the system's functionality and prepares it for deployment in real-world scenarios where bird classification can contribute to conservation, research, and educational efforts.

6.2 TEST CASES

Test Case ID	Test Case Description	Input	Expected Output	Actual Output	Status
TC1	Test user registration with valid data	Name, Username, Password, Email, Phone	User registered successfully	Correct output	Pass
TC2	Test user login with valid credentials	Username, Password	Login successful	Correct output	Pass
TC3	Test user login with invalid credentials	Invalid Username, Password	Login failed	Correct output	Pass
TC4	Test image upload with valid image	Animal image file (JPEG, PNG)	Image uploaded successfully	Correct output	Pass
TC5	Test image upload with unsupported file format	Unsupported file format (GIF)	Error message: Unsupported format	Correct output	Pass
TC6	Test pre-processing of image (resizing, normalization)	Animal image file	Image resized and normalized	Correct output	Pass
TC7	Test animal classification with a tiger image	Tiger image	Classified as "Tiger"	Correct output	Pass
TC8	Test animal classification with an Elephant image	Elephant image	Classified as "Elephant"	Correct output	Pass
TC9	Test animal classification with an unrecognized animal	Image of an unrecognized animal	"Unknown" class	Correct output	Pass
TC10	Test YOLO model detection for multiple animals	Image with a tiger and an elephant	"Tiger", "Elephant" detected	Correct output	Pass
TC11	Test anti-animal sound activation upon detection	Tiger image	Sound played for tiger detection	Correct output	Pass

TC12	Test email alert upon detection of animal	Image with detected animal (e.g., tiger)	Email alert sent with detection details	Correct output	Pass
TC13	Test detection when image has no animals	Image with no animals	No detection, no alert	Correct output	Pass
TC14	Test detection with low-quality image	Low-resolution image of an animal	Animal detected with reduced accuracy	Correct output	Pass
TC15	Test detection when image has overlapping animals	Image with overlapping animals (e.g., Tiger and Elephant)	Separate detection of "Tiger " and "Elephant "	Correct output	Pass
TC16	Test detection of animal with partial visibility	Image of an animal partially visible (e.g., animal behind object)	Correct detection with partial bounding box	Correct output	Pass
TC17	Test detection with various environmental conditions	Image taken in low light conditions	Accurate detection despite lighting issues	Correct output	Pass
TC18	Test detection of fast-moving animals	Image of fast-moving animal	Accurate detection of fast-moving animal	Correct output	Pass
TC19	Test system response time for image processing	Image of any animal	Detection and alert within 5 seconds	Correct output	Pass

Table 6.2:Test Case Table

CHAPTER-07

RESULTD AND DISCUSSIONS

7.1 INPUT IMAGE

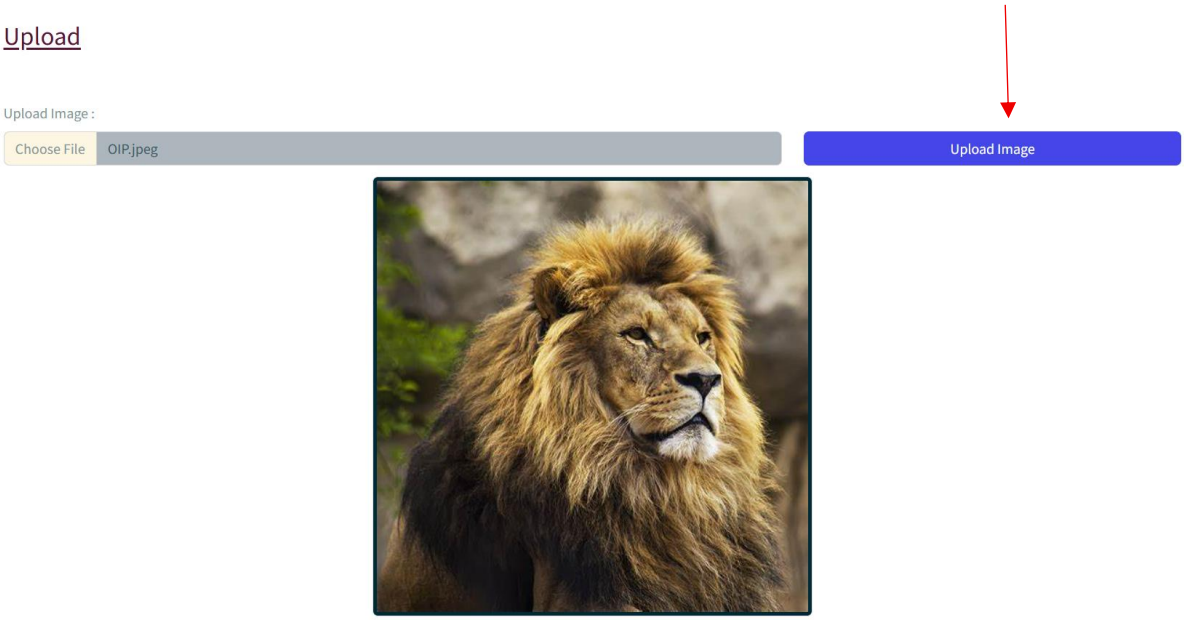


Figure 7.1:Input image Example

7.2 GRAY SCALE

Figure 7.2: Gray Scale Example

7.3 BINARIZATION

Figure 7.3: Binarization Example

7.4 THRESHOLDING

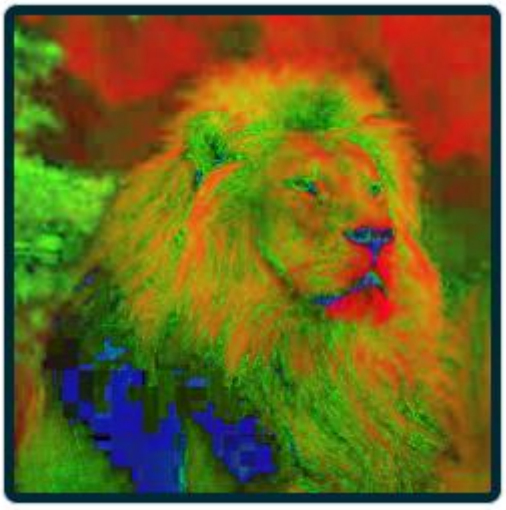


Figure 7.4: Thresholding Example

7.5 RESULT

Animal Detected : lion

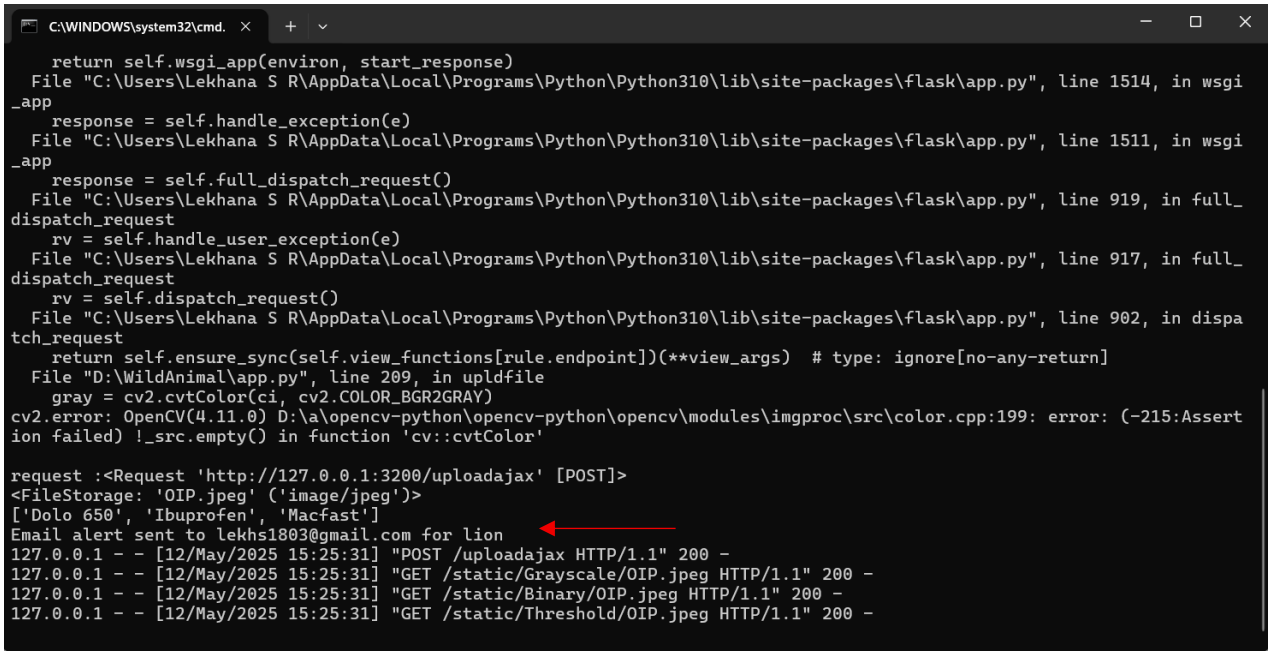


Figure 7.5.1: Result Example in cmd

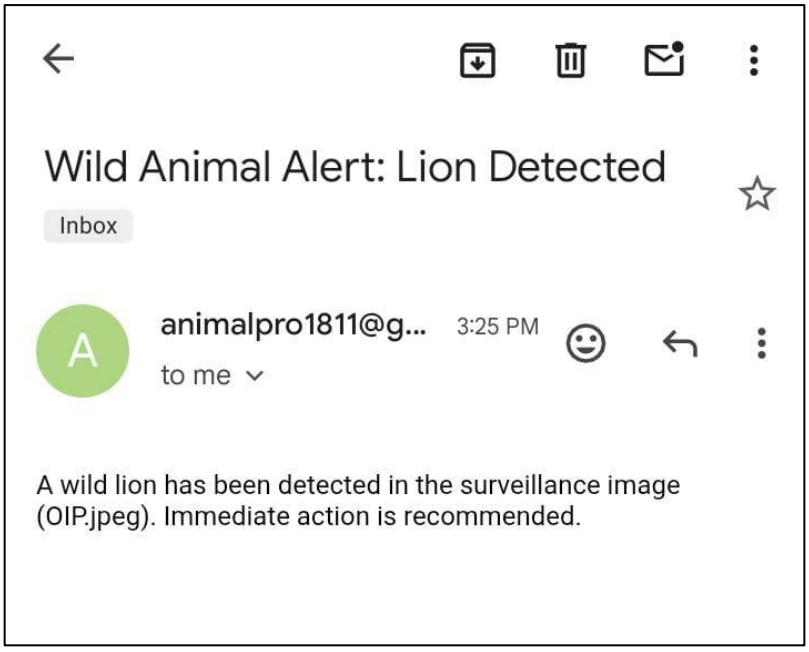


Figure 7.5.2: Email alert Example

CHAPTER-08

CONCLUSION AND FUTURE ENHANCEMENT

8.1 MAINTENANCE

Maintenance of the Animal Classification Using Deep Learning project involves the continuous monitoring, updating, and improving of the system to ensure its efficiency, accuracy, and relevance over time. Regular maintenance tasks include:

- **Model Updates:** The deep learning models used for animal detection (such as YOLO) may need periodic retraining with new datasets to enhance accuracy, especially as the system encounters new animal types or environmental conditions. Updating the model will ensure it remains effective at classifying animals in real-world scenarios.
- **Database Management:** The database, which stores registered user data, animal classifications, and email alerts, requires routine maintenance. This includes ensuring data integrity, performing backups, and optimizing database queries to handle larger datasets as the system scales.
- **Bug Fixes and Performance Improvements:** Over time, as new features are added or user demands increase, the system may encounter bugs or performance issues. These will need to be addressed promptly to maintain a seamless user experience.
- **System Compatibility and Security:** Keeping the software and hardware components up to date with the latest security patches and ensuring compatibility with newer versions of Python, YOLO, and other libraries is essential for keeping the system secure and functional.

On going support will be necessary to manage the smooth operation and longevity of the system, making it adaptable to future developments in both deep learning technologies and real-world environmental conditions.

8.2 CONCLUSION

The Animal Classification Using Deep Learning project has successfully demonstrated the capability to detect and classify animals in real-time using the YOLO (You Only Look Once) deep learning model. This system not only detects animals in uploaded images but also triggers anti-animal sounds to deter animals and sends timely email alerts to registered users.

By utilizing advanced deep learning techniques and leveraging the power of YOLO, the project offers a robust solution to animal classification and management.

The system operates efficiently with minimal delays, allowing for real-time processing of images while maintaining a high level of accuracy in animal detection. It integrates seamlessly with the user interface, providing an easy-to-use experience for animal detection and alerting.

Overall, this project contributes to solving real-world problems such as managing and controlling animal movements in sensitive areas like farms or urban environments. The integration of deep learning and automation makes it a significant step forward in wildlife management and protection.

8.3 FUTURE ENHANCEMENT

While the Animal Classification Using Deep Learning project is a highly functional and robust system, there are several potential enhancements to improve its capabilities:

- **Multi-Species Detection:**
  The system can be extended to detect a broader range of animals, especially rare or endangered species. This can be achieved by expanding the dataset and retraining the YOLO model to include more animal classes.

- **Integration with Drones:**
  The system could be enhanced by integrating it with drone technology for aerial surveillance. Drones equipped with cameras could capture images of large areas, and the animal classification system could process those images in real-time for broader environmental monitoring.

- **Environmental Adaptability:**
  Future iterations of the system could be designed to handle various environmental challenges, such as fluctuating light conditions or weather conditions, by utilizing data from multiple sensors (e.g., infrared sensors for night-time detection).

- **Mobile App Integration:**
  A mobile version of the application could be developed to allow users to upload images directly from their smartphones, receive real-time notifications, and interact with the system more efficiently.

- **Offline Functionality with Local Processing:**

  To improve reliability in remote or low-connectivity areas, the system can be upgraded to process data locally using edge devices like Raspberry Pi or NVIDIA Jetson. The AI model (e.g., VGG16) will run directly on the device to detect animals from live video feeds without needing internet access. Deterrent mechanisms and alert responses will also be triggered locally, ensuring uninterrupted protection. This reduces dependency on cloud services, lowers latency, and enhances real-time performance. Local data can be stored and synced to the cloud once the network becomes available.

- **SMS Alert Using API :**

  The system can be improved by integrating an SMS alert feature using APIs like Twilio or Fast2SMS. When a wild animal is detected, an automated SMS can be sent instantly to farmers or concerned authorities. This ensures timely notification even in areas where email alerts may be delayed or inaccessible. SMS alerts enhance responsiveness, especially during emergencies or nighttime intrusions. This feature adds a critical communication layer for better farm protection.

- **Hybrid Mode :**

  Hybrid mode enables the system to operate seamlessly in both **online** and **offline** environments. When internet connectivity is available, the system can sync data to the cloud, send email/SMS alerts, and receive remote updates. In offline mode, local processing handles detection and deterrence, storing data locally for later sync. This ensures continuous, uninterrupted functionality regardless of network conditions.

  Hybrid mode improves flexibility, reliability, and scalability of the system in diverse real-world settings.

- **More Alert Options:** Besides sound alerts, the system could be expanded to offer more deterrent methods, such as visual warnings (flashing lights) or even automated response mechanisms like motion-activated sprinklers for outdoor environments.

Through these enhancements, the system could be made even more versatile, scalable, and applicable to a wider range of real-world scenarios.

REFERENCES

[1] Elson, J., Douceur, J., Howell, J., *et al.*: 'Asirra: a CAPTCHA that exploits interest-aligned manual image categorization'. Proc. ACM Conf. Computerand Communications Security (CCS), Alexandria VA, USA, 2007, pp. 366–374

[2] Marcialis, G., Roli, F.: 'Score-level fusion of fingerprint and face matchers forpersonal verification under stress conditions. 14th IEEE Int. Conf. Image Analysis and Processing ICIAP, DC, USA, 2007, pp. 259–264

[3] Elmir, Y., Elberrichi, Z., Adjoudj, R.: 'Score-level fusion based multimodalbiometric identification (fingerprint & voice)'. 6th Int. Conf. Sciences of Electronics, Technologies of Information and Telecommunications, Sousse,2010, pp. 146–150

[4] Penga, Z., Lia, Y., Caib, Z., *et al.*: 'Deep boosting: joint feature selection andanalysis dictionary learning in hierarchy', *Neurocomputing*, 2016, 178, (20), pp. 36–45

[5] Afkham, H., Tavakoli, A., Eklundh, J., *et al.*: 'Joint visual vocabulary foranimal classification'.Int.Conf. Pattern Recognition, 2008. ICPR 2008,Tampa, FL, USA, 2008, pp.1-4

[6] Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*.

[7] Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

[8] Chen, J., Han, D., Liu, D., Song, H., & Liu, Q. (2019). Automatic wildlife monitoring using very-deep convolutional neural networks. *IEEE Access, 7*, 169989–170006.

[9] Norouzzadeh, M. S., Nguyen, A., Kosmala, M., Swanson, A., Palmer, M. S., Packer, C., & Clune, J. (2018). Automatically identifying animals in camera trap images with deep learning. *Proceedings of the National Academy of Sciences, 115*(25), E5716–E5725.

[10] Beery, S., Morris, D., & Yang, S. (2019). Efficient pipeline for camera trap image processing using machine learning. *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019, 38–46.

[11] Corrado, A. (2019). Animals-10: Image dataset for transfer learning. *Kaggle Datasets*. Retrieved from https://www.kaggle.com/datasets/alessiocorrado99/animals10

[12] Singh, D., Jain, N., & Vatsa, M. (2020). Real-time object detection using deep learning for wildlife monitoring. *Procedia Computer Science*, *167*, 1841–1850.

[13] Zhang, Y., Xie, F., & Yang, L. (2020). AI-based animal deterrent system for agriculture. *Computers and Electronics in Agriculture, 178*, 105751.

[14] Zhang, Z., Cui, P., & Wang, Z. (2018). Deep learning on small datasets for animal classification. *Journal of Visual Communication and Image Representation, 55*, 640–646.

[15] Fernando, P., & Kanchana, A. (2019). An automated system for detecting elephants to prevent human-animal conflict. *International Journal of Computer Applications, 975*, 8887.