International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

Enhancing Precision Agriculture Pest Control: A Yolov10-Based Deep Learning Approach For Insect Detection

Dr.M.Gokilavani[1],M.Gopi[2],P.Saikiran Reddy[3],R.Prasanna Srivatsa[4],V.Rakesh[5]

[1]Associate Professor; Guru Nanak Institutions Technical Campus Department Of Information Technology

[2,3,4,5]B.Tech Students; Guru Nanak Institutions Technical Campus Department Of Information Technology

Mail Id; Rakeshvilasagarapu@Gmail.Com[5]

Abstract

Precision Agriculture (PA) integrates modern technological solutions to enhance resource efficiency while maintaining crop productivity and quality. Despite these advancements, pest infestations continue to pose a significant threat to agricultural sustainability. Recent developments in deep learning-based object detection models, particularly the YOLO (You Only Look Once) series, have enabled real-time insect detection. However, many existing approaches, including YOLOv8-based systems, are often constrained to specific insect classes or crop environments, limiting their general applicability.

To overcome these challenges, this study proposes a more generalized and efficient insect detection framework utilizing the latest YOLOv10 architecture. The proposed system is designed to identify multiple insect categories across a wide range of crops, thereby supporting scalable and real-time pest monitoring in diverse agricultural settings. Experimental evaluation was conducted using a standard benchmark insect dataset to assess the model's performance.

The results demonstrate that the YOLOv10-based approach outperforms earlier models, including YOLOv8, in terms of mean Average Precision (mAP) and inference speed. These improvements can be attributed to the architectural enhancements in YOLOv10, which enable better feature representation and faster processing. Overall, the proposed method offers a robust and adaptable solution for pest detection, contributing to more effective pest management strategies in precision agriculture.

INTRODUCTION

Precision Agriculture (PA) represents a transformative approach to modern farming, enabling data-driven decision-making through continuous monitoring and analysis of field conditions. By leveraging advanced technologies such as sensors, imaging systems, and machine learning, PA improves resource utilization while enhancing crop productivity and quality. Despite these advancements, pest infestation remains a major constraint, often leading to significant yield losses and reduced agricultural efficiency.

Traditional pest monitoring methods rely heavily on manual inspection, which is time-consuming, labor-intensive, and prone to human error. Moreover, these approaches lack scalability and fail to provide real-time insights necessary for timely intervention. In recent years, deep learning-based object detection techniques have emerged as effective alternatives, offering automated and rapid pest identification using images and video streams.

Among these techniques, the YOLO (You Only Look Once) family of models has gained considerable attention due to its ability to perform real-time object detection with high accuracy. In particular, YOLOv8 has been widely adopted in agricultural applications because of its balance between speed and precision. However, most YOLOv8-based pest detection systems are designed for specific insect species and crop types, limiting their adaptability to new environments. Additionally, these models often face challenges in detecting small, occluded, or overlapping insects in complex field conditions.

To overcome these limitations, this work proposes a generalized pest detection system based on the latest YOLOv10 architecture. YOLOv10 introduces several enhancements, including improved feature extraction, efficient model scaling, and an anchor-free detection mechanism, which collectively enhance detection accuracy and inference speed. Unlike conventional approaches, the proposed system is trained to recognize insects as a single generalized class, enabling it to detect a wide range of insect types across diverse crops without requiring retraining.

By addressing the shortcomings of existing methods and adapting to real-world agricultural challenges, this study aims to develop a robust, scalable, and real-time pest detection framework that supports efficient and sustainable pest management in precision agriculture.

Scope of the Project

The scope of this project includes the design, implementation, and evaluation of a generalized pest detection system using the YOLOv10 object detection framework. Unlike conventional systems that are restricted to specific insect species or crop types, the proposed approach focuses on developing a scalable model capable of detecting a broad range of insects across multiple agricultural environments. The project encompasses the complete workflow, including dataset collection and preprocessing, model training and validation, performance testing,

and comparative analysis with existing YOLOv8-based systems. Additionally, the system is designed to process both images and video streams in real time, making it suitable for deployment in practical scenarios such as field monitoring, drone-based surveillance, and mobile applications.

By emphasizing generalization, efficiency, and real-time capability, the project contributes to improved pest monitoring and supports informed decision-making in precision agriculture.

Objectives

The primary objective of this project is to develop a generalized and real-time pest detection system using the YOLOv10 framework. The system is designed to overcome the limitations of existing models that are restricted to specific insect classes and agricultural settings.

The key objectives include:

- To design a YOLOv10-based model capable of detecting insects across diverse crop environments.
- To train the model using a comprehensive dataset labeled under a unified insect category to improve generalization.
- To enhance detection performance for small, occluded, and densely clustered insects.
- To optimize the model for real-time inference on resource-constrained edge devices such as drones and mobile platforms.
- To evaluate system performance using metrics such as mean Average Precision (mAP), inference time, and robustness, and compare results with YOLOv8-based approaches.

Existing System

Current pest detection systems in precision agriculture predominantly utilize deep learning-based object detection algorithms, with YOLOv8 being one of the most widely adopted models. YOLOv8 is a single-stage detector that processes images in a single pass, simultaneously predicting object locations and class labels. This design enables fast and efficient detection, making it suitable for real-time agricultural applications.

In practice, YOLOv8-based systems are typically trained on datasets containing specific insect species associated with particular crops. While this targeted training approach yields high accuracy under controlled conditions, it limits the system's ability to generalize to new pests or different agricultural environments.

Furthermore, real-world agricultural conditions introduce additional challenges, such as variations in lighting, complex backgrounds, and the presence of small or partially occluded insects. These factors can significantly reduce detection accuracy. Although YOLOv8 offers improved performance compared to earlier models, its computational requirements may still hinder deployment on low-power edge devices.

Overall, while the existing system represents a significant improvement over traditional manual methods, it remains constrained in scalability, adaptability, and robustness.

Disadvantages of Existing System

- Limited to specific insect species and crop types
- Reduced accuracy for small, overlapping, or occluded insects
- Requires retraining for new pests or agricultural environments
- High computational requirements for real-time edge deployment
- Performance degradation under varying lighting and complex backgrounds

Literature Survey

Recent research highlights the growing importance of integrating artificial intelligence into pest management systems. Studies emphasize the need for sustainable pest control strategies and early detection mechanisms to reduce reliance on chemical pesticides and ensure food security.

Comprehensive reviews of YOLO-based architectures demonstrate continuous improvements in detection accuracy, speed, and efficiency across different versions. These advancements have enabled the application of deep learning models in complex real-world environments, including agriculture.

Several works have focused on improving lightweight YOLO models for pest detection, particularly in scenarios involving small and densely packed insects. These studies highlight the importance of balancing model accuracy with computational efficiency for deployment on edge devices.

Systematic reviews of deep learning-based insect detection further identify key challenges such as limited dataset availability, difficulty in detecting small objects, and the need for generalized detection models. These findings strongly support the development of scalable solutions capable of handling diverse pest categories.

Additionally, advancements in loss functions and optimization techniques have contributed to improved detection performance, particularly in dense object detection scenarios. Such improvements are crucial for addressing the challenges associated with real-world agricultural environments.

Proposed System

The proposed system introduces a generalized pest detection framework based on the YOLOv10 architecture. Unlike existing approaches that focus on specific insect classes, the proposed model is trained to detect insects as a single unified category, enabling broader applicability across different crops and environments.

YOLOv10 incorporates several architectural enhancements, including improved feature

extraction, multi-scale detection capabilities, and an anchor-free design. These features enable more accurate detection of small, occluded, and overlapping insects commonly found in agricultural fields.

The system pipeline includes data preprocessing, model training, validation, and real-time inference on images and video streams. Detection results are visualized using bounding boxes, allowing users to easily identify pest presence in the field.

In addition to improved accuracy, the proposed system is optimized for computational efficiency, making it suitable for deployment on edge devices such as drones, mobile phones, and field cameras. This enables real-time monitoring and supports timely decision-making for pest control.

General Overview

This project focuses on enhancing precision agriculture through the development of a real-time and generalized insect detection system based on the YOLOv10 object detection framework. Conventional pest detection methods often rely on models trained to identify specific insect species associated with particular crops. While such approaches can achieve high accuracy under controlled conditions, they lack flexibility and scalability when applied to diverse agricultural environments.

To address these limitations, the proposed system adopts a generalized detection strategy in which all insect types are treated as a single class. This enables the model to detect a wide variety of pests across multiple crops and environmental conditions without requiring retraining for each new scenario. The system is designed as an end-to-end pipeline, incorporating data acquisition, preprocessing, model training, validation, testing, and deployment.

The use of YOLOv10 introduces several architectural advancements, including improved feature extraction, efficient multi-scale representation, and an anchor-free detection mechanism. These enhancements allow the system to achieve higher detection accuracy, particularly for small, overlapping, or partially occluded insects that are commonly encountered in agricultural fields. Additionally, the optimized computational design ensures efficient execution on resource-constrained devices such as drones, surveillance cameras, and mobile platforms.

Beyond technical performance, the system aims to provide practical benefits by enabling farmers to monitor pest activity in real time. This facilitates timely interventions, reduces crop damage, minimizes excessive pesticide usage, and promotes sustainable agricultural practices. Overall, the proposed solution offers a scalable, robust, and efficient approach to pest detection aligned with the goals of modern precision agriculture.

Techniques and Algorithms Used

Existing Technique: YOLOv8

The existing pest detection approach is primarily based on the YOLOv8 object detection framework, particularly the lightweight YOLOv8s variant. As a single-stage detector, YOLOv8 processes the entire image in one pass, predicting object locations and class probabilities simultaneously. This enables fast and efficient detection, making it suitable for real-time applications.

YOLOv8 incorporates improvements over earlier versions, such as enhanced feature extraction, better feature fusion, and a decoupled detection head. These advancements contribute to improved accuracy and faster training convergence. In agricultural applications, YOLOv8 models are typically trained on datasets containing specific insect species associated with particular crops.

Despite its advantages, YOLOv8 faces several limitations. Its dependence on crop-specific datasets restricts its ability to generalize to new pest species or different agricultural environments. Detection accuracy can also decline in complex field conditions, where insects may be small, partially occluded, or clustered together. Furthermore, deploying YOLOv8 on low-power devices can still be computationally demanding. Adapting the model to new pests often requires additional data collection and retraining, increasing operational complexity.

Proposed Technique: YOLOv10

The proposed system employs the YOLOv10 object detection model to overcome the limitations of existing approaches. YOLOv10 introduces significant architectural enhancements that improve detection accuracy, efficiency, and robustness, making it well-suited for real-world agricultural scenarios.

One of the key features of YOLOv10 is its anchor-free detection mechanism, which eliminates the need for predefined bounding box anchors. This allows the model to better detect objects of varying sizes and shapes, particularly small and densely packed insects. Additionally, the improved backbone architecture enhances feature extraction, enabling more effective representation of complex visual patterns.

The system adopts a generalized detection strategy by labeling all insect types under a single class. This eliminates the need for species-specific training and enables the model to detect a wide range of pests across different crops and environments. Transfer learning is employed to initialize the model with pre-trained weights, followed by fine-tuning on an insect dataset to improve domain-specific performance.

To further enhance robustness, data augmentation techniques are applied to simulate real-world variations. Hyperparameter tuning is also performed to optimize training efficiency and accuracy. Once trained, the model performs real-time detection on

image and video inputs, making it suitable for deployment on edge devices such as drones and mobile platforms.
The system is evaluated using standard performance metrics, including accuracy, precision, recall, F1-score, and mean Average Precision (mAP). Experimental results demonstrate that YOLOv10 provides improved detection performance, faster inference, and better generalization compared to YOLOv8.

REQUIREMENTS ENGINEERING

Requirements engineering is a critical phase in system development that involves identifying, analyzing, and defining the functional and non-functional needs of a system. In the context of the proposed pest detection framework, this phase ensures that the system is capable of delivering accurate, real-time insect detection while maintaining efficiency and scalability.
The effectiveness of the proposed system is largely influenced by the robustness of feature extraction and the predictive capability of the detection model. By leveraging advanced deep learning techniques, particularly the YOLOv10 architecture, the system is designed to achieve high detection accuracy with minimal error rates. Compared to existing approaches, the proposed solution demonstrates competitive performance in terms of accuracy, speed, and generalization across diverse agricultural datasets.

Hardware Requirements

Hardware requirements define the physical components necessary for the development, training, and deployment of the system. These specifications provide a baseline for implementation and ensure smooth system performance.

Minimum Hardware Configuration:

- **Processor**: Dual-Core Processor (or higher)
- **RAM**: 4 GB (8 GB recommended for training deep learning models)
- **Storage**: 250 GB Hard Disk (SSD preferred for faster processing)

Recommended Configuration (for better performance):

- **Processor**: Intel i5/i7 or equivalent
- **RAM**: 8 GB or higher
- **GPU**: NVIDIA GPU (for accelerated model training and inference)

Software Requirements

Software requirements specify the tools and platforms used to design, develop, and deploy the system. These components play a crucial role in ensuring flexibility, scalability, and ease of implementation.

- **Operating System**: Windows 10 / Windows 11 / Linux
- **Programming Language**: Python
- **Development Environment**: Anaconda Distribution
- **IDE / Editor**: Visual Studio Code / Jupyter Notebook
- **Libraries & Frameworks**:
  - Deep Learning: PyTorch / TensorFlow
  - Computer Vision: OpenCV
  - Data Handling: NumPy, Pandas
  - Visualization: Matplotlib

3.4 Functional Requirements

Functional requirements describe the core operations and behaviors of the system. These define what the system is expected to perform.

- The system shall accept input in the form of images or video streams from cameras, drones, or datasets.
- The system shall preprocess input data to improve image quality and normalize formats.
- The system shall detect insects in real time using the YOLOv10 model.
- The system shall localize detected insects by generating bounding boxes around them.
- The system shall classify detected objects under a generalized insect category.
- The system shall display detection results with visual annotations.
- The system shall store or log detection outputs for further analysis.
- The system shall operate efficiently on both high-performance systems and edge devices.

Non-Functional Requirements

Non-functional requirements define the quality attributes and constraints under which the system operates.

Usability

The system is designed to be user-friendly and requires minimal manual intervention. Automated detection and clear visualization make it accessible even to non-technical users such as farmers.

Reliability

The system ensures consistent performance under varying environmental conditions. The use of robust deep learning models enhances reliability in real-world scenarios.

Performance

The system is optimized for real-time detection with low latency. Efficient algorithms and lightweight architecture ensure fast processing of images and video streams.

Scalability

The system is capable of handling large datasets and can be extended to support additional functionalities such as pest classification or disease detection.

Supportability

The system is cross-platform compatible and can run on multiple operating systems. It supports integration with various hardware devices such as cameras and drones.

Maintainability

The modular design of the system allows easy

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

updates, debugging, and future enhancements without affecting overall performance.

Implementation Environment

The system is implemented using Python in environments such as Jupyter Notebook and Visual Studio Code. It can be deployed on local machines or cloud platforms depending on application requirements.

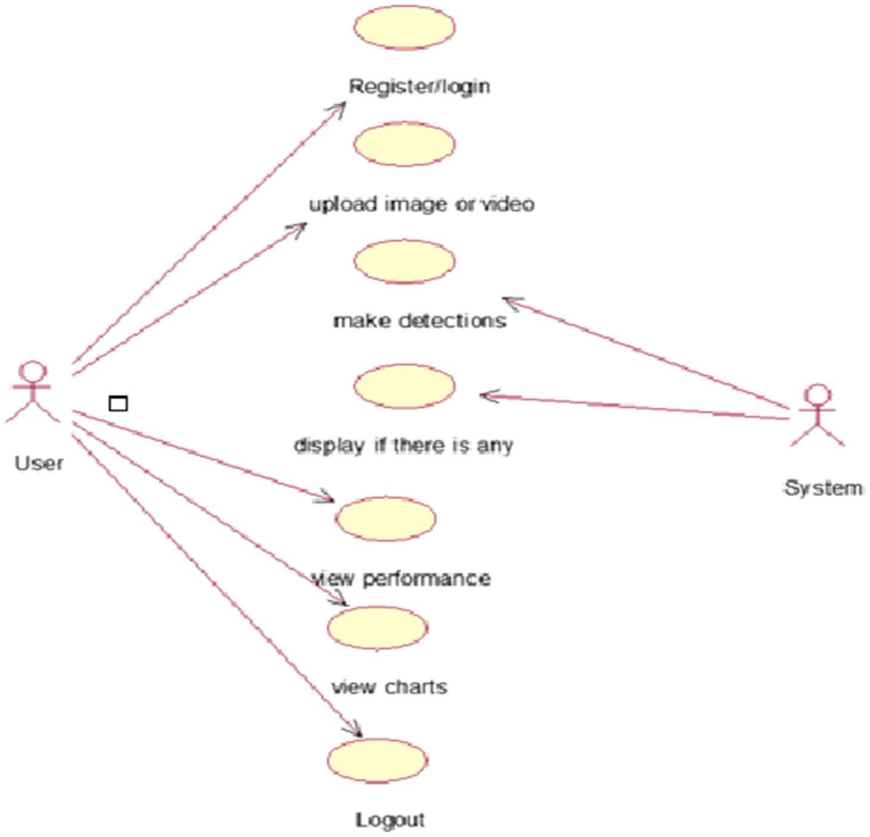
DESIGN ENGINEERING

Design engineering is a crucial phase in software development where system requirements are translated into structured models and representations. It provides a clear blueprint for implementation and ensures that the system is scalable, efficient, and maintainable.

In this project, Unified Modeling Language (UML) diagrams are used to represent different aspects of the proposed pest detection system. These diagrams illustrate system functionality, structure, data flow, and interactions between components. By defining the architecture and workflow in advance, the design phase helps in reducing development complexity and improving system quality.

UML Diagrams

Use Case Diagram



Explanation:

The use case diagram illustrates the interaction between users and the pest detection system. The primary actor is the **user (farmer or operator)** who interacts with the system to upload images or access real-time video feeds.

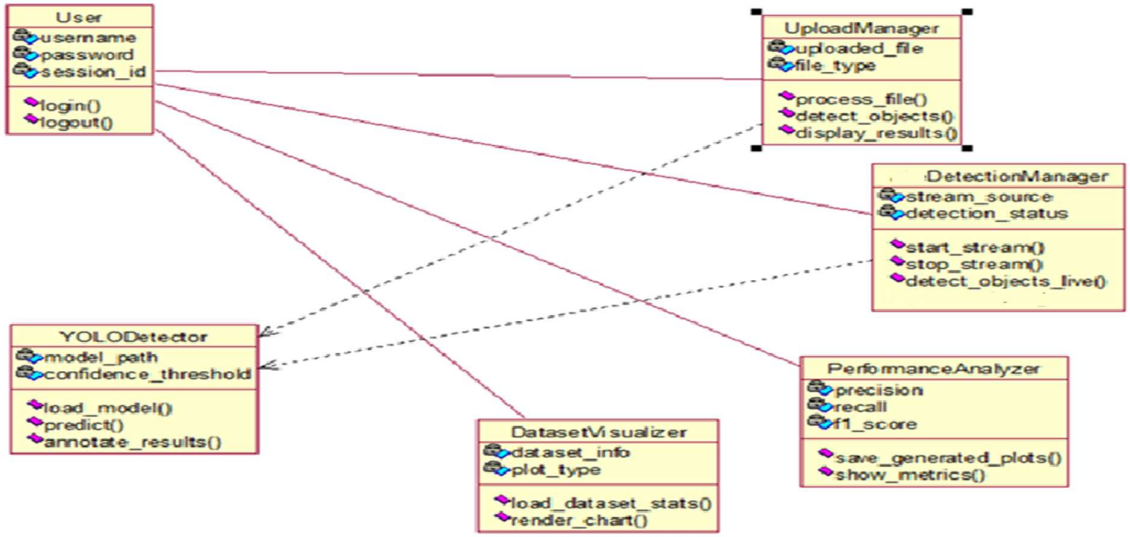
Key functionalities include:

- Input image/video data
- Initiate pest detection
- View detection results
- Monitor pest activity

This diagram helps in understanding how users engage with the system and what services are provided.

Class Diagram

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

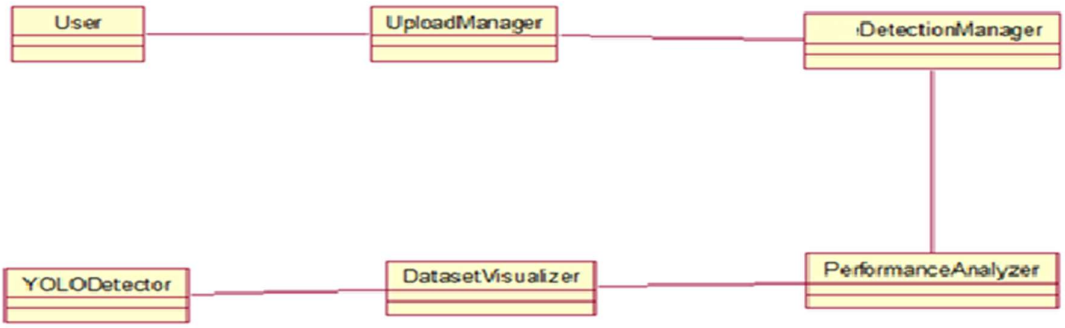


Explanation:
The class diagram represents the static structure of the system by defining classes, their attributes, and methods, along with relationships between them.
Main classes include:

- **InputHandler** (handles image/video input)
- **PreprocessingModule** (image enhancement and normalization)
- **DetectionModel (YOLOv10)** (core detection logic)
- **ResultVisualizer** (displays bounding boxes and outputs)
- **DataStorage** (stores results and logs)

These classes interact to perform end-to-end pest detection efficiently.

Object Diagram



Explanation:
The object diagram provides a snapshot of the system at a particular instance, showing how objects of different classes interact during execution.
For example:

- An input image object is processed by the preprocessing module
- The processed image is passed to the detection model
- Detection results are forwarded to the visualization module

This diagram helps in understanding real-time object interactions within the system.

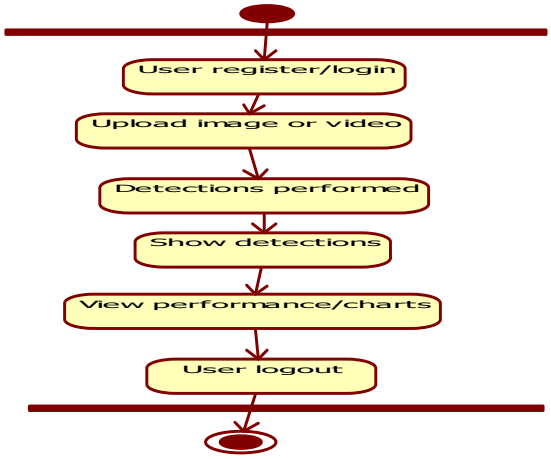
State Diagram
Explanation:
The state diagram represents the different states of the system during execution and the transitions between them.
Typical states include:

- Idle
- Input Received
- Preprocessing
- Detection in Progress
- Result Generated

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

This diagram is useful for modeling system behavior under different conditions and ensuring proper workflow transitions.

Activity Diagram

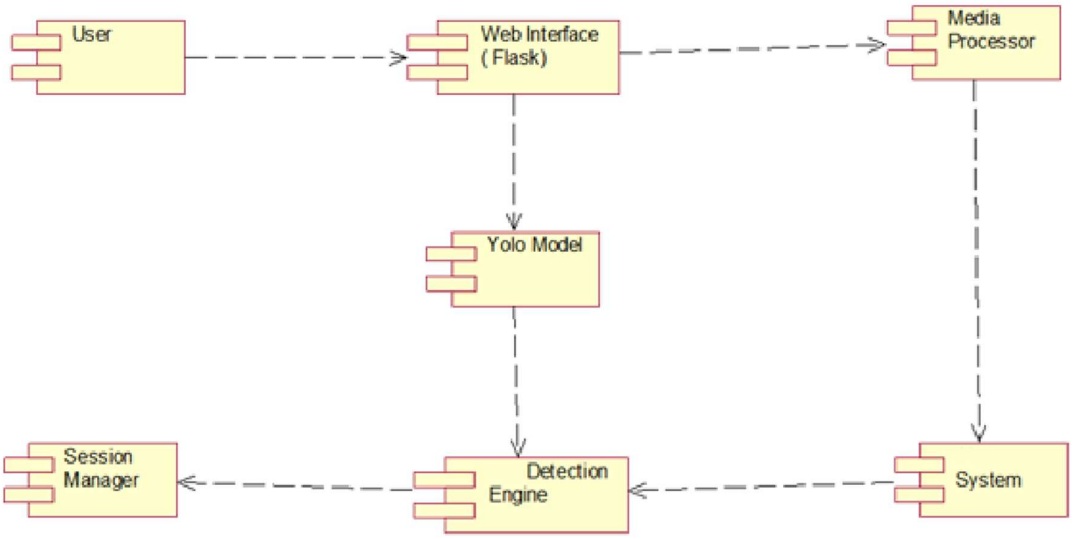


Explanation:
The activity diagram illustrates the step-by-step workflow of the pest detection process.
Workflow steps:

1. Capture or upload image
2. Preprocess input data
3. Apply YOLOv10 detection model
4. Identify insects and generate bounding boxes
5. Display results

It provides a clear representation of the system's operational flow.

Component Diagram



Explanation:
The component diagram represents the high-level architecture of the system by dividing it into independent modules.

Main components:

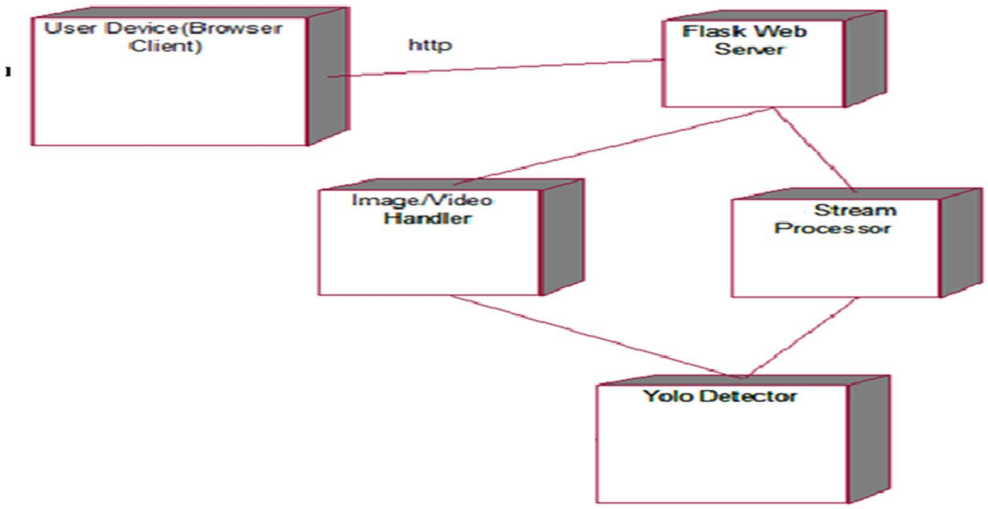
- User Interface
- Data Processing Module
- YOLOv10 Detection Engine

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

- Output Visualization Module
- Storage System

It shows how these components are interconnected and dependent on each other.

Deployment Diagram



Explanation:

The deployment diagram represents the physical implementation of the system, showing how software components are deployed on hardware devices.

Deployment setup includes:

- **User Device** (mobile/PC interface)
- **Processing Unit** (local system or cloud server)
- **Edge Devices** (drones, cameras)

This diagram ensures that the system can be effectively deployed in real-world agricultural environments.

System Architecture

Explanation:

The system architecture defines the overall structure and workflow of the proposed pest detection system.

Architecture Flow:

1. Data acquisition from camera/drone/dataset
2. Image preprocessing
3. YOLOv10-based detection
4. Feature extraction and classification
5. Output visualization with bounding boxes
6. Storage and monitoring

The architecture is designed to support real-time processing, scalability, and efficient deployment on edge devices. It ensures seamless integration of all modules to deliver accurate and timely pest detection.

DEVELOPMENT TOOLS

Python is a high-level, interpreted programming language that is widely used in modern software development, particularly in fields such as data science, machine learning, and computer vision. It is designed with a strong emphasis on readability and simplicity, allowing developers to write efficient and concise code. Python supports multiple programming paradigms, including object-oriented, procedural, and functional programming, making it a flexible choice for developing complex systems such as the proposed real-time pest detection framework.

The language was developed by Guido van Rossum in the late 1980s and officially released in 1991 at the Centrum Wiskunde & Informatica (CWI) in the Netherlands. Python was influenced by several programming languages, including ABC, C, and Unix shell scripting, and was created with the goal of improving code readability and developer productivity. Over the years, Python has evolved into one of the most popular programming languages globally, supported by a large open-source community and maintained by the Python Software Foundation.

Python plays a crucial role in this project due to its numerous advantages. As an interpreted language, it executes code line by line, eliminating the need for compilation and simplifying debugging. Its interactive nature allows developers to test code snippets in real time, which is particularly useful during model development and experimentation. Python also supports object-oriented programming, enabling modular and reusable code design. Additionally, its beginner-friendly syntax makes it accessible while still being powerful enough for

advanced applications such as deep learning and real-time object detection.

One of the key strengths of Python lies in its extensive set of features. It is easy to learn and use due to its simple syntax and minimalistic structure. The language promotes readability, making it easier to maintain and update code over time. Python is also highly portable, allowing programs to run across different platforms without modification. Its dynamic typing system and automatic memory management further enhance development efficiency. Moreover, Python supports scalability and integration with other programming languages such as C and C++, making it suitable for both small-scale and large-scale applications.

Figure : NumPy, Pandas, Matplotlib, Scikit-learn

In addition to its core features, Python provides a rich ecosystem of libraries that significantly simplify development. Libraries such as NumPy are used for numerical computations and handling multi-dimensional arrays, which are essential for processing image data. Pandas is utilized for data manipulation and preprocessing, offering efficient data structures like DataFrames. Matplotlib is employed for visualization purposes, enabling the generation of graphs and performance plots. Scikit-learn provides tools for data analysis and evaluation metrics, while OpenCV is widely used for image and video processing tasks, including real-time frame capture and enhancement. Furthermore, deep learning frameworks such as PyTorch and TensorFlow are used to design, train, and deploy the YOLOv10 model for insect detection.

Overall, Python serves as the backbone of the proposed system by providing a robust, flexible, and efficient development environment. Its combination of simplicity, powerful libraries, and strong community support makes it an ideal choice for implementing advanced artificial intelligence solutions in precision agriculture.

SOFTWARE TESTING

Software testing is a critical phase in the development lifecycle aimed at identifying errors, defects, and inconsistencies within a system. It involves systematically evaluating software components to ensure that they function according to specified requirements and meet user expectations. The primary objective of testing is to detect faults before deployment and to verify that the system performs reliably under various conditions. In the context of the proposed pest detection system, testing ensures that the model accurately detects insects, processes input data efficiently, and delivers real-time results without failure. Different types of testing are employed to validate various aspects of the system, including functionality, performance, and integration.

Developing Methodologies

The testing process begins with the formulation of a structured testing strategy designed to evaluate both general functionality and specialized features of the system across multiple environments. A comprehensive test plan is developed to define test cases, testing conditions, and expected outcomes. Quality assurance practices are strictly followed to ensure that the system adheres to the requirements specified during the design phase. The methodology includes validating data inputs, verifying detection accuracy, and ensuring smooth interaction between system modules. This structured approach helps in identifying defects early and guarantees that the application is stable, efficient, and ready for deployment.

Types of Testing

Unit Testing

Unit testing focuses on verifying the functionality of individual components within the system. Each module, such as data preprocessing, detection, and visualization, is tested independently to ensure correct operation. This type of testing validates internal logic, input-output relationships, and code flow. By isolating each unit, errors can be detected and corrected at an early stage, improving overall system reliability.

Functional Testing

Functional testing evaluates whether the system performs according to specified requirements. It ensures that all functionalities, such as image input, insect detection, and result visualization, operate correctly. The system is tested with both valid and

invalid inputs to confirm that it accepts appropriate data and handles errors effectively. Outputs are verified to ensure accuracy and consistency with expected results.

System Testing

System testing involves evaluating the complete integrated system to ensure that all components work together as intended. This type of testing verifies end-to-end functionality, including data flow between modules and overall system behavior. It ensures that the pest detection system produces consistent and predictable results under real-world conditions.

FUTURE ENHANCEMENTS

While the proposed YOLOv10-based pest detection system demonstrates significant improvements over previous approaches, there remain several opportunities for further enhancement and expansion. One promising direction is the extension of the system from general pest detection to fine-grained classification. This would enable the system not only to detect the presence of insects but also to identify specific species and classify them according to growth stages or threat levels. Such granularity would support more precise pest management strategies and improve decision-making for farmers.

Another potential enhancement involves incorporating temporal analysis and video-based tracking algorithms. By analyzing sequences of images rather than individual frames, the system could monitor insect movement patterns over time, predict potential infestation spread, and provide early warning alerts. Integrating this with weather and crop growth data could further enhance predictive modeling for dynamic pest risk assessment.

From a technical perspective, deploying the detection system using an edge–cloud hybrid architecture presents another avenue for future work. Real-time detection can continue on lightweight edge devices, such as drones or field cameras, while more computationally intensive tasks—such as long-term trend analysis and large-scale data storage—can be handled in the cloud. This approach would provide farmers with actionable insights via dashboards or mobile applications without compromising on field-side responsiveness. Furthermore, the integration of multi-modal data sources, including environmental sensors for temperature, humidity, and soil moisture, could enhance the system's context-awareness. By combining sensor data with visual detection, automated pest control interventions could be implemented more intelligently, activating mitigation measures only when necessary and thereby improving efficiency and sustainability in agricultural practices.

Conclusion

This study presents a deep learning-based pest detection system designed for precision agriculture, leveraging the YOLOv10 object detection framework. Unlike previous YOLOv8-based methods, which often focus narrowly on specific crops or insect species, the proposed system generalizes to detect a wide variety of pests across diverse agricultural environments in real time. Key architectural enhancements, including improved multi-scale feature extraction and anchor-free detection, enable YOLOv10 to achieve higher accuracy and robustness, particularly for small, overlapping, or partially occluded insects frequently encountered in real-field conditions.

The system also incorporates transfer learning, data augmentation, and hyperparameter optimization to maximize generalization capability without requiring extremely large datasets. As a result, it offers a scalable, efficient, and practical tool for farmers and agricultural stakeholders, supporting timely, data-driven pest management decisions. By reducing crop losses and facilitating sustainable farming practices, this system demonstrates the practical advantages of integrating state-of-the-art deep learning into precision agriculture. Furthermore, the study lays the foundation for future improvements, including species-level classification, predictive analytics, and integration with IoT sensor networks, to enhance the automation and intelligence of pest control solutions.

References

[1] B. Davis, E. Mane, L. Y. Gurbuzer, G. Caivano, N. Piedrahita, K. Schneider, N. Azhar, M. Benali, N. Chaudhary, R. Rivera, R. Ambikapathi, and P. Winters, Estimating Global and Country Level Employment in Agri Food Systems (FAO Statistics Working Paper Series), Rome, Italy: FAO, 2023, doi: 10.4060/cc4337en.

[2] Food and Agriculture Organization, Pest and Pesticide Management. Accessed: Jan. 10, 2024. [Online]. Available: https://www.fao.org/pest-and-pesticide-management/about/understanding-the-context

[3] R. Kestur, S. N. Omar, and S. Subhash, "Unmanned aerial system technologies for pesticide spraying," in Innovative Pest Management Approaches for the 21st Century, Singapore: Springer, 2020, pp. 37–60, doi: 10.1007/978-981-15-0794-6_3.

[4] M. Abbas et al., "Role of light traps in attracting, killing and biodiversity studies of insect pests in Thal," Pakistan J. Agricult. Res., vol. 32, no. 4, pp. 684–690, 2019.

[5] P. Trematerra and M. Colacci, "Recent advances in management by pheromones of thaumetopoea moths in urban parks and woodland recreational areas," Insects, vol. 10,

no. 11, p. 395, Nov. 2019, doi: 10.3390/insects10110395.
[6] T. R. E. Southwood and P. A. Henderson, Ecological Methods, Hoboken, NJ, USA: Wiley, 2009.
[7] K. Ennouri et al., "Usage of artificial intelligence and remote sensing as efficient devices to increase agricultural system yields," J. Food Qual., vol. 2021, pp. 1–17, Jun. 2021, doi: 10.1155/2021/6242288.
[8] H. Tian, T. Wang, Y. Liu, X. Qiao, and Y. Li, "Computer vision technology in agricultural automation—A review," Inf. Process. Agricult., vol. 7, no. 1, pp. 1–19, Mar. 2020, doi: 10.1016/j.inpa.2019.09.006.
[9] Y. Lu and S. Young, "A survey of public datasets for computer vision tasks in precision agriculture," Comput. Electron. Agricult., vol. 178, Nov. 2020, Art. no. 105760, doi: 10.1016/j.compag.2020.105760.
[10] W. Zhang et al., "Review of current robotic approaches for precision weed management," Current Robot. Rep., vol. 3, no. 3, pp. 139–151, Jul. 2022, doi: 10.1007/s43154-022-00086-5.