Cocoguard: A Comparative Study of YOLO-Family Models and Faster R-CNN for Coconut Pest Identification With Rule-Based Decision Support

Maynard A. Ermita., Arlene B. Laurel., Khristian Mae Lee., Erwin C. Martinez

BS in Computer Science, CITI Global College Inc., Cabuyao Campus Cabuyao, Laguna, Philippines

DOI: https://doi.org/10.47772/IJRISS.2026.100400350

Received: 18 April 2026; Accepted: 24 April 2026; Published: 09 May 2026

ABSTRACT

Coconut farming is a vital component of the Philippine agricultural sector; however, destructive pests such as the Asiatic Palm Weevil, Rhinoceros Beetle, Brontispa beetle, and Slug Caterpillar can significantly reduce crop yields when infestations are not detected early. Conventional pest monitoring relies on manual inspection and delayed expert consultation, making pest identification slow and often inaccessible for smallholder farmers. This study proposes CocoGuard, a mobile-accessible artificial intelligence-assisted system for coconut pest detection using deep learning object detection models. A dataset of 2,076 coconut pest images, expanded to 5,398 images through augmentation, was prepared and annotated into seven pest classes. Several object detection models were benchmarked using Precision, Recall, F1-score, and mean Average Precision. Results show that YOLO26s achieved the best performance, obtaining 92.72% mAP@0.5, 92.51% precision, and 89.49% recall while maintaining computational efficiency suitable for mobile deployment.

Keywords: Coconut Pest Detection, Image Processing, TensorFlow Lite, Mobile Agriculture, Deep Learning.

INTRODUCTION

Coconut farming plays a crucial role in the agricultural economy of the Philippines, serving as a primary source of livelihood for many rural communities and contributing significantly to both local consumption and international trade. As one of the country's major agricultural commodities, coconut production supports employment, food security, and economic stability. However, coconut plantations remain highly susceptible to pest infestations, which can damage trees and substantially reduce crop yield when not detected and managed at an early stage [1], [2].

Traditional pest monitoring methods rely on manual inspection performed by farmers or agricultural experts. Although effective, these methods are often time-consuming, labor-intensive, and dependent on the availability of specialized knowledge. In many rural areas, access to agricultural experts is limited, leading to delays in pest identification and intervention. Such delays increase the risk of infestation spread, resulting in significant economic losses and reduced agricultural productivity.

Recent advancements in artificial intelligence (AI) and deep learning have enabled the development of computer vision–based systems capable of automating pest detection through image analysis. These technologies have demonstrated strong potential in identifying agricultural pests and diseases with improved speed and accuracy [3], [4]. In particular, object detection models such as the You Only Look Once (YOLO) family have shown high efficiency in detecting small objects in complex environments while maintaining real-time processing performance [5], [6]. These characteristics make them highly suitable for deployment in mobile and field-based agricultural applications.

Despite these technological developments, existing pest detection systems are often designed for general agricultural use and lack specialization for coconut-specific pests. Furthermore, many studies focus primarily on model development without conducting comprehensive comparative evaluations of multiple object detection architectures. This limits the ability to determine the most suitable model for practical deployment. Additionally, current systems frequently lack integrated advisory mechanisms that translate detection results

into actionable pest management recommendations, reducing their effectiveness for real-world agricultural decision-making.

To address these limitations, this study proposes CocoGuard, an artificial intelligence–assisted coconut pest detection system that utilizes deep learning object detection models for image-based pest identification. The study evaluates multiple architectures, including Faster R-CNN and various YOLO-based models, to determine the most effective approach for coconut pest detection. Based on comparative analysis, the YOLO26s model was selected due to its superior detection performance and computational efficiency. The system is implemented as a mobile-accessible application that integrates pest detection with a rule-based advisory system aligned with Philippine Coconut Authority (PCA) guidelines, enabling users to receive both identification results and corresponding pest management recommendations.

This research contributes to the body of knowledge in artificial intelligence, computer vision–based agricultural monitoring, and digital decision-support systems by presenting a coconut-specific framework for automated pest detection and advisory integration. It expands existing literature by combining comparative model evaluation with a mobile-accessible application designed for real-world agricultural use. Furthermore, the study aligns with several United Nations Sustainable Development Goals (SDGs): SDG 2 (Zero Hunger), by improving agricultural productivity through early pest detection; SDG 9 (Industry, Innovation, and Infrastructure), by promoting the adoption of AI-driven technologies in agriculture; SDG 12 (Responsible Consumption and Production), by enabling more efficient pest management practices; SDG 13 (Climate Action), by enhancing resilience against pest outbreaks influenced by environmental conditions; and SDG 17 (Partnerships for the Goals), by encouraging collaboration between researchers, agricultural institutions such as the Philippine Coconut Authority, and farming communities to support sustainable and technology-driven agricultural development.

METHODS

A. Conceptual Framework

To provide a structured representation of the study, a conceptual framework was developed to illustrate the overall workflow of the CocoGuard system, as shown in Fig. 1. The framework presents the sequence of processes involved in transforming raw input data into meaningful pest detection outputs and advisory recommendations. It defines the relationship between the major components of the system, including data acquisition, preprocessing, model training, deployment, and user interaction. This structured approach ensures a clear understanding of how data flows through the system and how each component contributes to the overall functionality of CocoGuard.

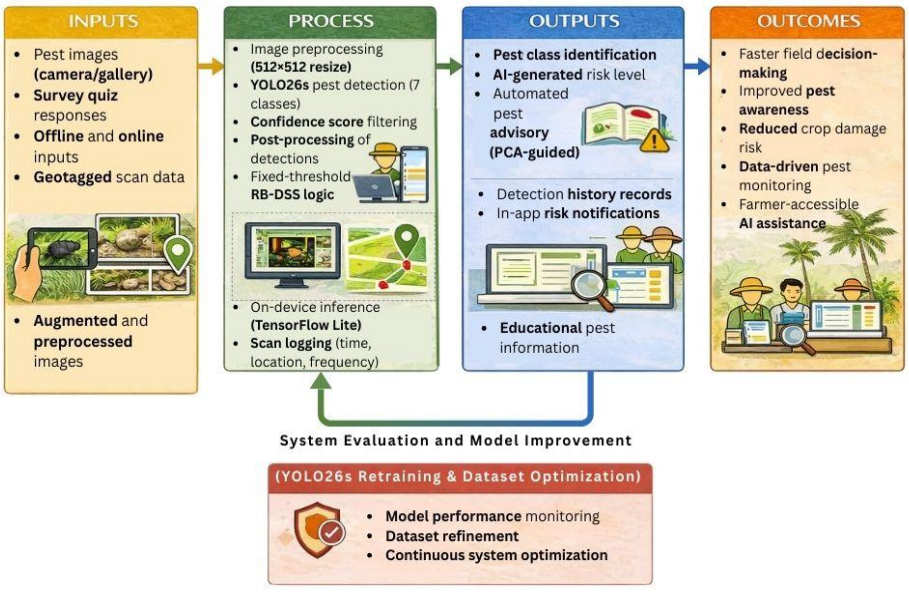

Figure 1. Conceptual Framework

The system begins with data acquisition, where coconut pest images are obtained either from dataset sources or through user-generated inputs such as camera capture and image upload. These images undergo preprocessing procedures, including resizing, normalization, and augmentation, to standardize input quality and improve model generalization. The processed dataset is then utilized in training and evaluating multiple deep learning object detection models under a controlled experimental setup. After training, the selected model is deployed within the system to perform real-time pest detection by generating outputs such as bounding boxes, class labels, and confidence scores. These detection results are further processed by a rule-based advisory component that provides pest-specific management recommendations aligned with established agricultural guidelines. Overall, the conceptual framework illustrates how CocoGuard integrates artificial intelligence with decision support mechanisms to deliver actionable insights for coconut pest monitoring.

B. Dataset Collection and Preparation

The dataset used in this study consists of coconut pest images sourced primarily from the Philippine Coconut Authority (PCA) and related institutional references. These images represent common coconut pests observed in real agricultural environments and were selected to support domain-specific pest identification. A total of 2,076 base images were collected and organized into a structured dataset. To enhance dataset diversity and improve model robustness, controlled data augmentation techniques were applied, increasing the dataset size to 5,398 images.

All images were annotated using bounding box labeling in YOLO format to enable object detection training. The dataset includes seven predefined pest classes: Asiatic Palm Weevil (Adult), Asiatic Palm Weevil (Larvae), Brontispa (Adult), Brontispa (Pupa), Rhinoceros Beetle, Slug Caterpillar, and White Grub. These classes were selected based on their agricultural relevance and frequency of occurrence in coconut plantations. The annotation process focused on accurately identifying pest locations and distinguishing between life stages to improve classification performance during model training.

Preprocessing and augmentation were performed using the Roboflow platform to standardize image quality and simulate real-world environmental variations. As shown in Fig. 2, preprocessing steps included auto-orientation correction, resizing to a fixed resolution of 512×512 pixels, and contrast enhancement. Data augmentation techniques such as horizontal flipping, rotation (−10° to +10°), brightness and exposure adjustments, noise injection, and slight blurring were applied to improve model generalization under varying field conditions.

Finally, the dataset was partitioned into training, validation, and testing subsets using an 80/10/10 split to ensure reliable and unbiased model evaluation. Specifically, 1,700 images were allocated for training, 208 images for validation, and 207 images for testing. This structured dataset preparation ensures consistency across all evaluated models and supports fair comparison under a controlled experimental setup.

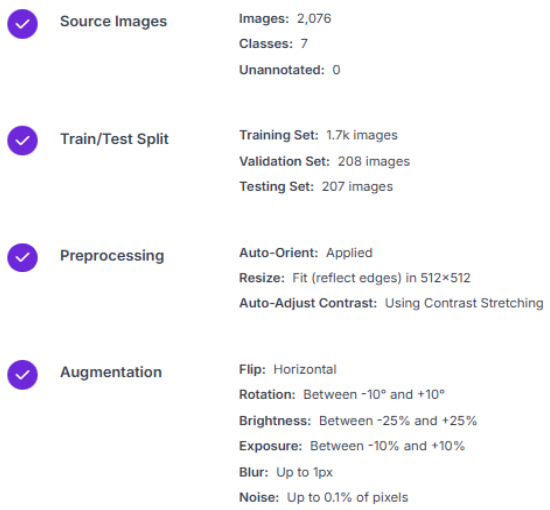

Figure 2. Roboflow Preprocessing and Augmentation Configuration for the PCA-Sourced Coconut Pest Dataset

C. Model Selection and Training

This study employed a comparative approach to evaluate multiple deep learning object detection models for coconut pest identification. The models considered include Faster R-CNN as a representative two-stage detector and several YOLO-based architectures, namely YOLOv5u_small, YOLOv8n, YOLOv8s, YOLOv9s, YOLOv11s, and YOLO26s. These models were selected based on their established use in object detection tasks and their applicability to real-time agricultural monitoring systems. The inclusion of both two-stage and single-stage detectors allows for a comprehensive assessment of detection performance and computational efficiency under a unified experimental framework.

All models were trained using the same dataset configuration, annotation format, preprocessing pipeline, and input resolution to ensure consistency and fairness in comparison. Training was conducted in a Google Colab environment utilizing an NVIDIA L4 GPU to support deep learning computations. Each model was trained using identical parameters, including image resolution of 512×512 pixels, batch size, and number of training epochs. During the training process, validation metrics were monitored to assess model learning behavior and to minimize the risk of overfitting.

The training pipeline follows a standard object detection workflow, where annotated images are fed into the model to learn feature representations corresponding to different pest classes. The models generate predictions in the form of bounding boxes, class labels, and confidence scores, which are refined through internal mechanisms such as non-maximum suppression. This process enables the models to detect and localize multiple pest instances within a single image under varying environmental conditions.

Faster R-CNN was included in the evaluation as a baseline model due to its strong performance in object detection tasks, despite its higher computational requirements. In contrast, YOLO-based models were considered for their ability to perform real-time detection through single-stage processing, making them suitable for mobile and field-based applications. The comparative training setup enables the identification of a model that balances detection accuracy and computational efficiency for integration into the CocoGuard system.

D. System Design and Development

The CocoGuard system was designed as a mobile-accessible application integrated with a web-based administrative platform to support both field-level pest detection and centralized monitoring. The overall system architecture, as illustrated in Fig. 3, consists of two primary components: the user-facing mobile application and the administrator dashboard. These components are interconnected through a backend processing layer responsible for handling data flow, model inference, and system transactions. This architecture ensures seamless communication between users and administrators while maintaining system efficiency and scalability.

The mobile application serves as the primary interface for farmers and end-users, allowing them to capture images using the device camera or upload images from the gallery for pest detection. The captured images are processed by the embedded deep learning model, which generates detection outputs including bounding boxes, class labels, and confidence scores. To enhance usability in real-world scenarios, the application also includes a survey-based input mechanism that enables users to identify potential pests based on observed symptoms when image data is unavailable. All detection results, along with pest identification and advisory information, are stored locally using a lightweight database to support offline functionality and enable users to access their scan history.

To enable real-time processing on mobile devices, the trained object detection model was converted into TensorFlow Lite (.tflite) format for on-device inference. This allows the system to perform pest detection efficiently without requiring continuous internet connectivity, making it suitable for deployment in remote agricultural areas. In addition, a rule-based advisory module is integrated into the system to generate pest-specific management recommendations based on detection results and established agricultural guidelines. The administrative platform complements the mobile application by providing a web-based dashboard for

monitoring detection records, visualizing pest distribution trends, managing advisory content, and tracking system activity. This integrated system design ensures that CocoGuard effectively supports both real-time pest identification and data-driven agricultural monitoring.

Figure 3. System Architecture (User and Admin System Flow)

E. Performance Evaluation Metrics

To evaluate the performance of the trained object detection models, a set of standard quantitative metrics was employed to measure both classification accuracy and localization capability. These metrics are computed by comparing the predicted bounding boxes and class labels generated by the models against the annotated ground truth data. The evaluation focuses on determining how accurately the models can detect and correctly classify coconut pests under varying image conditions.

The primary evaluation metric used in this study is mean Average Precision (mAP), which provides an overall measure of detection performance across all pest classes. The computation of mAP is based on the precision–recall relationship for each class and is averaged to obtain a single performance score. The evaluation incorporates the Intersection over Union (IoU) threshold to determine whether a predicted bounding box sufficiently overlaps with the ground truth annotation. A prediction is considered correct when the IoU value exceeds a predefined threshold, ensuring that both spatial localization and classification accuracy are taken into account during evaluation.

In addition to mAP, the study utilizes precision, recall, and F1-score to provide a more detailed assessment of model performance. Precision measures the proportion of correctly predicted pest detections relative to all predicted detections, while recall evaluates the ability of the model to identify all relevant pest instances present in the dataset. The F1-score is calculated as the harmonic mean of precision and recall, offering a balanced measure of both metrics. These evaluation metrics are mathematically defined as follows:

$$Precision = \frac{TP}{TP+FP} \qquad \text{(Eq.1)}$$

$$Recall = \frac{TP}{TP+FN} \qquad \text{(Eq.2)}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \qquad \text{(Eq.3)}$$

where:

TP represents true positive detections,

FP represents false positives, and

FN represents false negatives

These metrics collectively establish a standardized and objective basis for comparing the performance of different models under a consistent evaluation framework.

RESULTS AND DISCUSSIONS

A. Comparative Model Performance

This section presents the comparative evaluation of the trained object detection models based on standard performance metrics, including precision, recall, F1-score, and mean Average Precision (mAP). The evaluation was conducted using a consistent dataset and training configuration to ensure a fair comparison across all models. The results provide insight into the effectiveness of each model in detecting and classifying coconut pests under varying conditions.

Table 1 summarizes the performance metrics obtained from each evaluated model, including Faster R-CNN and multiple YOLO-based architectures. Among the models tested, the YOLO26s architecture demonstrated superior performance in terms of detection accuracy and overall consistency. It achieved the highest values across key metrics, indicating its effectiveness in both identifying pest classes and localizing them within images. In contrast, Faster R-CNN exhibited lower performance, particularly in terms of detection speed and overall efficiency, making it less suitable for real-time mobile deployment.

Table 1. Comparative Model Performance Metrics Across Evaluated Algorithms

Algorithm	Accuracy	Recall	Precision	AUC-ROC	Training Time	Inference Time
YOLOv5 u_small	0.94	0.91	0.913	0.78	28m 5s	28.02 ms
YOLOv8s	0.926	0.87	0.96	0.778	30m 36s	24.82 ms
YOLOv8n	0.924	0.86	0.93	0.75	24m 4s	24.28 ms
YOLOv9s	0.927	0.89	0.91	0.79	55m 5s	54.87 ms
YOLOv11s	0.926	0.869	0.964	0.786	34m 1s	33.87 ms
YOLO26s	0.927	0.89	0.93	0.80	46m 1s	35.86 ms
Faster R-CNN	0.90	0.74	0.90	0.69	3h 30m 18s	46.01 ms

The comparative results are further illustrated in Figure 4, which visualizes the performance differences among the evaluated models. The figure highlights the advantage of YOLO-based architectures in achieving a balance between accuracy and computational efficiency. The consistent performance of YOLO26s across all evaluation metrics suggests that it is well-suited for deployment in real-time agricultural applications, particularly in environments where computational resources are limited.

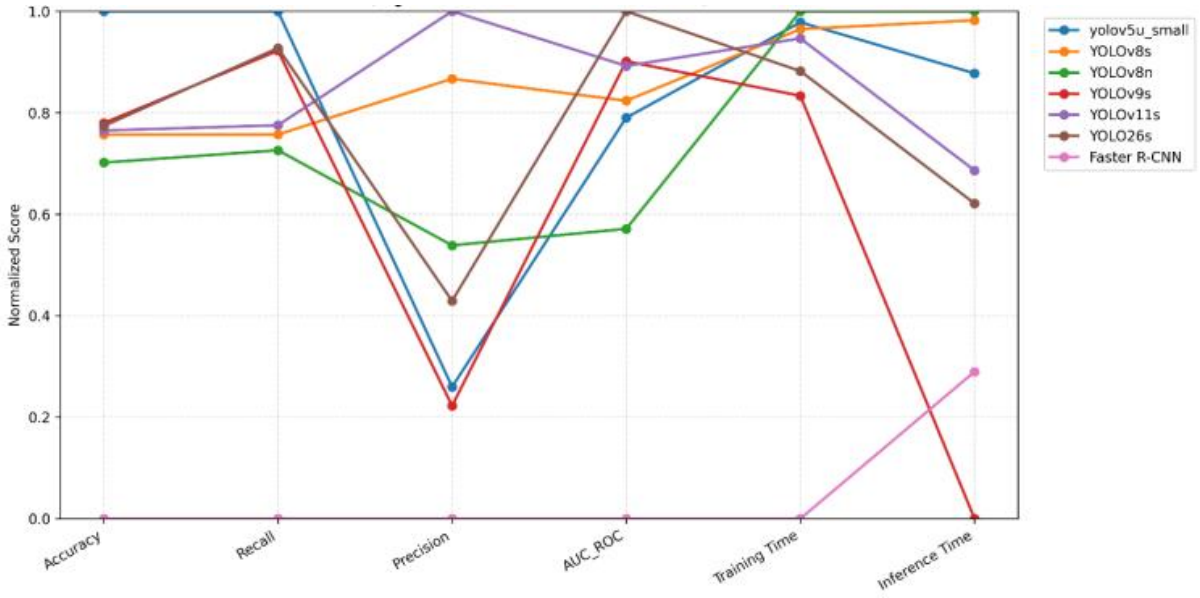

Figure 4. Comparative Model Performance

Overall, the results indicate that while multiple models are capable of performing object detection, the selection of an appropriate architecture is critical in achieving optimal performance. The superior results obtained by YOLO26s support its integration into the CocoGuard system as the primary detection model, enabling accurate and efficient coconut pest identification.

B. YOLO26s Training Performance Analysis

Following the model comparison, a detailed analysis of the YOLO26s training performance was conducted to evaluate its learning efficiency and convergence behavior. This analysis examines how the model's performance metrics evolved throughout the training process, providing insight into its ability to learn discriminative features for coconut pest detection while maintaining generalization capability.

The training performance curves of YOLO26s are shown in Figure 5, illustrating the progression of training loss, validation loss, precision, and recall over the training epochs. Both training and validation losses exhibit a steady decreasing trend, indicating effective optimization of model parameters during training. The absence of abrupt fluctuations or divergence between the loss curves suggests that the model maintained stable learning throughout the training process. Additionally, precision and recall values show gradual improvement before reaching a plateau, reflecting the model's ability to consistently detect pest instances with increasing accuracy as training progresses.

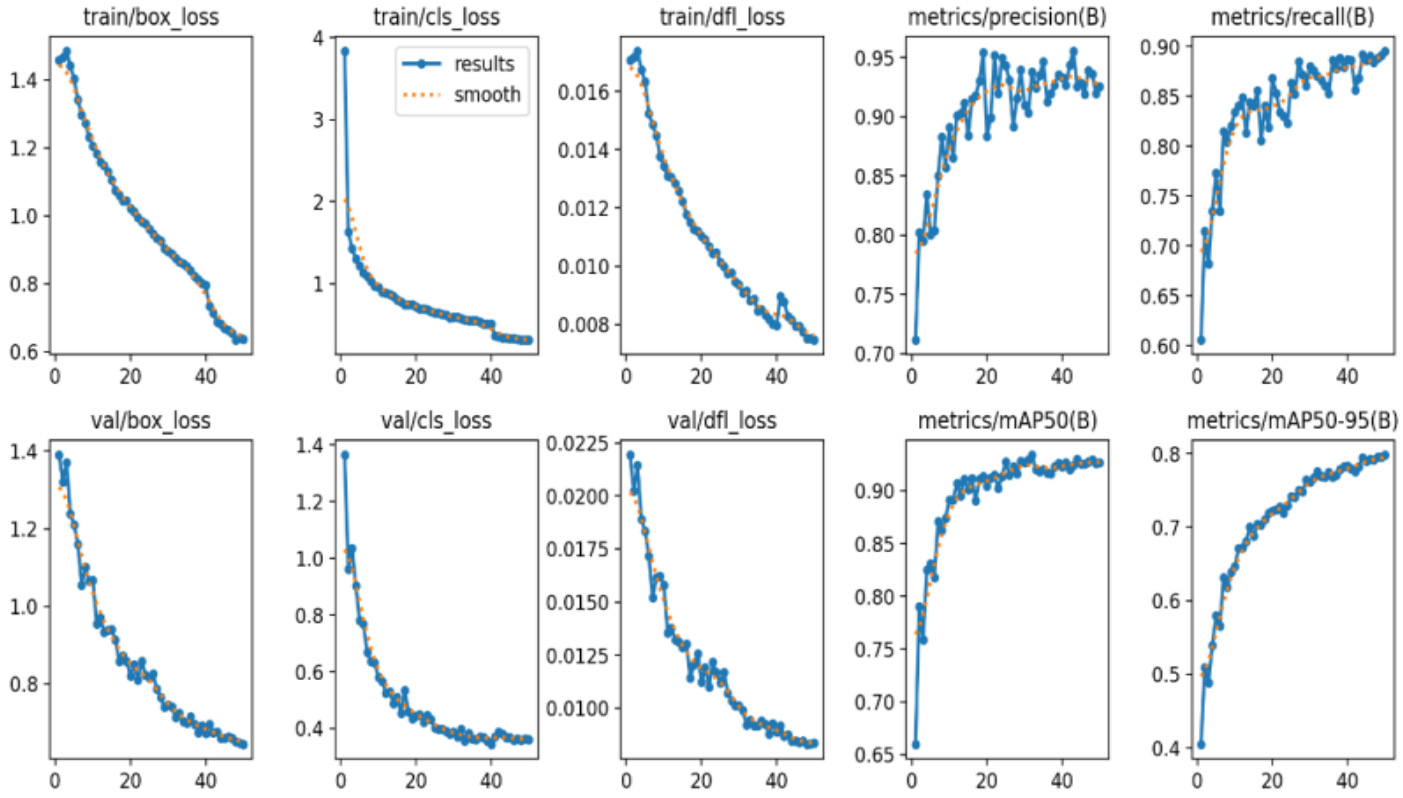

Figure 5. YOLO26s Training Performance Curves

The close alignment between training and validation metrics indicates that the model achieved good generalization without significant overfitting. This behavior can be attributed to the applied preprocessing and data augmentation techniques, which introduced variability in the dataset and improved the model's robustness to real-world conditions. Furthermore, the stabilization of performance metrics in later epochs suggests that the model reached convergence, where further training yields minimal performance gains.

C. Confusion Matrix and Classification Performance Analysis

To further evaluate the classification performance of the selected YOLO26s model, a confusion matrix analysis was conducted to examine its ability to correctly distinguish among the different coconut pest classes.

The confusion matrix provides a detailed breakdown of correct and incorrect predictions, allowing for the identification of class-specific strengths and misclassification patterns.

The normalized confusion matrix of YOLO26s is presented in Figure 6, where each row represents the actual class and each column represents the predicted class. The diagonal elements of the matrix indicate correct classifications, while off-diagonal values represent misclassifications between classes. The results show that the majority of predictions are concentrated along the diagonal, indicating that the model achieved high classification accuracy across most pest categories.

Figure 6. Confusion Matrix of YOLO26s

Despite the overall strong performance, minor misclassifications can be observed between certain pest classes, particularly those with similar visual characteristics or overlapping features. For example, confusion between different life stages of the same pest species, such as larvae and adult forms, may occur due to similarities in texture, shape, or color patterns. Additionally, variations in image quality, lighting conditions, and background complexity may contribute to occasional classification errors.

The analysis of these misclassification patterns provides valuable insights into the limitations of the model and highlights potential areas for improvement. Enhancing dataset diversity, refining class annotations, or incorporating additional training samples for visually similar classes may help reduce classification ambiguity. Nonetheless, the dominance of correct predictions across all classes demonstrates that YOLO26s maintains a high level of reliability in distinguishing coconut pests under varying conditions.

D. System Functionality Testing Results

To evaluate the operational reliability of the CocoGuard system, functionality testing was conducted on both the mobile application and the web-based administrative platform. The testing aimed to verify whether each

system component performed according to its intended function under defined test conditions. A black-box testing approach was employed, focusing on validating system inputs and outputs without examining the internal code structure.

Table 2. Mobile Application Functionality Testing Results

Module	Expected Output	Result
User Registration and Login	Successful authentication and secure account access	Passed
Image Capture (Camera Scan)	Captures and processes pest images for detection	Passed
Image Upload (Gallery)	Uploads image and send it to YOLO26s detection pipeline	Passed
Answer Survey	Generates pest identification based on user responses	Passed
AI Pest Detection (YOLO26s)	Detects pest type with confidence score and bounding box output	Passed
Detection Result Display	Displays pest label, confidence percentage, and risk level	Passed
PCA Advisory and Management Recommendations	Provides pest-specific PCA-guided management strategies	Passed
Detection History Tracking	Stores and retrieves previous scan records in SQLite database	Passed
Push Notifications and Pest Alerts	Sends alerts for high-risk or critical pest detections	Passed
Knowledge Base Access	Displays categorized pest management articles and guides	Passed

The results of the mobile application testing are summarized in Table 2, which outlines the evaluation of key functional modules, including image capture, image upload, pest detection, advisory generation, and local data storage. All tested features successfully met their expected outputs, indicating that the mobile application operates reliably across its core functionalities. The consistent “Pass” results demonstrate that the system is capable of supporting real-time pest detection and providing immediate feedback to users in field conditions.

Table 3. Admin System Functionality Testing Results

Module	Expected Output	Result
Admin Login Authentication	Secure admin access to the dashboard	Passed
Dashboard Overview Panel	Displays total scans, active users, and system metrics	Passed
Detection Records Management	Shows submitted pest detections and scan history	Passed
Pest Analytics Visualization	Displays pest distribution charts and trend analysis	Passed
Monthly and 7-Day Scan Trends	Generates graphical activity reports	Passed
Top Detected Pests Monitoring	Identifies most frequently detected pest classes	Passed
Knowledge Base Management	Add, edit, and manage pest management articles	Passed
User Feedback Management	Stores and displays farmer feedback records	Passed
Activity Logs Monitoring	Records system actions and administrative activities	Passed

Similarly, the functionality testing results for the administrative platform are presented in Table 3. The evaluation covered essential features such as data monitoring, user management, advisory content management, and system record tracking. The results indicate that all administrative functions performed as expected, confirming the reliability of the platform in managing system data and supporting backend operations.

The successful performance of both the mobile and administrative components demonstrates the robustness of the overall system architecture. The absence of functional errors during testing suggests that the integration between the detection model, mobile interface, and backend system is stable and effective. This ensures that CocoGuard can provide a seamless user experience while maintaining accurate data handling and system responsiveness.

E. System Interface and Implementation

To further illustrate the practical implementation of the CocoGuard system, the user interfaces of both the mobile application and the administrative dashboard are presented. These interfaces demonstrate how the system translates detection outputs into user-accessible information and actionable insights.

ISSN No. 2454-6186 | DOI: 10.47772/IJRISS | Volume X Issue IV April 2026

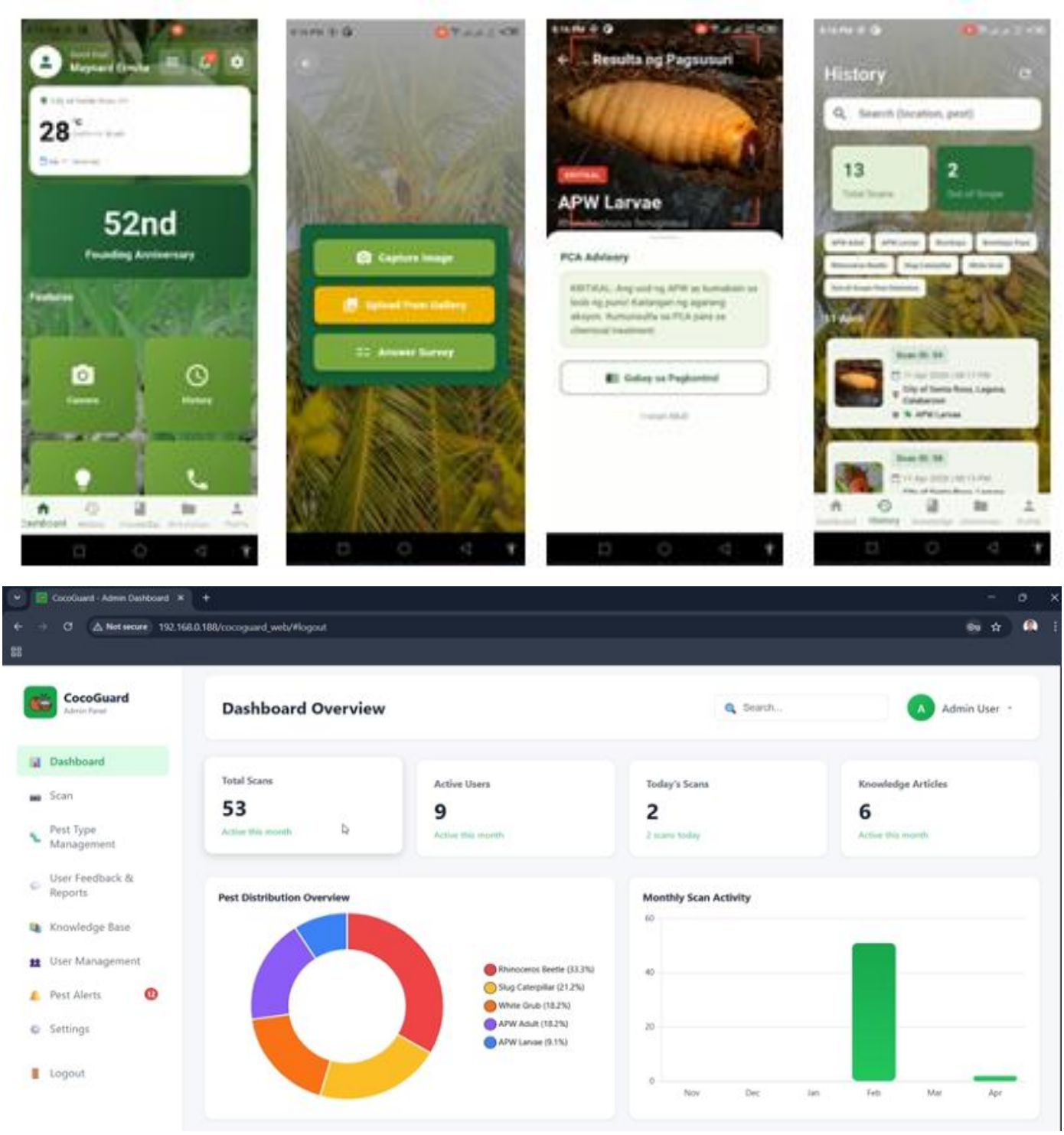

Figure 7. CocoGuard System Interfaces (Mobile Application and Admin Dashboard)

The administrative dashboard interface, also shown in Figure 7, enables system administrators to monitor detection records, analyze pest trends, and manage advisory content. Features such as data visualization, activity logs, and knowledge base management support centralized control and decision-making. This dual-interface design ensures that the system addresses both field-level usability and backend data management requirements.

The mobile application interface, shown in Figure 7, provides functionalities such as image capture, pest detection, and advisory display. Users can scan or upload images, view detection results with bounding boxes and confidence scores, and receive pest-specific recommendations aligned with Philippine Coconut Authority (PCA) guidelines. The interface is designed to be intuitive and accessible, allowing users with minimal technical background to effectively utilize the system in real-world agricultural settings.

F. User Acceptance Testing (UAT) Results and Analysis

To evaluate the usability and overall acceptability of the CocoGuard system, a User Acceptance Testing (UAT) was conducted based on selected quality characteristics adapted from ISO standards. The evaluation focused on key dimensions, including functionality, usability, reliability, and efficiency, to determine how well the system meets user expectations in practical agricultural scenarios.

The results of the UAT are summarized in Table 4, which presents the mean scores and corresponding qualitative interpretations for each evaluated criterion. The findings indicate that the system achieved an overall rating within the "Agree" range, reflecting a positive user perception of its performance and usability. Among the evaluated criteria, functionality and usability received relatively high ratings, suggesting that users were able to effectively interact with the system and utilize its core features, such as pest detection and advisory generation, with minimal difficulty.

Table 4. User Acceptance Testing Summary Based on ISO Quality Metrics

Evaluation Criteria	Mean Score (5-point scale)	Interpretation
Functionality	4.40	Agree
Reliability	4.29	Agree
Usability	4.31	Agree
Performance Efficiency	4.24	Agree
Maintainability	4.30	Agree
Portability	4.32	Agree
Compatibility	4.35	Agree
Security	4.35	Agree
Overall User Acceptance	4.32	Agree

In terms of reliability and efficiency, the system also demonstrated consistent performance, although slightly lower ratings in these areas may indicate opportunities for further optimization. Factors such as processing time, response consistency, and system behavior under varying conditions may have influenced user feedback. These observations highlight the importance of continuous refinement to enhance system responsiveness and robustness, particularly in real-world deployment environments.

CONCLUSION

The comparative evaluation of the selected object detection models revealed clear differences in performance, efficiency, and suitability for real-time agricultural deployment. Among the evaluated approaches, the YOLO26s model demonstrated the most balanced and robust overall performance. It achieved the highest results across key evaluation metrics, including precision, recall, F1-score, and mean Average Precision (mAP), indicating superior capability in accurately detecting and localizing coconut pests. Its stable training behavior and strong generalization further highlight its effectiveness in handling variations in image quality and environmental conditions, making it highly suitable for integration into real-world mobile-based detection systems.

Other evaluated models, including Faster R-CNN and earlier YOLO variants, also demonstrated functional detection capabilities but exhibited limitations in either computational efficiency or consistency of performance. Faster R-CNN, while reliable in object detection tasks, showed lower efficiency due to its two-stage architecture, making it less appropriate for real-time applications. Similarly, other YOLO-based models provided competitive results but did not achieve the same level of accuracy and stability as YOLO26s. These differences emphasize the importance of selecting an appropriate model that balances both detection performance and computational requirements for practical deployment.

Beyond model performance, the system-level evaluation confirmed that the CocoGuard system operates reliably across both mobile and administrative platforms. The functionality testing results verified that all core features, including image capture, pest detection, advisory generation, and data management, performed as

expected. Furthermore, the User Acceptance Testing results indicated that users positively evaluated the system in terms of usability, functionality, and overall effectiveness, confirming its readiness for real-world application.

Overall, the findings demonstrate that the CocoGuard system successfully integrates deep learning, mobile computing, and decision-support mechanisms to address the challenges of coconut pest detection and management. The strong performance of the YOLO26s model, combined with positive system validation results, supports its deployment as an effective tool for agricultural monitoring. Future work may focus on expanding the dataset, improving detection of visually similar pest classes, and optimizing system performance to further enhance scalability and long-term usability in diverse agricultural environments.

REFERENCES

1. B. J. M. Almarinez et al., "Biological control: A major component of the pest management program for the invasive coconut scale insect Aspidiotus rigidus Reyne in the Philippines," Insects, vol. 11, no. 11, p. 745, 2020.
2. Z.-J. Astoveza et al., "COCODY: Identifying coconut disease and pest infestation using CNN algorithm with simulation," AIP Conf. Proc., vol. 3287, p. 030008, 2024.
3. C. M. Badgujar, A. Poulose, and H. Gan, "Agricultural object detection with YOLO algorithm: A bibliometric and systematic review," arXiv preprint, 2024.
4. U. Barman, C. Pathak, and N. K. Mazumder, "Comparative assessment of pest damage identification of coconut plant," Multimedia Tools Appl., vol. 82, pp. 25083–25105, 2023.
5. P. Christakakis et al., "Smartphone-based citizen science tool for plant disease and insect pest detection," Technologies, vol. 12, no. 7, p. 101, 2024.
6. M. Daga, D. Parikh, and S. P. Ramu, "DeepSeqCoco: A mobile-friendly deep learning model for coconut disease detection," in Proc. IEEE Conf., 2024.
7. M. E. A. Dumale et al., "Evaluation of biological control through AutoPPLeX in smart farming," in DLSU Res. Congress Proc., 2022.
8. C. W. Fodulla, S. C. A. Manaig, and V. A. A. Gabin, "Design of a Raspberry Pi-based coconut pest detection system," Preprint, 2023.
9. J. Frayco, "Digital technologies in plant disease and pest management," 2025.
10. M. D. Gerance et al., "SaBaTech: A banana pest and disease detection web application," Asian J. Data Sci. AI, 2023.
11. A. C. Guiam et al., "SPIDTECH: A mobile application for pest monitoring in the Philippines," 2021.
12. M. L. A. Hosang et al., "Pest and disease challenges and control strategies for coconut," IOP Conf. Ser. Earth Environ. Sci., vol. 1235, p. 012010, 2023.
13. M. E. Karar et al., "Mobile application for agricultural pest recognition using deep learning," Ain Shams Eng. J., vol. 12, no. 4, pp. 3747–3756, 2021.
14. S. Khalid et al., "Small pest detection using deep learning object detection," Sustainability, vol. 15, no. 8, p. 6815, 2023.
15. R. A. Latina et al., "PCR-based marker for detection of Aspidiotus rigidus," J. Asia-Pacific Entomol., vol. 24, no. 4, pp. 873–881, 2021.
16. K.-R. Li et al., "Pest detection based on lightweight Faster R-CNN," Agronomy, vol. 14, no. 10, p. 2303, 2024.
17. R. I. Marasigan et al., "CocoSense: Coconut tree detection using YOLOv7," E3S Web Conf., vol. 488, p. 03015, 2024.
18. R. K. Megalingam et al., "Deep learning approach to identify pests in coconut trees," in Proc. IEEE, 2024.
19. M. L. Moreno, J. K. M. Kuwornu, and S. Szabo, "Overview of the coconut supply chain in the Philippines," J. Sustain. Dev., 2020.
20. A. A. Murat and M. S. Kiran, "A review on YOLO versions for object detection," J. King Saud Univ. Comput. Inf. Sci., 2025.
21. PCAARRD, "Coconut," 2023. [Online]. Available.
22. A. Redford et al., "Digital identification tools for plant biosecurity," Plant Protection Quarterly, 2022.
23. J. Ricofuerto, "Coconut palm tree detection using deep learning," 2025.

24. R. Sapkota et al., "YOLO26: Architectural enhancements for real-time detection," arXiv preprint, 2025.
25. A. Sharma, V. Kumar, and L. Longchamps, "Comparative performance of YOLO and Faster R-CNN models," Smart Agric. Technol., vol. 6, p. 100324, 2024
26. Z. Wang, L. Qiao, and M. Wang, "Agricultural pest detection based on Faster R-CNN," in Proc. SPIE, 2022.