YOLO-Driven Lightweight Mobile Real-Time Pest Detection and Web-Based Monitoring for Sustainable Agriculture

Wong Min On[1], Nirase Fathima Abubacker[2]
BSc in Data Analytics, Asia Pacific University of Technology & Innovation, Kuala Lumpur, Malaysia[1]
School of Computing, Asia Pacific University, Kuala Lumpur, Malaysia[2]

***Abstract*—Nowadays, pest infestations cause significant reductions in agricultural productivity all over the world. To control pests, farmers often apply excessive volumes of pesticides due to the difficulty of manually detecting the pest at an early stage. Their overuse of pesticides has led to environmental pollution and health risks. To address these challenges, many novel systems have been developed to identify pests early, allowing farmers to be alerted about the exact location where pests are detected. However, these systems are constrained by their lack of real-time detection capabilities, limited mobile integration, ability to detect only a small number of pest classes, and the absence of a web-based monitoring system. This paper introduces a pest detection system that leverages the lightweight YOLO deep learning framework and is integrated with a web-based monitoring platform. The YOLO object detection architectures, including YOLOv8n, YOLOv9t, and YOLOv10-N, were studied and optimized for pest detection on smartphones. The models were trained and validated using merging publicly datasets containing 29 pest classes. Among them, the YOLOv9t achieves top performance with a mAP@0.5 value of 89.8%, precision of 87.4%, recall of 84.4%, and an inference time of 250.6ms. The web-based monitoring system enables dynamic real-time monitoring by providing farmers with instant updates and actionable insights for effective and sustainable pest management. From there, farmers can take necessary actions immediately to mitigate pest damage, reduce pesticide overuse, and promote sustainable agricultural practices.**

Keywords*—Pest detection; YOLO; deep learning; real-time monitoring; smartphone application; web-based platform; object detection; pest management; pesticide reduction; sustainable agriculture

I. INTRODUCTION

Agriculture is a key activity that provides humans with basic needs including food, medicine, shelter, and clothing. Recent research indicates that 35-40% of the world's land area is used for agricultural purposes [1]. With the global population expected to grow to approximately 9.7 billion by 2050, the demand for agricultural output has never been higher [2]. Unfortunately, various factors continue to influence agricultural productivity, leading to reduced yields and placing farms at serious risk. According to the Food and Agricultural Organization (FAO), insects, pests, illnesses, and weed infestations are estimated to damage around 40% of agricultural production per year [3]. The main reason behind this issue is that the control levels of pests are not always attained due to the absence of an accurate diagnosis at the right moment.

Today, many farmers still depend on traditional methods that involve manual field inspections by themselves to determine signs of infestation. In simple words, they rely on their own knowledge and experience when faced with a pest infestation. Due to the wide variety of crops and pests, manual detection is challenging and error prone [4]. Aside from that, the frequent emergence and recurrence of pests has further hampered the effectiveness of traditional early detection measures [5]. Also, insufficient knowledge leads the farmers to use a variety of pesticides as their primary method for eliminating pests in order to protect crops and increase the quality of their yields [6]. Excessive use or misuse of pesticides can harm the ecosystem and potentially cause long-term diseases like cancer, respiratory infections and fetal deaths [7]. To reduce the widespread reliance on harmful pesticides, modern technology plays a key role in detecting pests at an early stage in agriculture.

Over the past few years, deep learning has changed the field of machine learning with the potential in revolutionizing numerous applications, particularly in object detection [8]. This has opened the possibility for innovative solutions that could tackle various agricultural challenges. Nevertheless, many existing systems designed for pest detection in agriculture have significant limitations. They often lack real-time processing capabilities, have minimal mobile integration, and can detect only a limited number of pest classes. Hence, most current systems do not perform well under diversified agricultural settings. Moreover, although some of these systems have already been integrated into smartphones, they lack a web-based monitoring system with real-time updating and tracking features necessary for effective pest management.

To overcome these challenges, one of the most prominent deep learning algorithms, YOLO (You Only Look Once), is remarkable with its ability to perform quick detection with high accuracy [9]. It is well-suited for applications that need rapid and accurate object detection, which includes identifying pests in agriculture. The YOLO family of algorithms has been very successful in many studies including those in agriculture. Among these, YOLOv8 has proven to be especially effective with many studies demonstrating its capability to balance detection accuracy with computational efficiency, making it a

widely accepted choice [10]. However, YOLOv9 and YOLOv10 which have been recently released in the market have not been applied yet to the field of pest detection. Therefore, this study involves training and evaluating these versions including YOLOv8, YOLOv9, and YOLOv10, to determine which model is most effective for pest detection in agricultural settings.

The proposed implementation of a pest detection system based on YOLO is a step into a new era in agriculture. This paper aims to develop a pest detection system using an optimal lightweight YOLO framework integrated with a smartphone for real-time detection of multiple pest classes and coupled with a web-based monitoring system. Farmers can easily monitor their crops by positioning their smartphones in various areas of their fields, with data being sent to a server for analysis. The web-based system then allows farmers to observe real-time results and receive instant recommendations for effective pest mitigation.

The rest of the paper is arranged as follows. Section II reviews related works in pest detection and classification. Section III covers the materials and methods used, including the methodology, pest image datasets, preprocessing techniques, YOLOv8, YOLOv9, and YOLOv10 model implementations, and the relevant evaluation metrics. Section IV describes the experimental setup and presents the results and discussion. The deployment of the optimal model in real-time applications for pest detection is discussed in Section V. Section VI outlines the limitations encountered during the study. Section VII concludes the paper and suggests enhancements for future work.

II. SIMILAR WORK

Recent advancements in deep learning and mobile technologies have significantly influenced the development of real-time image-based pest detection systems in agriculture [11]. Many studies have explored different CNN architectures for mobile devices. However, these systems often encounter limitations including restricted pest class detection, hardware constraints, and challenges in achieving real-time performance [12] [13]. Various researchers have investigated object detection techniques for identifying insect pests, each contributing unique approaches and highlighted persistent challenges.

For instance, Fuentes et al. integrated SSD, R-CNN, and Faster R-CNN deep learning models with a VGG network and residual networks to recognize nine different types of tomato plant pests and diseases [14]. Although this approach achieved a mean Average Precision (mAP@0.5) value of 83.06%, it was confined to a limited number of pest classes, which limited its application to more diversified agricultural settings. Lin et al. employed Fast R-CNN to develop an anchor-free regional convolutional network using an end-to-end model approach [15]. The model is able to categorize 24 pest classes and achieved a mAP@0.5 value of 56.4% and a recall of 85.1%. These results surpassed the performance of traditional Fast R-CNN in controlled environments. However, the method's applicability in real-time, dynamic agricultural environments remains uncertain. Another study conducted by Sabanci et al. constructed a convolutional recurrent hybrid network to identify wheat grain that has been affected by pests [16]. They combined AlexNet with bidirectional short-term memory (BiLSTM). The model achieved a remarkable cumulative accuracy of 99.50%. While this demonstrates high precision for specific tasks, the system's scope was limited to wheat grain since it can only distinguish between two types such as healthy wheat grains and those damaged by sunn pests (SPD). Also, it did not address broader pest detection needs across various crops.

In another effort, Koklu et al. devised a deep feature extraction method based on CNN-SVM [17]. The researchers classified five distinct species of grapevine leaves and achieved an accuracy of 97.60%. Although effective for leaf classification, this method has yet to be tested in the more complex domain of insect pest detection. Another notable example is that Li et al. created a real-time system for identifying pests and plant disease through the implementation of Faster R-CNN [18]. Their approach effectively detected unseen rice diseases in video footage but focused primarily on disease rather than insect pest detection.

Additionally, a visual flying insect detection system based on the YOLO architecture was introduced by Zhong et al. using a Raspberry Pi [19]. A cumulative accuracy of 92.50% and a classification accuracy of 90.18% are both achieved by the system. Although this system showed promise in identifying flying insects, the limited processing power of Raspberry Pi constrained its application in more computationally intensive tasks. In addition, Arunabha M. Roy and Jayabrata Bhaduri introduced an enhanced version of YOLOv4, called Dense-YOLOv4, by incorporating DenseNet into its backbone to improve the feature transfer and reuse [20]. This model achieved an impressive mAP@0.5 value of 96.20% in identifying various phases of mango growth within a complicated orchard setting. However, its heavy computational demands pose challenges for deployment on mobile devices. Although it achieved an impressive recognition rate of 99.3% within an average processing time of 44 milliseconds, but these models were only specialized for a single type of crop, which limits their broader applicability.

In recent years, the YOLO algorithm family has evolved significantly to enhance real time object detection for lightweight and mobile friendly applications. For example, a YOLOv5-S model was developed by Thanh-Nghi Doan for real-time insect detection and was integrated into resource-constrained mobile devices [21]. The model performance was reported up to 70.5% classification accuracy on the Insect10 dataset and 42.9% on the IP102 larger dataset. These results prove that the model performs reasonably well on smaller dataset but struggles to achieve the accuracy required for effective agricultural pest detection as the size and complexity of the pest dataset increases.

Moreover, an updated version, YOLOv8, is widely adopted due to its faster and more accurate performance in real-time object detection tasks. Additionally, its architecture allows for easy refinement and customization to adapt to specific tasks. For example, Yin Jian Jun enhanced the YOLOv8 model by refining its feature extraction algorithm and reducing the number of parameters count to a achieve lightweight model design [22]. Through the refined training techniques, the model

achieved a remarkable mAP@0.5 value of 97.3%. for detecting eight different species of rice pest. Although YOLOv8 has now been widely applied in many studies, it was usually applied to detect only a few pest species, hence making its application disadvantageous in more complicated agricultural settings.

Recently, the latest works in the YOLO series, such as YOLOv9 and YOLOv10, have come up to introduce more features that bring greater performance at lesser computational overhead. YOLOv9 is upgraded with enhancements like Generalized Efficient Layer Aggregation Network (GELAN) and Programmable Gradient Information (PGI), which are at the root of improved detection performance [23]. On the other hand, YOLOv10 has been developed by the Tsinghua University researchers with the aid of the Ultralytics Python package, introducing an innovative technique to real-time detection by solving post-processing issues and model architecture shortcomings that were present in previous YOLO versions [24]. By omitting non-maximum suppression (NMS) and refining several aspects of the model architecture, YOLOv10 achieves cutting edge results at a substantially lower computational cost. However, despite these advancements, the practical application of both YOLOv9 and YOLOv10 in detecting agricultural pests is yet to be applied.

Although these studies have made remarkable progress, numerous challenges are still unsolved. First, the majority of these current models can only detect a few numbers of pest or object classes. This limits the application of these methods in different agricultural scenarios. Also, since their datasets are small with fewer classes, their data preprocessing techniques such as data augmentation and clean processes are often simpler and overlooked. This deficiency can hinder accurate detection as the size and variety of classes increase. Third, the real-time deployment of these models in a mobile device is generally hindered by hardware limitations. In many cases, the real-time detection capabilities are frequently restricted by the computational demands of deep learning models [25]. Therefore, there is a great need for a lightweight model in terms of overcoming these weaknesses. Although some have been developed and integrated with smartphones, such systems are not capable of providing real-time updates and comprehensive pest detection across multiple classes. Additionally, they also lack a web-based platform responsible for pest data monitoring and analysis.

To address the limitations identified in existing studies, this paper aims to develop a pest detection system using a lightweight and optimized YOLO deep learning framework, integrated into mobile devices and complemented with a web-based monitoring system.

The main contributions of this paper are as outlined below:

- Employing advanced data augmentation techniques, including hue adjustment, horizontal flipping, and scaling, to significantly increase dataset diversity and enhance the model's ability to classify pests in complex agricultural conditions.
- Introducing a lightweight YOLO-based deep learning model that is optimized to detect a broader range of pest classes compared to previous studies, balancing accuracy and efficiency for real-time smartphone applications in diverse agricultural settings.
- Integrating the optimized model with smartphone technology for real time detection of multiple pest classes, making advanced pest detection tools more accessible and user-friendly.
- Creating an interactive web-based monitoring platform that offers dynamic real-time updates of pest detection and provides sustainable recommendations for effective pest management strategies.

III. Materials and Methods

A. Methodology

The proposed methodology follows a structured approach to develop an effective pest detection model for agricultural applications as shown in Fig. 1. At the beginning, the researchers collected multiple pest image datasets and merged them to form a unified dataset through several preprocessing processes such as data standardization. Data standardization is a technique to ensure that datasets are in a consistent and standardized format for configuring the YOLO format. Secondly, the researchers preprocessed the dataset by employing data augmentation techniques including hue adjustments, image translation, horizontal flipping and scaling to expand the training dataset for enhancing its variability. This step aims to prepare a dataset that can be used to train a robust detection model capable of generalizing across diverse agricultural scenarios. Following this, the images were annotated with bounding boxes to label pest instances accurately. After that, the researchers split the dataset by allocating 80% of the dataset for the model training and 20% for the model validation. Researchers then proceeded to train various YOLO object detection models with the pest training dataset. The trained models were fine-tuned to optimize their performance across different YOLO versions. Then, researchers validated and evaluated the performance of each base and fine-tuned model by using the shared validation dataset. During validation, new pest images that were not previously used for training or fine-tuning were introduced to assess the models' effectiveness with unseen data. The researchers also integrated each model into the smartphone application for measuring their real-time performance to assess its capabilities in a practical setting. By comparing the results, the optimal model was identified for practical field adaptation. This model was then deployed within a smartphone application to enable real-time pest detection in agricultural environments. Finally, a web-based monitoring system was developed to provide dynamic and real-time updates on pest detection. This allows users like farmers to monitor captured pest data and receive actionable insights for effective pest management.

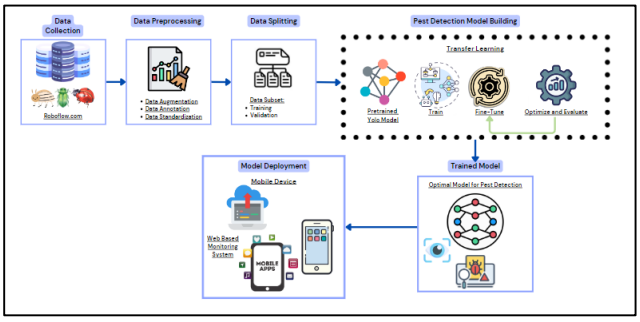

Fig. 1. Proposed methodology to develop the pest detection system.

B. Data Collection

To train and evaluate the pest detection model, the researchers utilized two publicly available datasets from Roboflow platform as a source in the experimental study: YOLOIP1 dataset and Pest_Dataset_3 [26] [27]. Since both datasets are different in terms of the structures and class distribution, the datasets were subjected to a data standardization process to ensure uniformity. Both datasets were reorganized to have consistent directory structures, file naming conventions, and annotation formats. After standardization, the researchers removed the redundant classes to avoid confusion due to the similar morphological features. For example, classes like "white margined moth" and "margined moth" which differed only slightly in color were removed. The researchers then merged the datasets manually by selecting the most relevant pest classes that are frequently encountered in agricultural scenarios for accurate detection. The final merged dataset comprised 29 pest classes. Each class contains a significant number of images as detailed in Table I. However, the merged dataset contained fewer images than the expected threshold of a total of 10k images. Therefore, data augmentation techniques were employed in the next step to expand the dataset's size.

TABLE I. FINAL CLASS DISTRIBUTION OF THE MERGED DATASET

Index	Pest Class	Total Count
0	Alfafa plant bug	99
1	Ampelophaga	126
2	Aphids	886
3	Beet spot flies	79
4	Flea beetle	414
5	Grain spreader thrips	118
6	Grub worm	522
7	Icerya-purchasi-Maskell	73
8	Limacodidae	95
9	Lycorma delicatula	111
10	Lygus	318
11	Mole cricket	888
12	Oides decempunctata	77
13	Paddy stem maggot	94
14	Peach moth	223
15	Pieries canidia	147
16	Plant hopper	856
17	Protaetia	180
18	Red spider	245
19	Rice gall midge	244
20	Rice leaf caterpillar	235
21	Rice leaf roller	233
22	Rice leafhopper	242
23	Rice shell pest	115
24	Rice stemfly	108
25	Weevil	650
26	Wireworm	464
27	Xylotrechus	68
28	Yellow rice borer	591
Total number of images		8,501

C. Data Augmentation

To improve the generalization capabilities of models, several data augmentation techniques were applied to the dataset. Firstly, the researchers adjusted the hue, saturation and value (HSV) of images to randomly alter the color intensity and brightness in a manner to mimic different environmental conditions. Secondly, the researchers shifted the images along the x and y axes through image translation. This step is to provide different angles of the same image. They also applied horizontal flipping to mirror the images for creating variations in how objects are oriented. Finally, they resized images while keeping their aspect ratios through image scaling process to ensure the model remains robust and can handle differences in object size effectively. Fig. 2 illustrates examples of the augmented images.

After completing the data augmentation process, the final dataset was prepared with sufficient images and met the expected requirements to have at least 10,000 images for training the robust model. The details of the dataset after augmentation are presented in Table II.

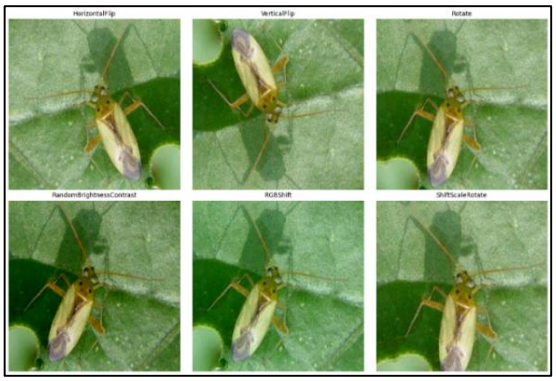

Fig. 2. Samples of augmented pest images.

TABLE II. DETAILS OF DATASET

Dataset Aspect	Number
Number of classes	29
Number of images	14,147
Number of instances (objects labeled within the images)	17,856

D. Data Annotation

After data augmentation, the researchers carefully reviewed the images to ensure that each one was annotated and assigned with a pest class ID correctly. A deep learning model such as YOLO model learn features from the labeled images [28]. Therefore, the correctness of feature labeling will greatly influence the performance of training model especially considering the similarities between many pest species. If any discrepancies were found such as incorrect class labels or annotations, the researchers made corrections to those particular images using the Roboflow platform. The annotation process involved normalizing the coordinates between 0 and 1 to accommodate varying image sizes. After this process, the annotations across the entire dataset were consistent and accurate. Each annotation was recorded in a text file containing the following details for each bounding box as shown in Eq. (1). A sample of a pest image with the correct annotation bounding box is shown in Fig. 3.

$$(id_{class}, x_{centre}, y_{centre}, width, height) \quad (1)$$

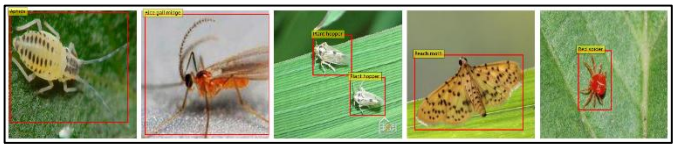

Fig. 3. Visualization of labeled images.

E. Data Splitting

Before training the YOLO model, the researchers split the dataset into training and validation sets with an 8:2 ratio. This step is to ensure that the model is trained on a diverse and representative set of images while also being evaluated on a separate validation set. The summary of final dataset aspects is shown in Table III. It is well prepared for the next stages of model training and evaluation.

TABLE III. SUMMARY OF FINAL DATASET ASPECTS

Dataset Aspect	Details
Number of classes	29
Image Size	640x640 pixels
Training Images and Label Files	10,097
Validation Images and Label Files	4,050
Total Instances in Training	12,658
Total Instances in Validation	5,198

F. Overview of YOLO Model

The YOLO algorithm, which stands for "You Only Look Once" was first introduced by Joesph Redmon et al. in 2015 [29]. It is renowned for its efficiency in object detection since it can identify objects and their positions by examining the entire image only once. As seen in Fig. 4 below, the YOLO algorithm divides the image into N grids, each of which contains an equal-sized S × S region. Each grid is responsible for detecting and locating the object within it.

Unlike the conventional methods that employ two-stage object detectors, YOLO treats object detection as a regression problem [30]. It predicts bounding boxes around the objects and their corresponding class probabilities straight from the feature maps in a single pass. This means that YOLO can recognize objects in an image faster than the two-stage detector. Therefore, the YOLO algorithm is considered as a one-stage detector that prioritizes speed and is thus ideal for real-time object detection.

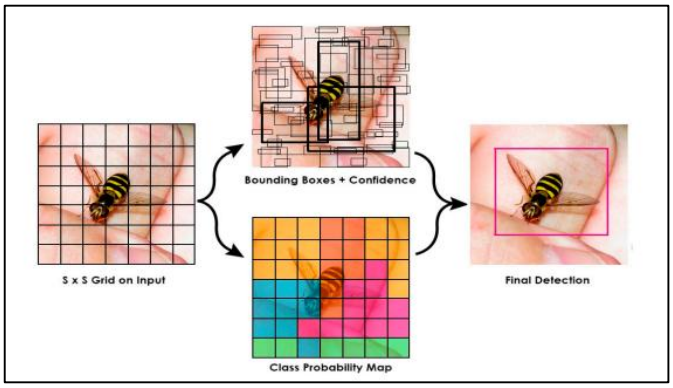

Fig. 4. Illustration of the YOLO framework for object detection [31].

1) YOLOv8: YOLOv8 is the most popular and widely used model in the whole You Only Look Once (YOLO) series. It signifies a notable development in the domain of object detection with its remarkable enhancements in both speed and accuracy through extensive architectural optimization and innovation. The architecture of YOLOv8 is depicted in Fig. 5. The YOLOv8 model considers the multi-scale attributes of objects by using three detection layers at different scales to handle objects that come in different sizes [32]. This method allows the model to efficiently manage objects of varying sizes and proportions.

In its architectural design, there are three main components including Backbone, Neck and Head [33]. The Backbone is used for extracting multi-scale features to ensure the model can perceive inputs across different scales. It includes modules such as Conv, C2f, and SPPF (Spatial Pyramid Pooling-Fast). In the Neck section, YOLOv8 combines features without enforcing standardized channel dimensions by integrating a path aggregation network and a feature pyramid network [34] [35]. This method reduces both the number of parameters and the overall tensor size. To simplify anchor box operations and prevent displacement issues, YOLOv8 adopts a decoupled head method to separate the detection and classification head [36]. The Head component is responsible for tasks such as bounding box regression, target classification and confidence assessment in the prediction layers. It ultimately delivers precise detection results through the use of non-maximum suppression.

To support diverse computational requirements, YOLOv8 is available in five versions as detailed in Table IV [37]. Each model comes with different parameter counts and resource consumption due to differences in width and depth parameters. The higher the scale of the model is, the higher the detection performance is as the parameter count and resource consumption rise. The MS COCO dataset is frequently used for benchmarking object detection models. By comparing the performance between models on the MS COCO dataset, the researchers selected YOLOv8n as their baseline model for this study because of its relatively low parameter count and resource efficiency. FLOPs (Floating Point Operations per Second) are listed in Table IV to indicate the computational complexity of each model.

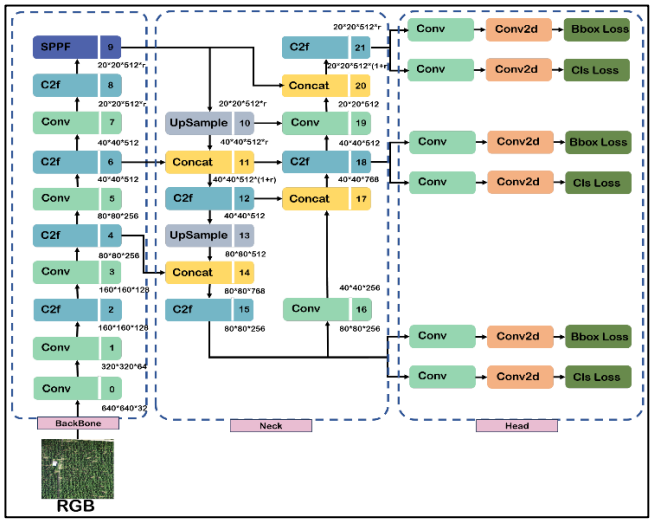

Fig. 5. Model architecture of YOLOv8 [38].

TABLE IV. YOLOv8 VARIANT COMPARISON

Model	Params (M)	FLOPs (B)
YOLOv8n	3.2	8.7
YOLOv8s	11.2	28.6
YOLOv8m	25.9	78.9
YOLOv8l	43.7	165.2
YOLOv8x	68.2	257.8

2) YOLOv9: YOLOv9 is the most recent addition to the YOLO versions in 2024. It introduces two major innovations such as the Programmable Gradient Information (PGI) and the Generalized Efficient Layer Aggregation Network (GELAN). The PGI framework tackles the challenges of information bottlenecks in deep neural networks and makes supervision mechanisms compatible with lightweight architecture [39]. By leveraging PGI, substantial accuracy improvements can be achieved in both complex and lightweight architectures. This is due to the fact that PGI mandates reliable gradient information during training, which in turn enhances the architecture's ability to learn and make predictions.

The GELAN architecture was designed with a purpose of improving the performance of objection detection tasks by lightweight and high efficiency footprint [40]. Because of its strong performance across different depth configurations and computational blocks, it is well-suited for deployment on various inference devices including devices that are resource constrained. In simple words, PGI and GELAN solved the issues related to computational efficiency and information loss. By integrating both GELAN and PGI frameworks, YOLOv9 represents a substantial advancement in lightweight object detection.

Table V presents five YOLOv9 variants along with their parameter count and FLOPs as assessed on the MS COCO Dataset [41]. YOLOv9t was chosen as another baseline model for this study due to its lowest parameter count and computational demand.

TABLE V. YOLOv9 VARIANT COMPARISON

Model	Params (M)	FLOPs (B)
YOLOv9t	2.0	7.7
YOLOv9s	7.2	26.7
YOLOv9m	20.1	76.8
YOLOv9c	25.5	102.8
YOLOv9e	58.1	192.5

3) YOLOv10: In 2024, another groundbreaking development emerged with the introduction of YOLOv10. This version further expands the boundaries of real-time object detection by resolving its inherent difficulties. It sets a new benchmark by completely eliminating non-maximum suppression (NMS) during post-processing, thereby enhancing inference speed [42]. The model introduces a novel dual-label assignment system to maintain an optimal balance between accuracy and speed. By integrating one-to-one and one-to-many label assignments, YOLOv10 benefits from rich supervisory signals during its training, ensuring computational efficiency while acquiring important detection features without any post-preprocessing NMS [43].

Furthermore, the enhancements in YOLOv10's architecture involve the implementation of spatial-channel decoupled down sampling, lightweight classification heads, and rank-guided block design [42]. Each of these additions lowers the computational requirements and the number of parameters. These enhancements enhance both the efficiency and scalability of the model over a wide range of devices from powerful servers to edge devices with limited processing power and storage capacity. The performance of both the lightweight and heavyweight versions of YOLOv10 model on the COCO dataset is compared in Table VI [44]. YOLOv10-N with the least parameters and computational resource is selected for this study as the baseline model for YOLOv10.

TABLE VI. YOLOv10 VARIANT COMPARISON

Model	Params (M)	FLOPs (G)
YOLOv10-N	2.3	6.7
YOLOv10-S	7.2	21.6
YOLOv10-M	15.4	59.1
YOLOv10-B	19.1	92.0
YOLOv10-L	24.4	120.3
YOLOv10-X	29.5	160.4

G. Evaluation Metrics

In object recognition methods, the detection result and classifier performance are the two key indices used to evaluate models [45]. Mean Average Precision (mAP) is a widely used metric for evaluating the overall performance of object detection systems [46]. The mAP score is derived by comparing the predicted bounding boxes with the ground truth boxes. Higher mAP score indicates more accurate model detections. This metric is calculated using several sub-metrics: Precision, Recall, Intersection over Union (IoU), and data from the Confusion Matrix.

To evaluate the detection result, Intersection over Union (IoU) metric is employed. IoU measures the ratio of the intersection to the union of predicted and ground truth bounding boxes [47]. It indicates how closely the predicted bounding box matches the ground truth. When the IoU value exceeds a threshold of 0.5, the detection is regarded as a True Positive (TP). Conversely, if the IoU value falls below 0.5, the detection is considered a False Positive (FP). A False Negative (FN) occurs when the model fails to detect an object that exists in the ground truth. A True Negative (TN) refers to the model correctly identifying that no object exists in a region where there is indeed no object. The concept of IoU is illustrated in Fig. 6, where the rectangles R1 and R2 serve as bounding frames for the object's ground truth and prediction.

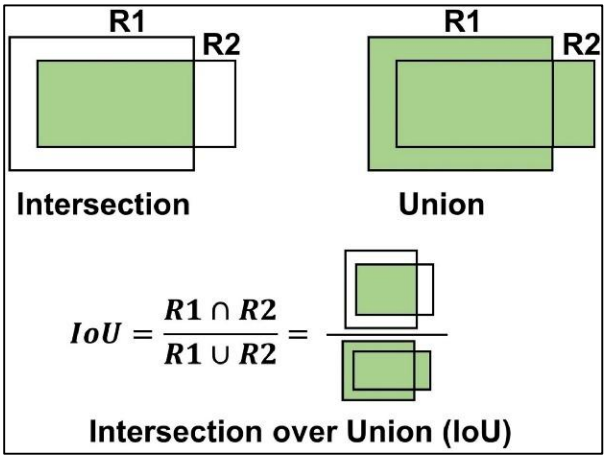

Fig. 6. Concept of Intersection over Union (IoU).

For accessing the performance of the classifier, a Confusion Matrix is used. It categorizes the model's predictions into four attributes [48]:

- True Positives (TP): The model correctly identifies and matches a label with the ground truth data.
- True Negatives (TN): The model correctly identifies that a label is absent when it is indeed not present in the ground truth data.
- False Positives (FP): The model incorrectly predicts a label that is not actually present in the ground truth data.
- False Negatives (FN): The model fails to detect a label that is present in the ground truth data.

Using the IoU value along with the Confusion Matrix outputs (TP, TN, FP, and FN), key performance metrics such as Precision, Recall, and mAP are calculated. All these metrics are used to assess the overall performance of object detection models by calculating them using Formulas (1) to (3).

$$Precision = \frac{TP}{TP+FP} \quad (1)$$

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k \quad (3)$$

AP_k = the average precision of class k, and n = number of classes

IV. RESULT AND DISCUSSIONS

A. Experimental Setup and Training

In this study, the training and evaluation of the selected YOLO models like YOLOv8n, YOLOv9t and YOLOv10-N were conducted on Google Colab using Nvidia L4 GPU with 24 GB of RAM. The dataset which had already been preprocessed and split into 80:20 ration in previous phase was used to ensure consistent and high-quality input for model training. Each YOLO model was trained on the training set which consists of 10,097 pest images while their performances were evaluated on the validation set containing 4,050 images.

To ensure a fair comparison, all YOLO models were trained using the same hyperparameters. The model was created using Pytorch and runs for 150 epochs to train with a batch size of 16, learning rates of 0.001 and default dropout rates provided by the YOLO framework. Adam optimizer which is known as "Adaptive Moment Estimation" was used as the optimization algorithm. After training the model, a fine-tuning process was conducted using a grid search method to optimize each model. Grid search method tuned the model by testing out all combinations of critical parameters [49]. This fine-tuning process ran for 10 additional epochs to refine the models' performance.

During training, the YOLO models will automatically perform in-training evaluations at the end of each epoch using the validation set. The metrics reflect the ongoing learning process and may result in slightly higher performance or overfitting due to continuous weight adjustments [50]. Once training is completed, the reseachers conducted a post-training evaluation to access the model's generalization ability by using the same validation set in a static environment with fixed weights.

The experimental configuration for training and testing these models is detailed in Table VII.

TABLE VII. EXPERIMENTAL CONFIGURATION

Configuration	Params (M)
Platform	Google Colab
CPU	2.0 Intel Xeon (R) CPU @ 2.20GHz × 12
GPU	Nvidia L4 (24 GB RAM)
Accelerated Environment	Nvidia L4 CUDA 12.2
Operating System	Ubuntu 22.04.3 LTS
CUDA Version	12.2
PyTorch GPU Availability	True
PyTorch CUDA Version	12.4

B. Experimental Results

1) Training loss: Training loss is the first focus of this analysis. It encompasses multiple components including classification loss (cls_loss), distribution focal loss (dfl_loss) and bounding box loss (box_loss). Each component of the training loss tracks the degree of error in the model's outputs. It provides insights into how well the models fit the data and aids in determining the best weights [51]. As shown in Fig. 7 to Fig. 9, all training loss curves demonstrate a clear downward trend throughout the training process. This means that the proposed models excel at learning early and detecting pests during the training time. As the networks undergo further epochs, the classification loss declines slowly. The models converged after 140 epochs. This enabled researchers to conclude that 150 epochs was a good parameter for building the model.

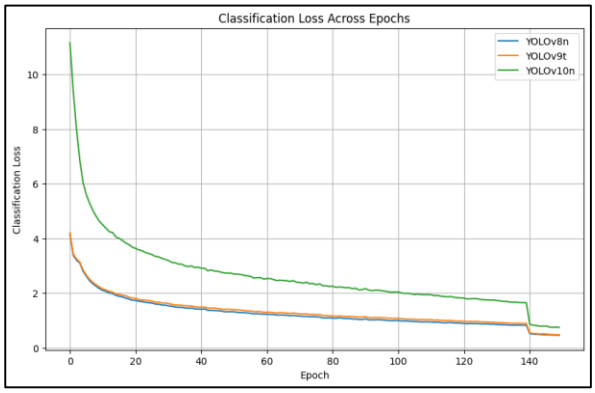

Fig. 7. Classification loss graph.

As can be seen in Fig. 7 to Fig. 9, YOLOv8n and YOLOv9t architecture have achieved similar and lower training losses when compared to the YOLOv10-N. This observation could be attributed to the increased complexity of YOLOv10-N architecture. YOLOv10-N learns more intricate patterns from the data to handle the object detection tasks. However, this statement does not imply that other models are inadequate. Further analysis is required to determine a balanced approach where the model performs well on both training data and real-world data.

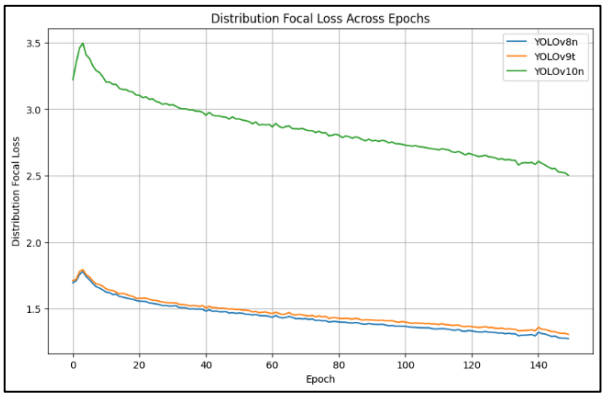

Fig. 8. Distribution focal loss graph.

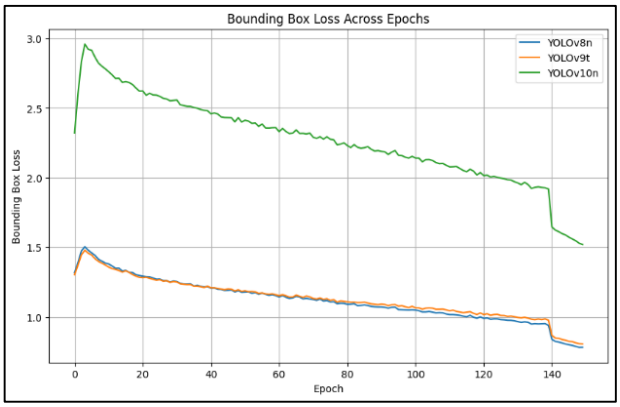

Fig. 9. Bounding box loss graph.

2) Performance evaluation: To further evaluate the models' performance, the researchers compared the baseline and fine-tuned versions of YOLOv8n, YOLOv9t, and YOLOv10-N using the statistical indicators which includes precision, recall, mAP@ 0.5 and mAP@ 0.5:0.95. The results of models are presented in Table VIII and illustrated in Fig. 10. All fine-tuned models demonstrate improvements across all the key metrics. As depicted by the graph below, the YOLOv8n model does not top in any of these metrics but shows improvement over its baseline. In fact, it has actually achieved quite good results when not compared with other versions. Moving to the YOLOv9t model, the fine-tuned version attained the highest precision of 94.2% and the best value for recall of 91.4%. This points to a high effectiveness of the YOLOv9t model in doing accurate pest identifications. Meanwhile, the fined-tuned version of YOLOv10-N model achieved the highest mAP@0.5 at 96.7% and mAP@0.5:0.95 at 77.1%. This means that the YOLOv10-N model performs exceptionally well in detecting pests across various IoU thresholds. It is particularly robust for handling complex detection tasks. From this point onward, all references to the YOLO models will pertain to their fine-tuned versions.

TABLE VIII. RESULTS OF YOLOV8N, YOLOV9T AND YOLOV10-N ON VALIDATION DATASET DURING TRAINING

Models	Precision	Recall	mAP^{val}@0.5	mAP^{val}@0.5:0.95
YOLOv8n Baseline	0.853	0.847	0.881	0.629
YOLOv9t Baseline	0.85	0.823	0.874	0.637
YOLOv10-N Baseline	0.884	0.829	0.884	0.643
YOLOv8n Fine-tuned	0.932	0.891	0.951	0.733
YOLOv9t Fine-tuned	0.942	0.914	0.962	0.751
YOLOv10-N Fine-tuned	0.94	0.93	0.967	0.771

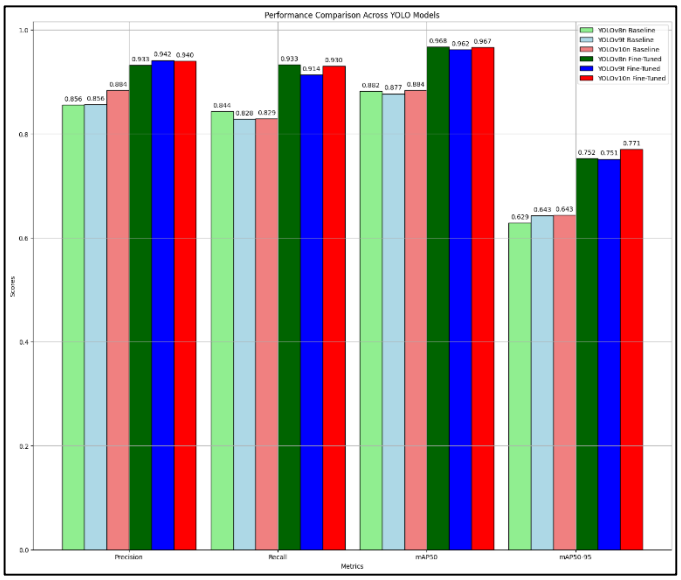

Fig. 10. Comparison of YOLOv8n, YOLOv9t, and YOLOv10-N on validation data during training.

3) *Confusion matrix analysis:* To better demonstrate the results, confusion metrices were employed to evaluate and visualize the performance of the trained YOLO models. A confusion matrix describes the actual and predicted object classification of a classification system [52]. The significance of the prediction results is indicated by the diagonal line in the central of the confusion matrix. The horizontal line shows false negatives, whereas the vertical line shows false positives. The darker the blue in the matrix, the higher the count of correct classifications. Based on the confusion matrices presented in Fig. 11 to Fig. 13, all YOLO models have done quite well with high accuracy values among their predictions.

4) Post-Training evaluation: The previous validation results were all generated automatically by YOLO itself during training. These in-training metrics tend to be higher due to the dynamic adjustments the models make during the learning process such as ongoing weights and gradient updates. Therefore, the researchers reassessed the models using the same validation set containing 4050 images in a post-training static environment where the weights and parameters of each model were fixed. The results are presented in Table IX. It reveals that the YOLOv9t model emerged as the top performer with the highest scores in Precision at 87.4%, Recall at 84.4%, mAP@0.5 at 89.8%, and mAP@0.5:0.95 at 66.7%. It was followed closely by the YOLOv10-N and YOLOv8n model. These findings indicate that although models perform excellent during training and validation, but their performance changes when they are exposed to new data. The causes for this change in performance might be due to slight overfitting during training or the complexities of the real-world image environments. Sample predictions made by each model on test data are presented in Fig. 14, Fig. 15, and Fig. 16.

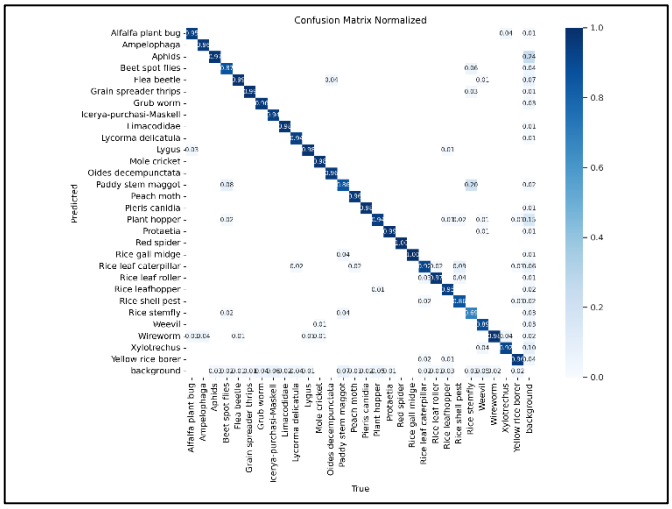

Fig. 11. Confusion matrix of YOLOv8n.

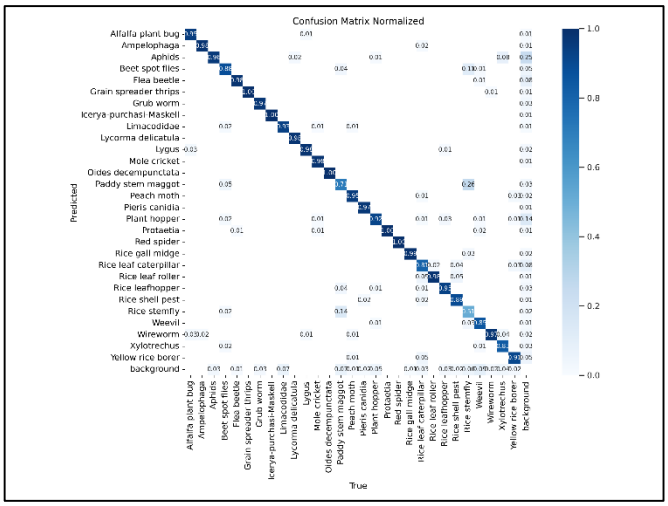

Fig. 12. Confusion matrix of YOLOv9t.

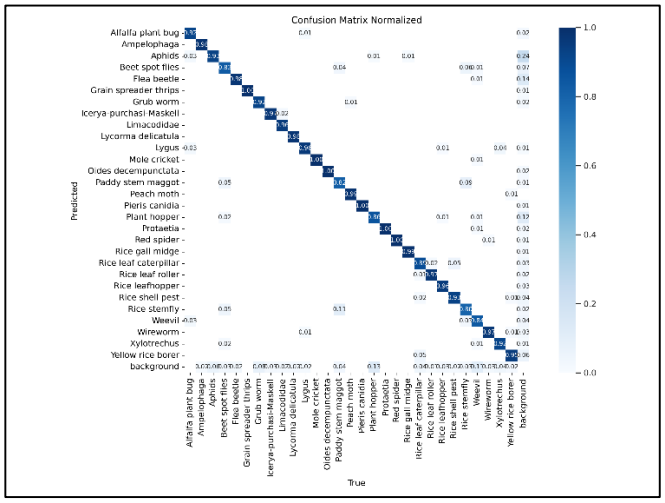

Fig. 13. Confusion matrix of YOLOv10-N.

TABLE IX. RESULTS OF YOLOV8N, YOLOV9T AND YOLOV10-N ON VALIDATION DATASET IN POST-TRAINING PHASE

Models	Precision	Recall	mAP^{val}@0.5	mAP^{val}@0.5:0.95
YOLOv8n	0.859	0.827	0.871	0.618
YOLOv9t	0.874	0.844	0.898	0.667
YOLOv10-N	0.873	0.837	0.883	0.639

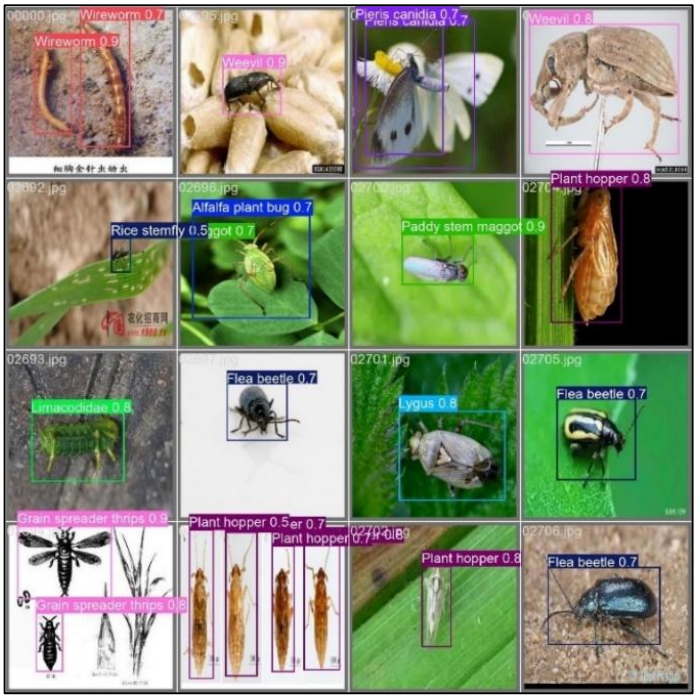

Fig. 14. Sample predictions made byYOLOv8n.

5) *Real-world application testing:* Strong performance of model during training and validation does not always guarantee equal success in real-world situations [53]. To further evaluate the YOLO models, they were converted to TensorFlow Lite format and integrated into a smartphone application. The testing was performed on a smartphone configured as detailed in Table X. This setup was used to evaluate the average inference time of each model by running them 10 times on a set of printed pest photos placed in a plant environment. The purpose of this test was to assess the models' practicality for real-time pest detection in agricultural fields.

Fig. 15. Sample predictions made byYOLOv9t.

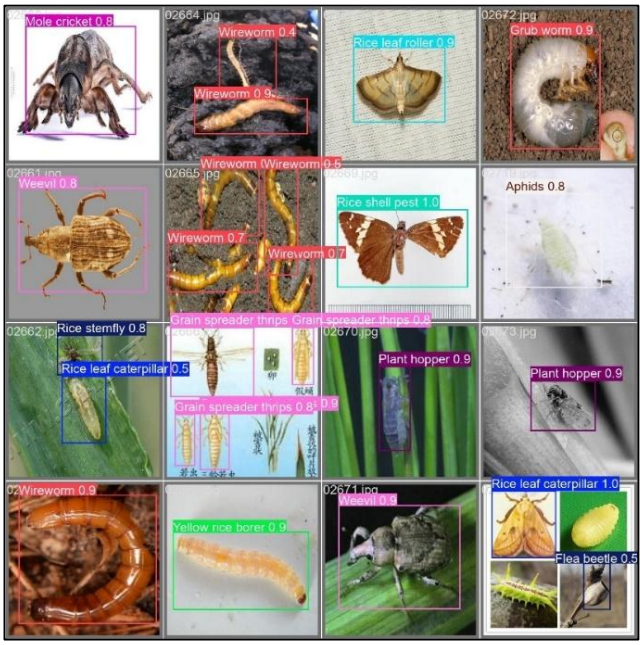

Fig. 16. Sample predictions made byYOLOv10-N.

As shown in the Table XI, lighter models like YOLOv9t recorded the fastest average inference time at 250.6ms, followed by YOLOv8n at 320.5ms and YOLOv10-N at

550.7ms. These results provide strong evidence that YOLOv9t is the optimal model to deploy on resource-constrained devices for achieving real-time detection in complex scenes.

TABLE X. ANDROID DEVICE SPECIFICATIONS

Component	Specification
Operating System	Android
Model	Xiaomi 11T Pro
RAM	16GB
Internal Memory	129GB
Chipset	Snapdragon 888 5G (5 nm)
CPU	Octa-core (1x2.84 GHz Cortex-X1 & 3x2.42 GHz Cortex-A78 & 4x1.80 GHz Cortex-A55)
GPU	Adreno 660

TABLE XI. AVERAGE INFERENCE TIME FOR EACH MODEL PERFORMED IN ANDROID APP (CALCULATED OVER 10 RUNS)

Model	Average Inference Time (ms)
YOLOv8n	320.5
YOLOv9t	250.6
YOLOv10-N	550.7

C. Discussion

A summary of the overall results can be seen in Table XII. The YOLOv9t model outperformed the other proposed models by achieving the highest values in key metrics such as accuracy and speed. Among the models tested, YOLOv9t provided the best balance between accuracy and speed. Therefore, it can accurately detect various tiny pests in real-time within challenging agricultural environments.

When compared the outcome of this study with recent studies as shown in Table XIII, it is evident that those models classified fewer object classes which ranged from 1 to 24 classes. While several studies achieved over 90% accuracy, their focus was often limited to specific object categories such as detecting leaves or identifying the presence of certain pests. Although the proposed model does not achieve the highest accuracy, it still obtains a remarkable result with 89% accuracy. It is closely comparable to the best-performing models while handling a larger and more diverse range of pest species.

The comparison also highlights the advantages of YOLOv9t over previous methods. For instance, Fuentes et al. and Lin et al. used SSD and Fast R-CNN, but these models struggled to detect a wide variety of pest classes. As a result, they achieved lower mAP scores and were less effective in diverse agricultural environments. Similarly, Sabanci et al.'s model demonstrated impressive precision. However, it was limited to binary classification for specific crops. This limitation restricted its overall versatility. Koklu et al. and Zhong et al. also made progress in tasks like leaf classification and insect recognition, but their models faced challenges due to a smaller number of pest classes and hardware limitations.

In contrast, the proposed YOLOv9t model stands out by successfully detecting 29 different pest classes with an accuracy of 89.8% and an inference time of just 250.6ms. Its lightweight architecture and scalability make it well-suited for real-time applications across a range of agricultural settings. Furthermore, the conversion of the YOLOv9t model to TensorFlow Lite (TFLite) and its integration into a mobile application significantly enhance its deployment capabilities on resource-constrained devices. The TFLite model allows for faster and more efficient inference, enabling real-time pest detection directly on smartphones. This integration ensures that pest management can be carried out in the field, improving efficiency for farmers and agricultural workers. This model's ability to overcome the limitations of earlier approaches highlights its practical advantages for pest management in real-world agricultural environment.

TABLE XII. OVERALL PERFORMANCE COMPARISON OF YOLOV8N, YOLOV9T AND YOLOV10-N USED IN THE EXPERIMENTAL STUDY

Models	Precision	Recall	mAP^{val}@0.5	mAP^{val}@0.5:0.95	Average Inference Time (ms)
YOLOv8n	0.859	0.827	0.871	0.618	320.5
YOLOv9t	0.874	0.844	0.898	0.667	250.6
YOLOv10-N	0.873	0.837	0.883	0.639	550.7

TABLE XIII. COMPARISON OF PEST DETECTION MODEL WITH PREVIOUS STUDIES

Author & Reference	Model	Pest Classes	Accuracy (%)	Key Insights
Fuentes et al. [14]	SSD, R-CNN, Faster R-CNN	9	83.06	Limited pest classes; lacks real-time performance.
Lin et al. [15]	Fast R-CNN	24	56.4	High recall but poor mAP, unsuitable for real-time dynamic environments.
Sabanci et al. [16]	AlexNet + BiLSTM	2	99.5	Extremely high precision but limited to specific crops and pests.
Koklu et al. [17]	CNN-SVM	5	97.6	Effective for leaf classification; untested for insect pest detection or broader applications.
Zhong et al. [19]	YOLO + Raspberry Pi	1	90.18	High accuracy but lacks scalability for detecting diverse pest classes.
Thanh-Nghi Doan [21]	YOLOv5-S	10; 102	70.5; 42.9	Performs well on small datasets; struggles with complex datasets due to limited scalability.

Author & Reference	Model	Pest Classes	Accuracy (%)	Key Insights
Yin Jian Jun [22]	YOLOv8	8	97.3	High accuracy but lacks scalability for detecting diverse pest classes.
Proposed YOLOv8n	YOLOv8n	29	87.1	Provides balance between computational efficiency and detection performance; lower accuracy than YOLOv9t.
Proposed YOLOv9t	YOLOv9t	29	89.8	Achieves high accuracy and scalability; balances speed and accuracy effectively.
Proposed YOLOv10-N	YOLOv10-N	29	88.3	Higher computational complexity; excels in detecting complex patterns but slower inference speed.

V. APPLICATION

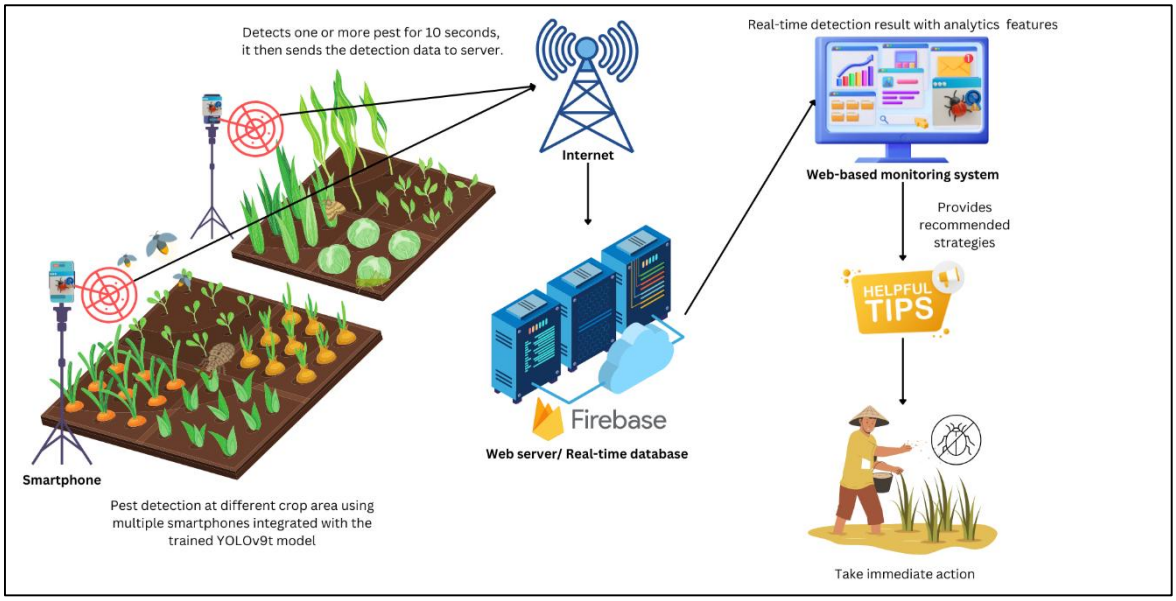

Fig. 17. Overview of the proposed lightweight mobile pest detection system with real-time monitoring.

A. Deployment

The YOLOv9t model was selected for deployment in the smartphone pest detection application due to its strong performance compared to other versions. The application detects pests in real-time but is programmed with a 10-second delay before sending the data to Firebase. This delay ensures that the pest remains on the plant and is not just passing by. Users can specify the crop area where the monitoring is taking place. Once the application detects one or more pests for 10 seconds, it sends the detection result to the Firebase server where it integrates a web-based monitoring system. This setup allows the user to track which particular area of the crop is getting infected. By providing this information, users can evaluate the situation and take immediate action if necessary. An overview of the pest detection system is shown in Fig. 17.

B. Example of Successful Pest Detection by YOLOv9t

Fig. 18 below shows the predictions made by the YOLOv9t model integrated into a smartphone application for detecting pests in real-time. It performs well in correctly identifying various pests in a simulated plant environment.

Fig. 18. Correct real-time predictions of pests using the YOLOv9t model integrated into a smartphone application in a simulated plant environment.

By using the app, users can also upload pest images for detection. They can retrieve detailed information about that uploaded pest, including a description and recommendations for managing the pest. If the pest is not detectable by the model, user can even upload the image to Firebase for experts to further refine the capabilities of model in detecting new pests in future. The web-based monitoring system complements the app by providing real-time detection logs, pest summary details, analytics dashboard, and access to detailed pest information for recommended strategies. Fig. 19 to Fig. 24 illustrate these additional features of the pest detection system.

Fig. 19. Upload pest image feature.

Fig. 20. Retrieved detailed information about the uploaded pest.

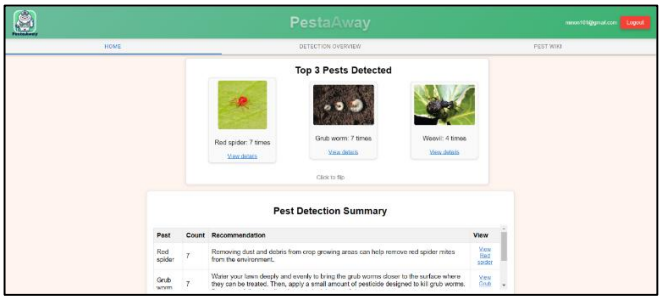

Fig. 21. Web-based monitoring system displaying pest population trends.

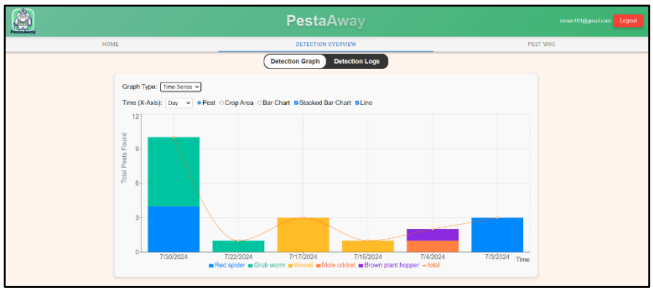

Fig. 22. Web-based monitoring system's pest analytics dashboard.

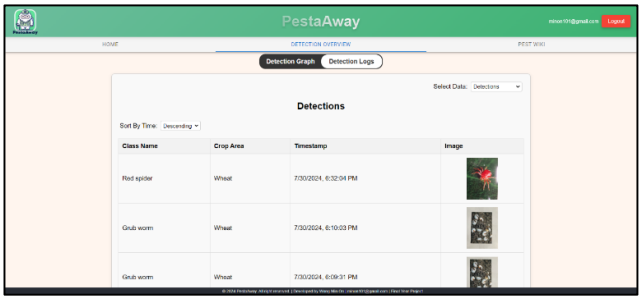

Fig. 23. Pest detection logs in the web-based monitoring system.

Fig. 24. Example of detailed pest information for one particular pest provided by the web-based monitoring system.

VI. Limitations and Potential Improvements

In this study, several limitations were identified that affect the detection performance of the integrated model within the application. One notable issue is when the pests overlap or move rapidly in the real-world condition. This makes the model difficult to detect pests and differentiate between them. As shown in Fig. 25, the integrated model successfully detects the aphids but fails to identify the mole cricket underneath. This results in false negative detection. Another challenge is that the model mistakes a leaf for aphids due to their similar color and shape. This misidentification affects the accuracy of the

detection especially when the environment closely resembles the pests. These limitations underscore the need for further refinement of models to improve detection accuracy in complex environments in future study.

To address these issues, incorporating object tracking and motion filtering techniques could be beneficial. By using these methods, the model could track and identify pests even when they overlap or move rapidly. Expanding the dataset to include more diverse pest images and backgrounds could also help the model generalize better to handle a broader range of real-world conditions.

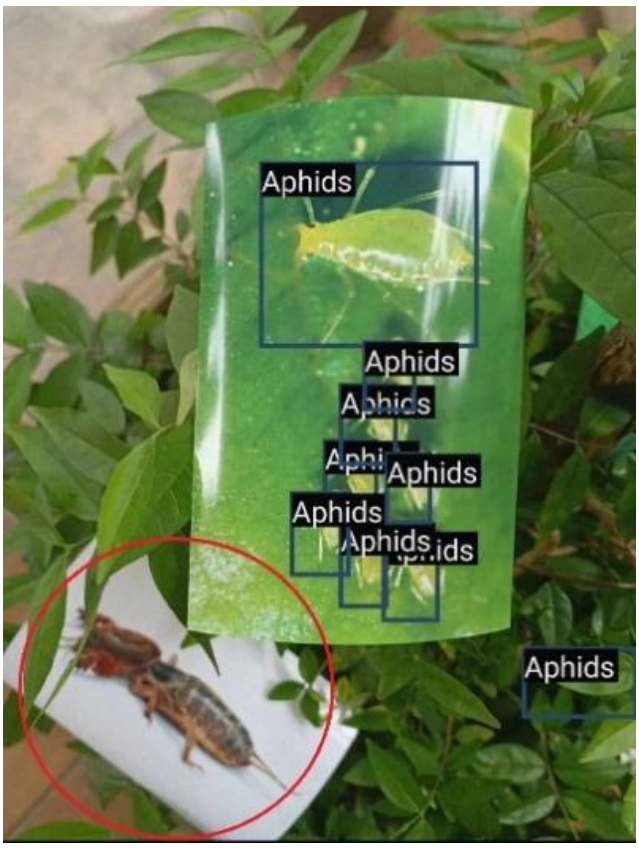

Fig. 25. The integrated YOLOv9t model detecting aphids but missing the mole cricket and misidentifying a leaf as aphids.

VII. CONCLUSION

This paper presented the development of a YOLO-based pest detection system integrated with a smartphone for real-time detection of multiple pest classes along with a dynamic web-based monitoring system for sustainable agriculture. The proposed system employs the YOLOv9t model due to its ability to strike a balance between computational resources, detection accuracy and speed. It has been shown to outperform similar systems by detecting a wider range of pests with fewer computational resources while achieving similar high detection accuracy.

The improved results can be attributed to the lightweight architecture of the proposed YOLOv9t model. It ensures faster inference times for detecting different kinds of pests on mobile devices without compromising accuracy. The numerical results show that the proposed YOLOv9t model achieved a remarkable accuracy with an mAP@0.5 of 89.8%, an mAP@0.5:0.95 of 66.7%, a Precision of 87.4%, a Recall of 84.4%, and an inference time of 250.6 ms. Also, the system enables farmers to monitor their crops by receiving instant updates through the web-based platform in real time. This combination allows for better pest management by providing detailed insights into pest activity across different fields. This approach offers notable advantages in terms of features, computational efficiency, practical implementation, and real-time detection speed.

While some detection inaccuracies might still occur when pests are overlapping or when there are similar backgrounds, shapes, and colors, future improvements will focus on addressing these challenges. The researchers plan to expand the dataset by including a wider variety of pest images taken from more diverse and complex environments. This will help the model better generalize to different real-world scenarios especially those with varying lighting conditions, backgrounds, and angles. Additionally, the researchers also plan to incorporate object tracking and motion filtering techniques to enable the model for tracking the pests more accurately even in situations where they overlap or move rapidly. These enhancements will improve the performance of the pest detection model in complex agricultural environments by ensuring higher detection accuracy and robustness in diverse pest management applications.

REFERENCES

[1] A. Balkrishna, G. Sharma, N. Sharma, P. Kumar, R. Mittal and R. Parveen, "Global perspective of agriculture systems: from ancient times to the modern era," In Sustainable Agriculture for Food Security, pp. 3-45, 2021.

[2] D. Gu, K. Andreev and M. Dupre, "Major trends in population growth around the world," China CDC weekly, vol. 3, no. 28, p. 604, 2021.

[3] FAO, "About FAO's work on plant Production and Protection," [Online]. Available: https://www.fao.org/plant-production-protection/about/en. [Accessed 20 July 2024].

[4] T. Domingues, T. Brandão and J. Ferreira, "Machine learning for detection and prediction of crop diseases and pests: A comprehensive survey," Agriculture, vol. 12, no. 9, p. 1350, 2022.

[5] M. John, I. Bankole, O. Ajayi-Moses, T. Ijila, O. Jeje and P. Lalit, "Relevance of advanced plant disease detection techniques in disease and Pest Management for Ensuring Food Security and Their Implication: A review," American Journal of Plant Sciences, vol. 14, no. 11, pp. 1260-1295, 2023.

[6] P. Mkenda, P. Ndakidemi, P. Stevenson, S. Arnold, I. Darbyshire, S. Belmain, J. Priebe, A. Johnson, J. Tumbo and G. Gurr, "Knowledge gaps among smallholder farmers hinder adoption of conservation biological control," Biocontrol Science and Technology, vol. 30, no. 3, pp. 256-277, 2020.

[7] M. Gulzar, R. Maqsood, H. Abbas, M. Manzoor, M. Suleman, H. Bajwa, A. Hamza, S. Yar, M. Zain, A. Wadood and N. Aslam, "Use of Insecticides and their impact on viral diseases in Humans, Animals and Environment," Hosts and Viruses, vol. 11, pp. 64-77, 2024.

[8] N. Manakitsa, G. Maraslidis, L. Moysis and G. Fragulis, "A review of machine learning and deep learning for object detection, semantic segmentation, and human action recognition in machine and robotic vision," Technologies, vol. 12, no. 2, p. 15, 2024.

[9] A. Nazir and M. Wani, "You only look once-object detection models: a review," in 2023 10th International Conference on Computing for Sustainable Global Development (INDIACom), 2023.

[10] M. Sohan, T. Sai Ram, R. Reddy and C. Venkata, "A review on yolov8 and its advancements," in International Conference on Data Intelligence and Cognitive Informatics, 2024.

[11] C. Ren, D. Kim and D. Jeong, "A survey of deep learning in agriculture: Techniques and their applications.," Journal of Information Processing Systems, vol. 16, no. 5, pp. 1015-1033, 2020.

[12] W. Changji, C. Hongrui, M. Zhenyu, Z. Tian, Y. Ce, S. Hengqiang and C. Hongbing, "Pest-YOLO: A model for large-scale multi-class dense and tiny pest detection and counting," Frontiers in Plant Science, vol. 13, p. 973985, 2022.

[13] Y. Zhang and C. Lv, "TinySegformer: A lightweight visual segmentation model for real-time agricultural pest detection," Computers and Electronics in Agriculture, vol. 218, p. 108740, 2024.

[14] A. Fuentes, S. Yoon, S. Kim and D. Park, "A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition," Sensors, vol. 17, no. 9, p. 2022, 2017.

[15] L. Jiao, S. Dong, S. Zhang, C. Xie and H. Wang, "AF-RCNN: An anchor-free convolutional neural network for multi-categories agricultural pest detection," Computers and Electronics in Agriculture, vol. 174, p. 105522, 2020.

[16] K. Sabanci, M. Aslan, E. Ropelewska, M. Unlersen and A. Durdu, "A novel convolutional-recurrent hybrid network for sunn pest–damaged wheat grain detection," Food analytical methods, vol. 15, no. 6, pp. 1748-1760, 2022.

[17] M. Koklu, M. Unlersen, I. Ozkan, M. Aslan and K. Sabanci, "A CNN-SVM study based on selected deep features for grapevine leaves classification," Measurement, vol. 188, p. 110425, 2022.

[18] L. Dengshan, W. Rujing, X. Chengjun, L. Liu, Z. Jie, L. Rui and W. Fangyuan, "A Recognition Method for Rice Plant Diseases and Pests Video Detection Based on Deep Convolutional Neural Network," Sensors, vol. 20, no. 3, p. 578, 2020.

[19] Y. Zhong, J. Gao, Q. Lei and Y. Zhou., "A vision-based counting and recognition system for flying insects in intelligent agriculture," Sensors, vol. 18, no. 5, p. 1489, 2018.

[20] A. M. Roy and J. Bhaduri., "Real-time growth stage detection model for high degree of occultation using DenseNet-fused YOLOv4," Computers and Electronics in Agriculture, vol. 193, p. 106694, 2022.

[21] T.-N. Doan, "An efficient system for real-time mobile smart device-based insect detection," International Journal of Advanced Computer Science and Applications, vol. 13, no. 6, 2022.

[22] J. Yin, P. Huang, D. Xiao and B. Zhang, "A Lightweight Rice Pest Detection Algorithm Using Improved Attention Mechanism and YOLOv8," Agriculture, vol. 14, no. 7, p. 1052, 2024.

[23] M. Hussain and R. Khanam, "In-depth review of yolov1 to yolov10 variants for enhanced photovoltaic defect detection," In Solar, vol. 4, no. 3, pp. 351-386, 2024.

[24] C.-Y. Wang and H.-Y. M. Liao, "YOLOv1 to YOLOv10: The fastest and most accurate real-time object detection systems," arXiv preprint, p. arXiv:2408.09332, 2024.

[25] C. Chen, P. Zhang, H. Zhang, J. Dai, Y. Yi, H. Zhang and Y. Zhang., "Deep learning on computational-resource-limited platforms: A survey," Mobile Information Systems, vol. 1, p. 8454327, 2020.

[26] Roboflow. Available online: https://universe.roboflow.com/ip102110000/yoloip1/dataset/1. [Accessed 12 June 2024].

[27] Roboflow. [Online]. Available: https://universe.roboflow.com/oubio/pest-dataset-naoyq/dataset/3. [Accessed 12 June 2024].

[28] U. Sirisha, S. P. Praveen, P. N. Srinivasu, P. Barsocchi and A. K. Bhoi, "Statistical analysis of design aspects of various YOLO-based deep learning models for object detection," International Journal of Computational Intelligence Systems, vol. 1, no. 126, p. 16, 2023

[29] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.

[30] T. Diwan, G. Anirudh and J. V. Tembhurne, "Challenges, architectural successors, datasets and applications," multimedia Tools and Applications, vol. 82, no. 6, pp. 9243-9275, 2023

[31] N. Kumar, Nagarathna and F. Flammini, "YOLO-based light-weight deep learning models for insect detection system with field adaption," Agriculture, vol. 13, no. 3, p. 741, 2023

[32] L. Liu, P. Li, D. Wang and S. Zhu, "A wind turbine damage detection algorithm designed based on YOLOv8," Applied Soft Computing, vol. 154, p. 111364, 2024.

[33] J. Terven, D. Córdova-Esparza and J. Romero-González, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas," Machine Learning and Knowledge Extraction, vol. 5, no. 4, pp. 1680-1716, 2023.

[34] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature Pyramid Networks for Object Detection," arXiv, arXiv:1612.03144, 2017.

[35] S. Liu, L. Qi, H. Qin, J. Shi and J. Jia, "Path Aggregation Network for Instance Segmentation," arXiv, arXiv:1803.01534, 2018.

[36] T. Wu and Y. Dong, "YOLO-SE: Improved YOLOv8 for remote sensing object detection and recognition," Applied Sciences, vol. 13, no. 24, p. 12977, 2023.

[37] M. Noyce, G. Jocher, R. Munawar and Laughing-Q, "COCO Dataset," Ultralytics YOLO, November 2023. [Online]. Available: https://docs.ultralytics.com/datasets/detect/coco/. [Accessed 15 August 2024].

[38] T. Luo, S. Rao, W. Ma, Q. Song, Z. Cao, H. Zhang, J. Xie, X. Wen, W. Gao, Q. Chen, J. Yun and D. Wu, "Individual Tree Spatial Positioning and Crown Volume Calculation Using UAV-RGB Imagery and LiDAR Data," Forests, vol. 15, no. 8, p. 1375, 2024.

[39] C.-Y. Wang, I.-H. Yeh and H.-Y. M. Liao, "Yolov9: Learning what you want to learn using programmable gradient information," arXiv preprint, arXiv:2402.13616, 2024.

[40] W. Xu, D. Zhu, R. Deng, K. Yung and A. W. Ip., "Violence-YOLO: Enhanced GELAN Algorithm for Violence Detection," Applied Sciences, vol. 14, no. 15, p. 6712, 2024.

[41] M. Noyce, R. Munawar, G. Jocher, B. Q and L. Q, "YOLOv9: A Leap Forward in Object Detection Technology," Ultralytics YOLO, March 2024. [Online]. Available: ttps://docs.ultralytics.com/models/yolov9/#what-are-the-advantages-of-using-ultralytics-yolov9-for-lightweight-models. [Accessed 15 August 2024]

[42] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han and G. Ding., "Yolov10: Real-time end-to-end object detection," arXiv preprint, p. arXiv:2405.14458, 2024.

[43] S. Geetha, Athulya, M. A. R. Alif, M. Hussain and P. Allen, "Comparative Analysis of YOLOv8 and YOLOv10 in Vehicle Detection: Performance Metrics and Model Efficacy," Vehicles, vol. 6, no. 3, pp. 1364-1382, 2024.

[44] M. Noyce, R. Munawar, G. Jocher, H. Haffari, Z. X. Wei, A. Vina and B. Q, "YOLOv10: Real-Time End-to-End Object Detection," Ultralytics YOLO, June 2024. [Online]. Available: https://docs.ultralytics.com/models/yolov10/. [Accessed 15 August 2024].

[45] L.-D. Quach, K. N. Quoc, A. N. Quynh and H. T. Ngoc, "Evaluating the eEffectiveness of YOLO models in different sized object detection and feature-based classification of small objects," Journal of Advances in Information Technology, vol. 14, no. 5, pp. 907-917, 2023.

[46] R. Kaur and S. Singh, "A comprehensive review of object detection with deep learning," Digital Signal Processing, vol. 132, p. 103812, 2023.

[47] J. Stodt, C. Reich and N. Clarke, "Unified intersection over union for explainable artificial intelligence," in Intelligent Systems Conference, IntelliSys 2023, Amsterdam, 2024.

[48] A. Tharwat, "Classification assessment methods," Applied computing and informatics, vol. 17, no. 1, pp. 168-192, 2021.

[49] D. A. Anggoro and S. S. Mukti, "Performance Comparison of Grid Search and Random Search Methods for Hyperparameter Tuning in Extreme Gradient Boosting Algorithm to Predict Chronic Kidney Failure," International Journal of Intelligent Engineering & Systems, vol. 14, no. 6, 2021.

[50] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie and L. Farhan, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," Journal of big Data, vol. 8, pp. 1-74, 2021

[51] D. Bergmann and C. Stryker, "What is a loss function?," IBM, 12 July 2024. [Online]. Available: https://www.ibm.com/think/topics/loss-function#:~:text=The%20term%20%E2%80%9Closs%20function%2C%E2%80%9D,intelligence%20(AI)%20model's%20outputs.. [Accessed 17 August 2024]

[52] R. Raj and A. Kos, "An improved human activity recognition technique based on convolutional neural network," Scientific Reports, vol. 13, no. 1, p. 22581, 2023

[53] C.W. Hoe, M. Raheem, and N.F. Abubacker, "News Aggregation and Summarisation," Journal of Applied Technology and Innovation, vol. 8, no. 4, p. 51, 2024.