IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 29 July 2024, accepted 15 August 2024, date of publication 21 August 2024, date of current version 3 October 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3446890

APPLIED RESEARCH

Oil Palm Loose Fruit Detection Using YOLOv4 for an Autonomous Mobile Robot Collector

AHMED FAREED JAPAR, HAFIZ RASHIDI RAMLI, (Member, IEEE), NOR MOHD HAZIQ NORSAHPERI, (Member, IEEE), AND WAN ZUHA WAN HASAN, (Senior Member, IEEE)

Department of Electrical and Electronics Engineering, Universiti Putra Malaysia (UPM), Serdang, Selangor 43400, Malaysia

Corresponding author: Hafiz Rashidi Ramli (hrhr@upm.edu.my)

This work was supported in part by Malaysian Palm Oil Board (MPOB), and in part by Universiti Putra Malaysia Research Grant (Putra) under Grant GP-GPB/2021/9699100.

ABSTRACT This study researches the usage of YOLOv4 for real-time loose fruit detection in oil palm plantations as the first step in implementing automation in the collection of loose fruits. Our system leverages high-resolution video data (4K and 1080p) from various plantation settings. To address the challenges of detecting small and numerous loose fruits, we introduced an image preprocessing technique called ''image tiling'' into the vision system workflow. We studied the effects this has on the performance of the detection model. This involves slicing the image into smaller sections (i.e., tiles) for individual processing by YOLOv4 and YOLOv4-tiny models, enhancing detection accuracy. Refined models (YOLOv4-tiling and YOLOv4-tiny-tiling) are then evaluated. YOLOv4 achieved the highest precision (97%) and F1-score (86.3%), while YOLOv4-tiling offered a slight improvement in recall (80.8%). Notably, YOLOv4-tiny, initially underperforming (precision: 37.2%, recall: 20.9%, F1-score: 25%), showed significant improvement with tiling (precision: 90.5%, recall: 67.1%, F1-score: 73.8%). Also, replacing the SPP layer in YOLOv4 with SPP-Fast resulted in increased precision (92.6%) and a significantly improved F1 score of 91.4%. This vision system was then integrated with a custom-designed Loose Fruit Collector Robot through the Robot Operating System (ROS).

INDEX TERMS Object detection, oil palm automation, loose fruit (LF), YOLO.

I. INTRODUCTION

Palm oil is pivotal in various global economies and everyday life products, from food items to cosmetics and biofuel. In 2022, Malaysia was the second-largest producer of palm oil in the world, accounting for 23.3% of the world's total production [1]. In today's market, collection of loose fruits (i.e. fruitlets that scatter to the ground during the fresh fruit bunch harvesting process, which typically remain uncollected) proves vital to maximizing profit and overall oil output [2], [3]. However, the current process for collecting these loose fruits (hereafter referred to as LF) is ineffective, time-consuming, and physically demanding for the workers. An autonomous robotics system is needed to perform the collection task. UPM developed a proof-of-concept robot through a joint industrial collaboration project in 2020-2021 for such a purpose, and this paper describes the detection system developed for and implemented in that robot.

The main problem faced with implementing an autonomous solution is the detection of these small, numerous, and occluded loose fruits across the plantation field before the collection can be performed. Feng et al. [4] achieved improved recall and average precision in detecting 22 small objects by utilizing an eight (8) times down-sampled feature map of YOLO-V4. This enhancement maintained the model's speed while enabling the detection of objects as small as 5×5 pixels. The model also exhibited superior accuracy in detecting mid-sized and large objects compared to previous versions. The authors further enhanced the accuracy by incorporating anchor boxes that better matched the aspect ratio of small objects and a feature pyramid network [5] to detect tiny objects more effectively. Their modifications resulted in a 5% increase in average precision on the PASCAL VOC dataset [6], with similar frames per second (FPS) as YOLO-V3 [7]. As efficiency becomes a top priority, manual

The associate editor coordinating the review of this manuscript and approving it for publication was Senthil Kumar.

labor is gradually replaced by technological advancements. When trained on natural images and tested on artwork, YOLO outperforms other detection methods like SSD [8] and R-CNN [9], making it more reliable in new domains or when faced with unexpected inputs. However, it is important to note that YOLO falls short of the accuracy achieved by state-of-the-art detection systems. While it excels at quickly identifying objects, it struggles with accurate localization, particularly for small objects.

Currently, existing commercial solutions for fruit detection fall short, highlighting the critical need for tailor-made computer vision systems specifically designed for oil palm plantations [10]. The challenge lies in accurately identifying small fruits against a backdrop of ever-changing environmental factors, such as fluctuating lighting conditions and complex terrain. This emphasizes the need for a robust and efficient computer vision approach, particularly when navigating densely populated plantation environments [11]. YOLOv4-Tiny [12] maintains satisfactory performance and is suitable for mobile devices. It has two YOLO heads and utilizes CSPDarknet53-tiny as its backbone network. YOLOv4 introduced new features such as DropBlock regularization [13], mosaic data augmentation, Cross mini-batch Normalization (CmBN), self-adversarial 62 training (SAT), and Complete Intersection over Union loss (CIoU loss) [14]. Moreover, the emergence of object detection algorithms like YOLOv4 and its derivatives provide the solution. These algorithms come prepared to enhance the accuracy of small object detection across various sectors, including agriculture and robotics [15]. In yet another study, a Yolov4 tiny-based algorithm showed impressive results for detecting green peppers, matching the current state-of-the-art technology [16].

More recently, YOLO has been applied in the detection of LF. Junos et al. [17] focused on the intricate task of individual fruit detection, employing an improved YOLO model to analyze UAV imagery. However, their research acknowledges a limitation where validation is needed on a wider range of datasets. Daud et al. [18] used a broader approach to detecting palm oil trees and LF using the Faster R-CNN algorithm. This information is then used to predict harvest readiness, a valuable tool for optimizing yield. The model's ability to detect fruits beyond oil palm may be limited due to its specific training on oil palm characteristics. This raises concerns about its performance when encountering significant fruit shape, size, and color variations. Deep learning approaches like YOLO have been successful in agricultural object detection. However, there's a need for models optimized for specific tasks. One study by [19] improved YOLOv5 for sprouted potato detection, achieving high accuracy. This highlights the benefit of tailoring models. Additionally, while YOLO is popular, alternative architectures like vision transformers are emerging. A recent study using a lightweight vision transformer by Ke et al. [20] showcases their potential in agriculture in vineyard blade measurement. These findings motivate exploring both improved YOLO models and new architectures for palm oil loose fruit detection.

In this work, the YOLOv4 [12] the model was selected and applied for the real-time detection of LF and tested its performance in the plantation. This is because, during the development process, which began in 2020, YOLOv4 was the state-of-the-art YOLO object detection model. Furthermore, YOLOv4 has high compatibility with ROS 1 compared to the newer models. We also applied a preprocessing technique known as the Image Tiling method to investigate how this technique can affect detection performance. In our hypothesis, the detection performance should be improved but at the cost of longer processing time. Nevertheless, this knowledge will be helpful in determining its feasibility for use in real-time detection of small objects in real-world applications.

II. MATERIALS AND METHODS

A. DATA ACQUISITION

The objective of data collection was to produce a diverse dataset for effective model training and validation. High-resolution video data encompassing LF presence within the field environment was captured in both 4K and 1080p resolutions using the MP4 format. Notably, 4K resolution was prioritized for achieving precise LF detection. Furthermore, a multifaceted approach was employed to fortify the dataset's generalizability and mitigate overfitting tendencies. Video data encompassing a wide spectrum of LF scenarios was collected from a sample size (n=100) of trees distributed across diverse terrain with varying lighting conditions, viewing angles, and occlusions by surrounding foliage. As shown in Fig. 1, data collection for the model's training is done across the various oil palm fields. High-resolution videos (4K and 1080p) in mp4 format were captured, recording LF in different field conditions. Using 4K resolution was crucial for capturing the fine details essential for accurate loose fruit detection. To ensure a robust and versatile model, videos were collected from a hundred trees across diverse terrain with varying lighting, angles, and foliage occlusions, preventing overfitting.

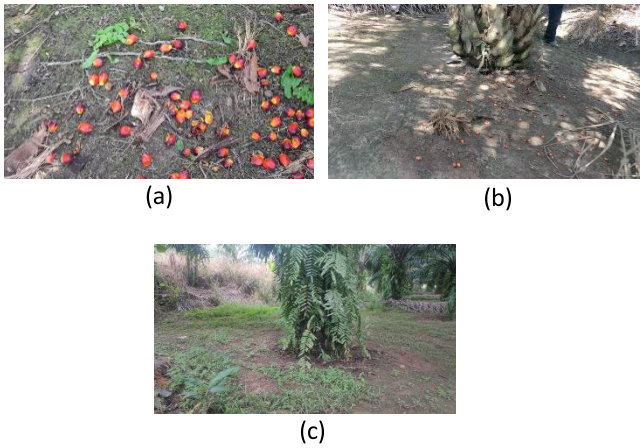


FIGURE 1. (a) Close-up image of LF using 1080p, (b) Far image of LF using 1080p, and (c) Far image of LF using 4K resolution.

TABLE 1. Summary of augmentations with corresponding positive and negative values.

Augmentation	Positive	Negative
Original	949	72
Bright	949	72
Resize	949	72
Zoom and Crop	2575	1081
Total	5422	1297

B. DATA PREPARATION AND TRAINING

Data labeling was conducted following data collection. This involved manually identifying and marking loose fruits within the vast collection of images and videos. This labeled dataset, containing many LF images, helped train the model to recognize loose fruits in real-world scenarios. A combination of dataset variations, early stopping techniques, and regularization methods were employed to prevent overfitting. Furthermore, data augmentation techniques like random zoom and crop, resizing, and brightening were applied to further diversify the dataset and enhance the model's resilience and adaptability for real-world deployment. Referring to Table 1, positive values represented the number of images that were augmented with LF present. However, negative values represent images that do not have LF present in the image.

Data labeling is a crucial step in training a model for LF detection which involves the manual labor of identifying and labeling the LF in the images collected. This task was performed by subject matter experts using specialized software called 'labeling' [21] that allows them to draw the bounding boxes around the LF. The total number of images labeled is 1021 images, containing a total of 14,872 LF. The smallest annotation noted was 13 pixels wide and 10 pixels tall, which indicates that the object labeled is relatively small and the annotation box is big enough to encompass it. On the other hand, the largest annotation was 458 pixels wide and 311 pixels tall, indicating that the annotation box needed to be larger to accommodate the size differences.

After data labeling, data augmentation is conducted. Data augmentation is a technique used to vary the size and diversity of the LF's dataset. It involves applying transformations to the existing images, such as rotation, flipping, scaling, and cropping, to help increase the number of samples in the data set while also making it more robust. For this paper, three techniques were implemented which are: random zoom and crop, resizing to 416 × 416 pixels, and brightening. The dataset was adjusted for brightness using Python's *NumPy* library [22], with a constant value of 125 added to each pixel across different color channels (e.g., Red, Green, Blue) to effectively increase image brightness.

Each augmented dataset was split into 80:20, with 80% used as positive images and the remaining 20% as negative images, as shown in Table 1. After the augmentations, the total dataset collected increased from 1,021 images to 6,719 images. Specifically, this increase was the result of augmenting the images by randomizing zoom and crop images (3,656 images), resizing to 416 × 416 pixels (1,021 images) as well as brightening the images (1,021 images).

The parameters for batch size and subdivision for training YOLOv4 in Table 2(a) and YOLOv4-tiny Table 2(b) were carefully chosen based on AlexeyAB [12] the creator of YOLOv4 as well YOLOv4-tiny. With YOLOv4's larger architecture, batch size 64 and subdivision 16 ensured efficient processing of larger model size, whereas YOLOv4-tiny's smaller model required a smaller batch size 64 and subdivision 1 to prevent memory overflow and ensure stable training. These changes improved training efficiency and resource utilization, aligning with the models' complexities and available hardware capacities.

TABLE 2. Parameters set for testing and training on.

(a) YOLOv4

Model	Parameters	Testing	Training
YOLOv4	Batch	1	64
	Subdivision	1	16
	Input Size	416 x 416, 3 channels	416 x 416, 3 channels
	Momentum	0.949	0.949
	Decay	0.0005	0.0005
	Learning rate	0.001	0.001
	Burn-in	1000	1000
	Max batches	6000	6000
	Policy	"steps"	"steps"
	Steps	4800	5400
	Scales	0.1	0.1

(b) YOLOv4-tiny

Model	Parameters	Testing	Training
YOLOv4-tiny	Batch	1	64
	Subdivision	1	16
	Learning rate	0.001	0.001
	Burn-in	1000	1000
	Max batches	6000	6000
	Policy	"steps"	"steps"
	Steps	4800	5400
	Scales	0.1	0.1

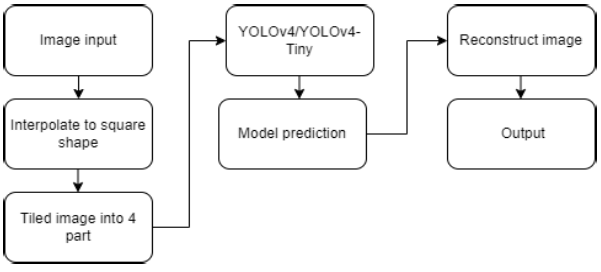


FIGURE 2. Flowchart of tiling method.

C. TILING METHOD

The YOLOv4 object detection algorithm leverages image tiling to achieve significant improvements in the detection

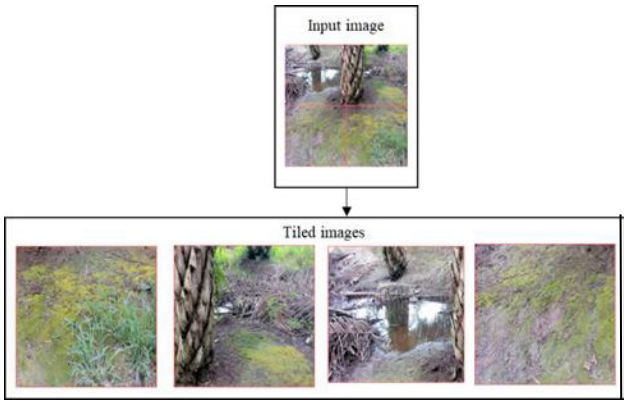


FIGURE 3. Tiling method on an input image.

of small, loose fruits. The Image Tiling process involves an image segmentation technique that partitions the entire image into a grid of smaller sub-images called tiles. Shewajo and Fante [23] this method for malaria screening utilizes a deep learning model on smaller, manageable sections (tiles) obtained by dividing high-resolution microscopic images. Particularly for small objects, advancements have been made through the application of image decomposition techniques such as slicing and tiling. Akyon et al. [24] successfully leveraged this approach, achieving significant improvements in small object detection. Tiling intensifies the model's focus on specific image regions, which in this application would lead to enhanced precision in LF detection. Notably, by concentrating on smaller areas, tiling effectively reduces the generation of false positives, thereby increasing the model's overall reliability and suitability for real-world applications. Also, the tiling technique helps to reduce information loss when entering a YOLOv4 model or any other CNN. This is achieved by dividing the image into smaller tiles, enabling the CNN to focus on specific regions and alleviate information loss caused by limitations in the filter process of the models.

Fig. 2 shows the tiling method's flowchart, which shows how the tiled images are processed. The image is first interpolated into a square shape since the input image to the YOLOv4 algorithm is required to be square (typically with a dimension of 416 × 416 or more). The image is then evenly split into four tiles, each of which is also square-shaped. These tiles are then individually fed into the YOLOv4 model for model prediction.

The illustration in Fig. 3 shows the process of image tiling as performed on an input image. Splitting the image into more tiles could potentially increase the detection accuracy even further, but at the cost of processing time. For this work, the number of tiles selected is four because the aim is to investigate if this tiling approach can have any effect on the detection performance, and four is the minimum number of tiles for a square image. Next, the image is reconstructed, and the detection result for each tile is totalled to give the summative detection result for the entire image. Therefore, the final detection output would be bounding boxes according to the original image's global pixel coordinates based on each tile's relative pixel coordinates.

D. AUTONOMOUS ROBOT LOOSE FRUIT COLLECTOR (ARLFC) FUNCTIONS

The core functionality of the Automated Robotic Loose Fruit Collector (ARLFC) revolves around its ability to navigate autonomously through palm tree rows while simultaneously performing loose fruit detection and retrieval. This system utilizes data streams from two Logitech Brio webcams to effectively identify loose fruits on the palm trees' left or right sides. Upon successful detection, a communication protocol transmits a signal to the autonomous robot, instructing it to maneuver toward the identified palm tree.

In the context of LF detection, as shown in Fig. 4, the system leverages the publishing of LF presence values by two cameras – '0' indicating no LF, '1' for the left camera detecting an LF, and '2' for the right camera. This information is published to the 'Loose fruit detection camera' topic within ROS [25]. ROS is a robot operating system that provides a framework for different parts of the robot to communicate using nodes. Nodes subscribed to this topic can then utilize this data effectively, for instance, to guide robotic movements towards identified loose fruits.

The collection system comprises three stages, mainly the collection stage, vacuum stage, and filtration stage. We used a specially designed roller in the collection stage to efficiently collect the LFs scattered across the ground. Once the LFs have been collected, they are transferred to a collection drum via a vacuum that is powered by three brushless DC motors that work unilaterally as a vacuum to drag all the LFs from the roller and into the collection drum. Once collection is completed and extraction is needed, the LFs that were collected are then released onto a separation drum that rotates at a constant speed, acting as a separator for the LFs that may have dirt or any unwanted debris attached. The end products are filtered LFs that were successfully and efficiently collected using the ARLFC.

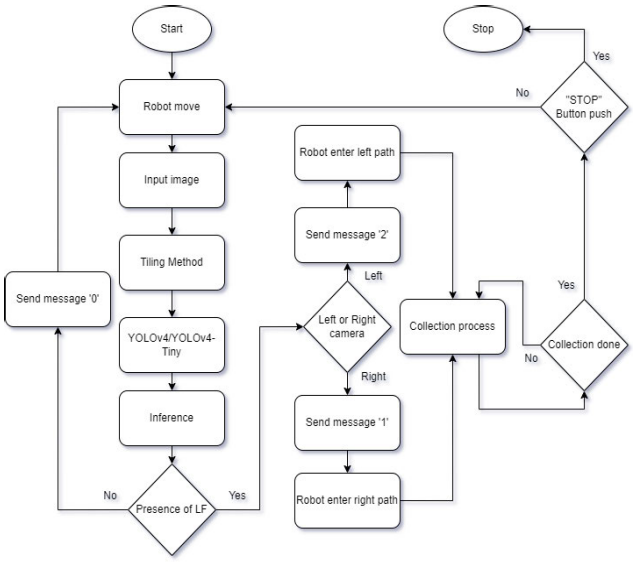


FIGURE 4. Flowchart of ROS integration with the ARLFC.

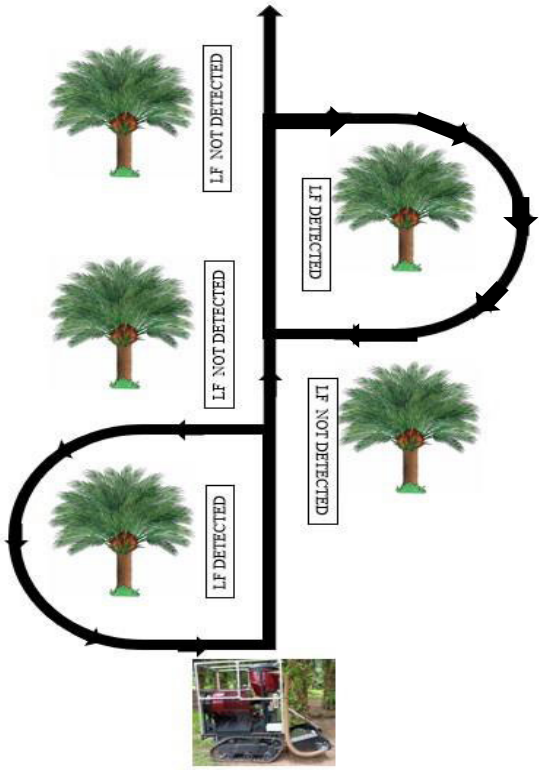


FIGURE 5. Path for ARLFC collection.

The planned path of the ARLFC is shown in Fig. 5. To ensure comprehensive fruit collection, the ARLFC employs a systematic path planning strategy, encircling each tree twice. This double-pass approach mitigates the risk of overlooking any loose fruits. Once activated, the ARLFC will advance in a straight line based on the path planning made prior. As it moves, the onboard vision system will actively search for loose fruits in the area to the left and right of the robot and notify whether there are any LF around the individual trees it passes. If LF is present, the ARLFC will enter the tree site, which will start collecting the LF around the tree base following a circular path. This path planning was made based on the layout of the plantation at the testing site. The ARLFC design and movements are to accommodate the vegetation and terrain conditions in this specific plantation site, but we were informed that the same layout was used in other plantations as well. Following the fruit collection maneuver, the ARLFC seamlessly progresses through the plantation row, enabled by the continuous operation of the YOLOv4 object detection program. Lastly, a Nvidia Jetson AGX Xavier series GPU and Ubuntu 20.04-LTS [26] was used as the OS to run the detection system.

E. PERFORMANCE COMPARISON WITH OTHER YOLO MODELS

As previously stated, this work represents the culmination of a research project initiated in 2020. Given the current benchmark in object detection, YOLOv8 [27], which surpasses previous models in performance, this study employs both YOLOv8n and YOLOv8m for a comprehensive comparative analysis against our prior methodologies. A key innovation in YOLOv8 is the integration of the SPP-Fast layer, which was also used in YOLOv5 [28], a mechanism designed to accelerate pooling operations without compromising accuracy. This enhancement positions YOLOv8 as a prime candidate for real-time applications due to its improved efficiency. The training parameters used are shown in Table 3.

TABLE 3. Parameters set for training YOLOv8.

Parameters	Value
task	detect
mode	train
model	yolov8n.pt
epochs	300
batch	16
imgsz	416
workers	8
optimizer	auto
verbose	TRUE
seed	0
deterministic	TRUE
val	TRUE
iou	0.7
max_det	300
lr0	0.01
lrf	0.01
momentum	0.937
weight_decay	0.0005
warmup_epochs	3
box	7.5
cls	0.5
dfl	1.5
nbs	64
flipud	0
flip	0.5
mosaic	1

Furthermore, in this study, the potential of incorporating the SPP-Fast layer, as shown in Fig. 6, into the YOLOv4 architecture shown in Fig. 7 to achieve a similar performance boost is investigated. A comparative analysis of detection performance between YOLOv4-SPPFast and multiple YOLO models under diverse conditions, including varying object distances and occlusions, will be presented. Theoretically, SPP-Fast lowers resource consumption, facilitating the deployment of YOLO models on resource-constrained devices. Its simpler implementation reduces development overhead, making integrating into existing YOLO architectures easier [29].

III. RESULTS AND DISCUSSION

A. YOLOv4 AND YOLOv4-TINY TRAINING PERFORMANCE COMPARISON

The object detection model undergoes training with annotated images, where objects are labeled with bounding boxes and class identifiers to optimize the model's parameters, enabling it to recognize and locate objects in new images. Mean average precision (mAP) is a crucial metric reflecting precision and recall. In Fig. 8(a), the standard YOLOv4 model

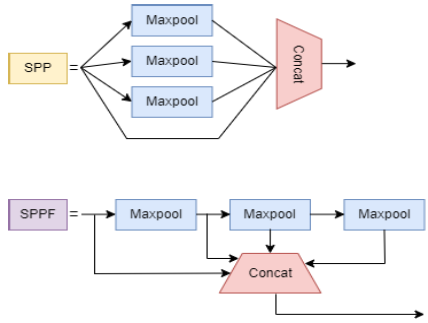


FIGURE 6. SPP-fast and SPP architecture.

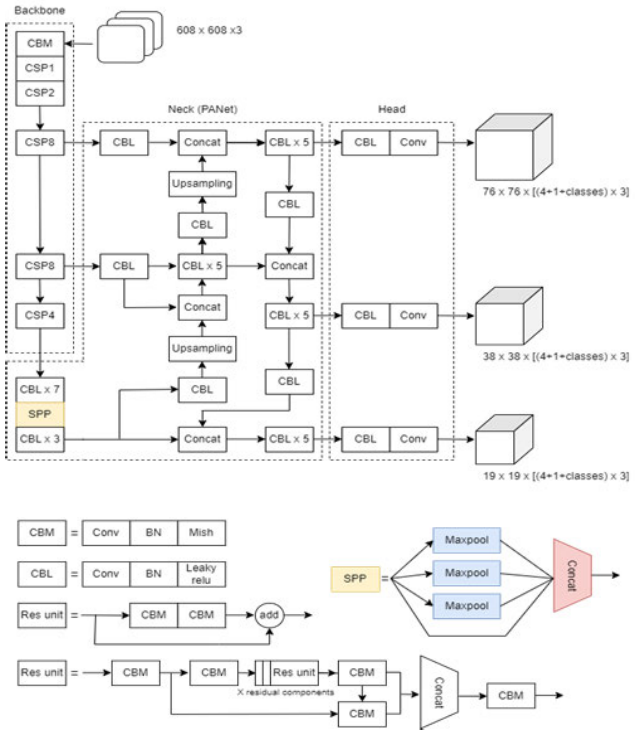
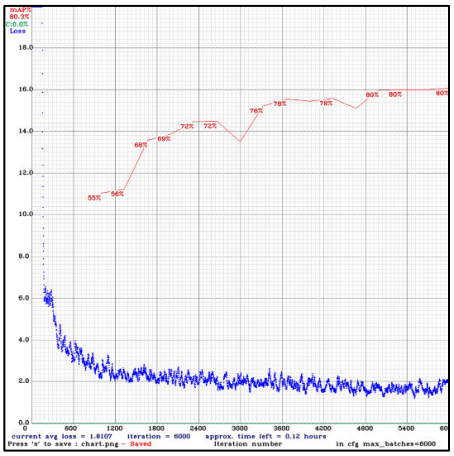
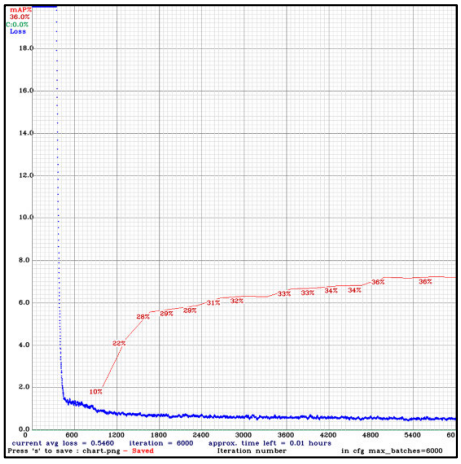


FIGURE 7. YOLOv4 architecture.



(a)



(b)

FIGURE 8. Training result for (a) YOLOv4 (b) YOLOv4-Tiny.

achieved an 80% mAP during training. However, its loss function shows stagnation around 1.817, indicating limited learning progress and diminishing returns. Further training may not significantly improve performance. For Fig. 8(b), using YOLOv4-Tiny, with a lower mAP of 34%-36%, persistent fluctuations around 0.54 in its loss function are observed. This suggests limited learning effectiveness and constraints. Further training is unlikely to enhance performance significantly.

B. COMPARISON OF DETECTION PERFORMANCE CLOSE-UP IMAGES

As shown in Fig. 9, the YOLOv4 model performed better, achieving exceptionally high precision scores. Precision is a metric for gauging the model's ability to accurately identify objects within proximity (under 2 meters). YOLOv4's recall and f1-score were also the highest among the models. Recall reflects the model's capacity to detect a significant portion of the actual objects in the images, while f1-score provides a harmonic mean, balancing precision and recall. The equation of precision (P), recall (R), and the F1-score are stated below, where TP refers to True Positive data, and TN refers to True Negative data:

$$P = \frac{TP}{TP + FP} \tag{1}$$

$$R = \frac{TP}{TP + FN} \tag{2}$$

$$F_1 = \frac{2 \times P \times R}{P + R} \tag{3}$$

Conversely, the YOLOv4-Tiny model encountered significant challenges in close-up object detection. While the implementation of tiling techniques demonstrably improved the performance of both YOLOv4 and YOLOv4-Tiny in this domain, their overall accuracy remained lower compared to YOLOv4. It is important to note that all models exhibited limitations when handling close-up objects. This could be due to an inaccurate representation of the size of each LF.

The evaluation process unequivocally identified YOLOv4 as the most adept model for close-up object detection tasks. YOLOv4-Tiny, on the other hand, exhibited the weakest

performance. While YOLOv4-Tiling and YOLOv4-Tiny-Tiling demonstrated some promise, their suitability might be limited to specific applications that utilize larger input

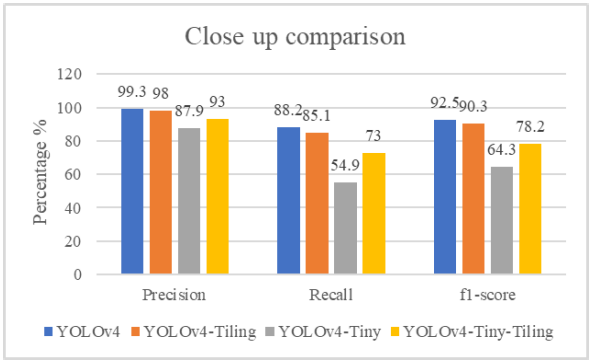


FIGURE 9. Average performance for close up images.

FIGURE 10. YOLOv4 model detection effect on close-up images.

images and prioritize accuracy over real-time processing. Therefore, the optimal model selection hinges on a thorough understanding of the specific task requirements to ensure the most effective performance.

1) NORMAL DISTANCE IMAGES (>2 METERS)

The evaluation process revealed that both the YOLOv4 and YOLOv4-Tiling models excel at detecting loose palm oil fruits located further away from the tree (beyond 2 meters). They achieved higher precision (correctly identified fruits), recall (all actual fruits detected), and f1-score (harmonic mean of precision and recall) compared to other models. As shown in Fig. 11 and Fig. 12, when comparing the performance of all the models, the YOLOv4-Tiling model exhibited a slight edge in detection performance, it comes at a cost – its processing time is significantly longer, making it unsuitable for real-time applications where speed is crucial. On the other hand, the YOLOv4-Tiny and YOLOv4-Tiny-Tiling models, designed for efficiency with a reduced size, displayed lower precision, recall, and f1-score. This trade-off in size for speed comes at the expense of accuracy, particularly for detecting small or distant objects. mean of precision and recall) compared to other models.

As shown in Fig. 11 and Fig. 12, when comparing the performance of all the models, the YOLOv4-Tiling exhibited a slight edge in detection performance, but it comes at a cost. Its processing time is significantly longer, making it unsuitable for real-time applications where speed is crucial. Fig. 11 shows a sample image of the detected object in 4k for above 2 meters, which shows that the pixel density of the detected LF is around 35 × 25 pixels, as shown in Fig. 14. The increase in density brings in more information compared to the average pixel density of 1080p LF in the image is at 20 × 10 pixels. On the other hand, the YOLOv4-Tiny and YOLOv4-Tiny-Tiling models were designed for efficiency with a reduced size and displayed lower precision, recall, and f1-score. This trade-off in size for speed comes at the expense of accuracy, particularly for detecting small or distant objects.

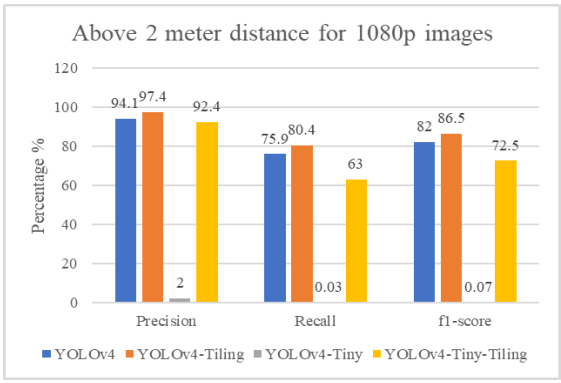


FIGURE 11. Average performance for distance above 2 meters for 1080p.

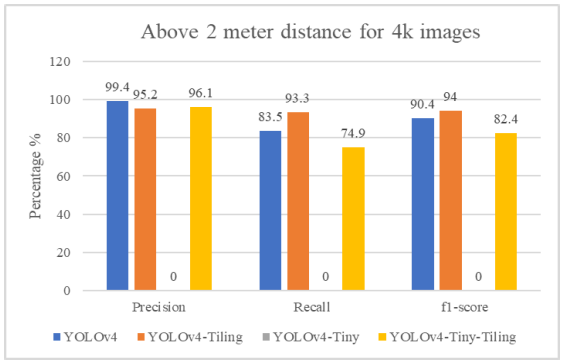


FIGURE 12. Average performance for distances above 2 meters for 4K.

FIGURE 13. YOLOv4-Tiling above 2 meters for 4K.

2) OCCLUSION IMAGES

Fig. 15 presents the evaluation results of four object detection models on a test dataset containing occluded images of palm oil loose fruit. YOLOv4 achieved the highest precision

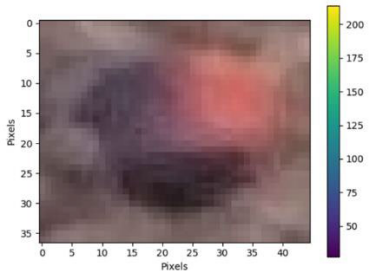


FIGURE 14. The pixel density of LF detected in 4K resolution images.

(0.974), signifying high accuracy in its detections. However, its lower recall (0.74) indicates it missed a significant portion of the actual loose fruits in the images. This imbalance between precision and recall led to a lower F1-score (0.82) compared to other models. YOLOv4-Tiling displayed a different trade-off. While its F1-score (0.747) fell slightly below YOLOv4's, it exhibited higher recall and precision. This suggests that YOLOv4-Tiling successfully detected a greater proportion of the occluded loose fruits while maintaining good accuracy. YOLOv4-Tiny encountered significant challenges in detecting occluded objects, reflected in its low F1-score (0.358). This model struggled to accurately identify loose fruits, generating a relatively high number of false positives (precision: 0.59) and missing a substantial number of actual fruits (low recall: 0.285). Conversely, YOLOv4-Tiny-Tiling exhibited a noteworthy improvement with an F1-score of 0.623. This highlights the effectiveness of the tiling technique in boosting model performance for occluded object detection.

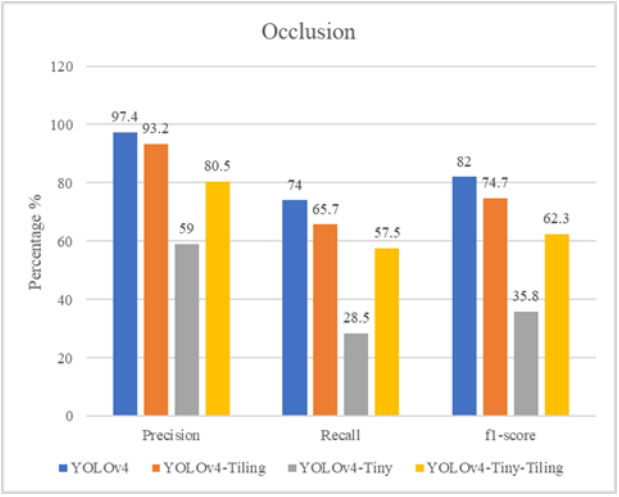


FIGURE 15. Average performance for occluded images.

The study further investigates the limitations of YOLOv4s-Tiling. It suggests that its lower overall performance compared to YOLOv4 might be attributed to overfitting. Overfitting occurs when a model becomes overly specialized on the training data, potentially prioritizing irrelevant details over critical features. This can be particularly problematic in small object detection scenarios, where the model might focus on minute details in the training data that are not generalizable to real-world

FIGURE 16. YOLOv4-Tiling model detection effect on occlusion images.

occluded images. Furthermore, occlusions themselves might exacerbate overfitting issues in YOLOv4-Tiling.

Fig. 16 illustrates this phenomenon. The figure shows numerous false negatives (missed detections) by YOLOv4-Tiling, potentially decreasing overall detection accuracy. This suggests that occlusions, rather than assisting the model, might hinder its ability to effectively generalize and identify loose fruits.

C. OVERALL DETECTION PERFORMANCE

The dataset encompassed a diverse range of images, including close-up shots, distant objects, and scenarios with occluded palm oil fruits. Table 4 shows that the original YOLOv4 model achieved the highest precision and F1 score, indicating its ability to accurately detect objects and achieve a good balance between precision and recall. YOLOv4-Tiling displayed a slight edge in the recall, suggesting it might detect a few more true positives compared to YOLOv4.

This suggests it struggled with accurate object detection in the diverse scenarios presented by the dataset. YOLOv4-Tiny-Tiling, however, demonstrated improvements compared to its smaller counterpart, suggesting the tiling technique might offer some benefit in specific situations. The optimal model selection ultimately depends on the specific characteristics of the dataset you're working with and the available computational resources. YOLOv4 and YOLOv4-Tiling emerged as the top performers in this evaluation, but the choice might differ depending on your project's requirements. To understand the evaluation metrics better, let's delve into the terminology. The dataset was classified into four

TABLE 4. Analysis of different YOLOv4 models used.

Model Name	Inference (ms)	Precision	Recall	F1-Score
YOLOv4	16.129	97%	80%	86.3%
YOLOv4-Tiling	334.811	95.5%	80.8%	86%
YOLOv4-Tiny	4.244	37.2%	20.9%	25%
YOLOv4-Tiny-Tiling	84.634	90.5%	67.1%	73.8%

categories: True Positive (TP) represents correctly identified objects, False Positive (FP) indicates incorrectly identified objects, True Negative (TN) signifies correctly classified

non-objects, and False Negative (FN) denotes missed objects. These categories contribute to the calculation of precision, recall, and F1-score.

Table 4 illustrates that the trade-off between processing speed and detection accuracy becomes evident when choosing a YOLOv4 variant. YOLOv4-Tiny boasts the fastest inference time, making it ideal for real-time applications where speed is paramount. However, this speed comes at a cost since its detection accuracy is lower compared to other YOLOv4 variations. The standard YOLOv4 model strikes a good balance between speed and accuracy. It offers faster inference time compared to both YOLOv4-Tiling and YOLOv4-Tiny-Tiling, making it a versatile option for various applications. YOLOv4-Tiling, due to its dependence on processing image tiles, incurs the longest inference time among the models evaluated. This extended processing time makes it less suitable for real-time applications. YOLOv4-Tiny-Tiling offers a compromise compared to the other two. While it processes images faster than YOLOv4-Tiling, it's still slower than the standard YOLOv4 model. This intermediate processing speed makes it a potential choice for applications that can tolerate some delay but require handling larger images. In essence, the selection of the most suitable YOLOv4 variant hinges on the specific needs of your application. If real-time processing is crucial, YOLOv4-Tiny might be the best option, even if it means sacrificing some accuracy. Conversely, if dealing with large images is a priority and real-time processing isn't essential, YOLOv4-Tiling or YOLOv4-Tiny-Tiling could be better choices.

D. ROS INTEGRATION AND ON-FIELD TESTING

For practical implementation, all four YOLOv4 models (YOLOv4-Tiling, YOLOv4, YOLOv4-Tiny, and YOLOv4-Tiny-Tiling) were transformed into a ROS (Robot Operating System) package. This integration streamlines communication between the object detection system and the robot.

FIGURE 17. Real-time demonstration of onboard LF detection system.

Upon detecting a loose fruit (LF) object, the vision system transmits a message to the central control node within the ROS framework. This message encapsulates crucial information about the detected LF, including its location (position and dimensions) relative to the robot. A simple coding scheme is employed: "1" signifies LF detected on the left camera, "2" indicates LF on the right camera, and "0" represents no detection. Leveraging this information, the autonomous robot strategically adjusts its positioning to approach the identified loose fruit. Fig. 18 exemplifies a typical message structure, visualizing the camera used and the detected object's details. Additionally, Fig. 17 provides a glimpse of the real-world scenario, while Fig. 19 offers the robot's computer vision perspective.

The system prioritizes trees with a higher concentration of loose fruits, dynamically adjusting the focus on the corresponding camera for optimal detection. Experimental evaluation explored the impact of streaming resolutions (1080p and 4K) on performance. However, field testing with higher-resolution video streams was limited due to battery life constraints. Power consumption is critical for real-world deployment as it directly affects the robot's operational performance and endurance.

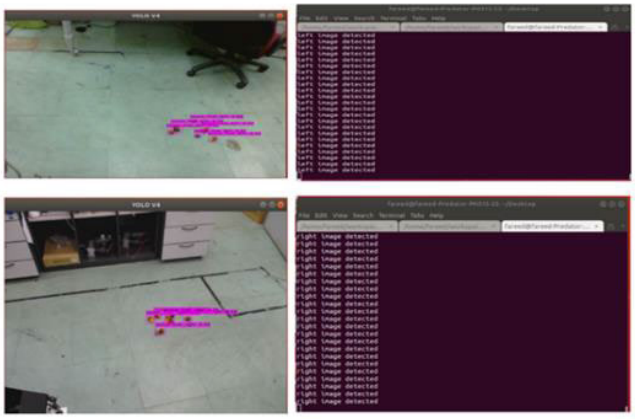


FIGURE 18. ROS message published by the vision system.

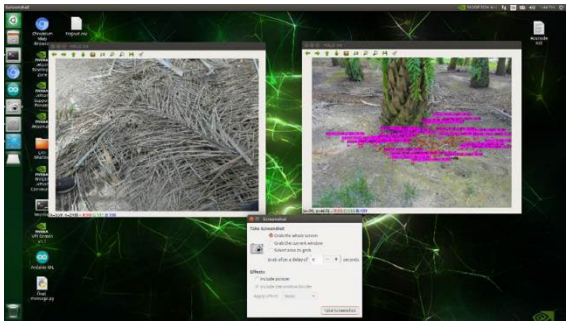


FIGURE 19. Vision system interface.

Table 5 investigates the effectiveness of YOLOv4 models for counting LF during on-site testing after the system has been integrated into ROS. The evaluation compares the performance of four models that are fed with the same real-time video: non-tiling YOLOv4, Tiling YOLOv4, non-tiling YOLOv4-Tiny, and Tiling YOLOv4-Tiny.

Each model's accuracy is assessed by comparing its predicted LF count with the actual number of fruits present (ground truth). The results reveal that Tiling YOLOv4 exhibits the most promising performance overall, achieving accurate fruit counts in some cases and minimal discrepancies in others, with an average error rate of 7.833%. Non-tiling YOLOv4 also demonstrates potential with an average error

TABLE 5. LF collection performance on-field.

Model	Tree	Actual	Model Estimation	Difference	Error %
YOLOv4	1	20	17	-3	15
	2	8	10	2	25
	3	6	6	0	0
	4	6	6	0	0
	5	4	4	0	0
	6	4	3	-1	25
	7	16	13	-3	18.75
	8	7	6	-1	14.29
	9	7	7	0	0
	10	20	20	0	0
				Average Error	9.804
YOLOv4-Tiny	1	20	0	-20	100
	2	8	0	-8	100
	3	6	0	-6	100
	4	6	0	-6	100
	5	4	0	-4	100
	6	4	0	-4	100
	7	16	1	-15	93.75
	8	7	1	-6	85.71
	9	7	0	-7	100
	10	20	1	-19	95
				Average Error	97.446
YOLOv4-Tiling	1	20	24	-4	20
	2	8	8	0	0
	3	6	6	0	0
	4	6	4	2	33.33
	5	4	5	-1	25
	6	4	4	0	0
	7	16	16	0	0
	8	7	7	0	0
	9	7	7	0	0
	10	20	20	0	0
				Average Error	7.833
YOLOv4-Tiny-Tiling	1	20	18	-2	10
	2	8	12	4	50
	3	6	3	-3	50
	4	6	5	-1	16.67
	5	4	4	0	0
	6	4	4	0	0
	7	16	14	-2	12.5
	8	7	6	-1	14.29
	9	7	7	0	0
	10	20	18	-2	10
				Average Error	16.346

of 9.804% for certain trees. From the table, we can see that for YOLOv4, the error occurs in 5 trees compared to YOLOv4-Tiling, which has errors on only 3 trees, thus lowering its overall average error. The benefit of tiling shows a little bit of improvement, around a 2% increase in performance compared to normal YOLOv4.

Interestingly, both Non-Tiling and Tiling versions of YOLOv4-Tiny, the smaller model architecture, performed poorly in all scenarios, with YOLOv4-Tiny having the highest average error of 97.446% while YOLOv4-Tiny-Tiling has a lower average error 16.346%. From this observation, the tiling method increases the performance of YOLOv4-Tiny by 81.1%. The impact of the tiling method is due to more information being gathered and given to the model than in the old architecture, where the image is compressed to accommodate the input architecture of the model of YOLOv4-Tiny. This underestimation is also likely attributed to the reduced model size and potentially less training data on palm oil fruit images compared to the larger YOLOv4 models.

TABLE 6. Training models mAP50.

Model Name	mAP50 (%)
YOLOv4	**80.3**
YOLOv4-Tiny	36
YOLOv4-SPP-fast	76.3
YOLOv8m	77.6
YOLOv8n	73

The findings highlight the potential of using preprocessing techniques such as image tiling and deep learning algorithms for real-time fruit counting in precision agriculture, particularly palm oil plantations. However, the study also emphasizes the importance of algorithm workflow design, model selection, and configuration. Notably, the research reveals a significant benefit of using a tiling approach with the YOLOv4-Tiny model. While both YOLOv4 models perform well, the YOLOv4-Tiny-Tiling model substantially improves fruit count accuracy compared to its non-tiling counterpart. This suggests that tiling can potentially overcome limitations associated with the smaller model size and potentially less training data. Utilizing a tiling approach allows for the deployment of a model on devices without the need for computationally intensive larger models.

E. PERFORMANCE COMPARISON WITH OTHER YOLO MODELS

The result of the training mAP50 comparison for all the models selected for this work is shown in Table 6. In the case of the metrics measured here, mAP50 was used, which means the confidence must be at a 50% confidence threshold. The standard YOLOv4 model achieved an 80.3% mAP50 during training, which is the highest. YOLOv8m follows closely with a commendable 77.6%, indicating its efficiency while potentially offering improvements in speed and model size compared to YOLOv4. The YOLOv4-SPP-fast model, designed to enhance processing speed, achieves a respectable 76.3%, showcasing a balance between speed and accuracy. Meanwhile, YOLOv8n, a more compact version of the YOLOv8 series, records a solid 73%, reflecting its ability to maintain reasonable accuracy with reduced computational requirements. In contrast, YOLOv4-Tiny, designed for high-speed and real-time applications, achieves a significantly lower mAP50 of 36%, highlighting the trade-offs between model size, speed, and detection accuracy in lightweight models.

To compare the performance of our proposed models to the current state-of-the-art model, refer to Table 7, which presents an overview of the performance metrics for various YOLO models. Table 7 illustrates that the trade-off between processing speed and detection accuracy is evident when choosing the optimal model. YOLOv4-Tiny, with an inference time

of 4.244 milliseconds, is the fastest, making it suitable for real-time applications where lower detection accuracy (Precision: 37.2%, Recall: 20.9%, F1-Score: 25%) is acceptable. However, the newly proposed YOLOv4-SPP-fast provides a balanced performance with an inference time of 12.11 milliseconds with relatively high accuracy (Precision: 92.6%, Recall: 92%, F1-Score: 91.4%). It outperforms the standard YOLOv4 for this specific application, with an inference time of 16.129 milliseconds and an F1-Score of 86.3%.

TABLE 7. YOLO models comparison.

Model Name	Inference (ms)	Precision (%)	Recall (%)	F1-Score (%)
YOLOv4	16.129	**97.1**	80	86.3
YOLOv4-Tiling	334.811	95.5	80.8	86
YOLOv4-Tiny	**4.244**	37.2	20.9	25
YOLOv4-Tiny-Tiling	84.634	90.5	67.1	73.8
YOLOv4-SPP-fast	12.11	92.6	**92**	**91.4**
YOLOv4-SPP-fast-Tiling	457.71	88.8	90.8	88.76
YOLOv8m	5.712	86.1	61.51	68.34
YOLOv8m-Tiling	172.133	84.2	82.5	81
YOLOv8n	4.317	86.4	60.8	67.3
YOLOv8n-Tiling	163.253	85.3	84.7	82.5

Among the tiling models, YOLOv4-Tiny-Tiling offers a fair compromise with an inference time of 84.634 milliseconds, as it is faster than YOLOv4-Tiling's 334.811 milliseconds but slower than the standard YOLOv4. YOLOv4-SPP-fast-Tiling, with the longest inference time of 457.71 milliseconds, is less suitable for real-time applications despite its relatively high accuracy (F1-Score: 88.76%). YOLOv8 models show improvements, with YOLOv8n achieving 4.317 milliseconds and an F1-Score of 67.3% and YOLOv8m achieving 5.712 milliseconds and an F1-Score of 68.34%. The tiling versions of YOLOv8 (YOLOv8n-Tiling and YOLOv8m-Tiling) also demonstrate good performance, with F1 scores of 82.5% and 81%, respectively. Ultimately, selecting the most suitable model depends on the application's specific requirements, balancing the need for speed and accuracy.

IV. CONCLUSION

This work describes the development of an oil palm loose fruit detection system for an autonomous loose fruit collection robot. The system utilizes custom YOLOv4 and YOLOv4-Tiny models, trained on a specifically made dataset to detect LF in the real-world plantation environment. These models displayed impressive accuracy when tested in real-world scenarios. While there have been previous publications investigating the application of the YOLO models in the detection of loose fruits, this is the first study that involves using the YOLO models together with the Image Tiling preprocessing method to improve LF detection.

The YOLOv4-Tiling model emerged as a standout performer, particularly adept at detecting small objects even at long distances. It achieved a significant improvement, with an f1-score increase of approximately 4-5% compared to other models. This tiling technique further enhanced the system's effectiveness when dealing with challenging tasks like processing large 4K images. In addition, the newly proposed YOLOv4-SPP-fast model emerged as a standout performer, particularly adept at detecting small objects even at long distances. Replacing the SPP layer in YOLOv4 with SPPFast significantly increased precision and F1 scores, achieving an F1 score of 91.4%, compared to 86.3% for the standard YOLOv4. This improvement highlights the effectiveness of the SPPFast layer in enhancing detection accuracy while maintaining faster inference times.

Furthermore, this work describes the integration of the vision system with a robotic loose fruit collection platform that was tested in an oil palm plantation environment. The tests were successful, although occasional frame rate drops (fps) occurred due to GPU overheating. The robot's battery life limited its operational continuity to around four hours. While the initial research primarily used YOLOv4 models trained on 1080p images, future advancements could benefit from utilizing high-resolution 4K training data exclusively. This data offers richer detail, which is crucial for detecting small objects. Additionally, the reliance on expensive and power-hungry Jetson Xavier hardware presented challenges with overheating. Exploring more affordable and energy-efficient hardware alternatives is crucial for long-term, reliable loose fruit detection and collection.

Outside of the specific application presented in this paper, the findings of this work can also be applied to any detection problem where the object of interest is particularly small relative to the entire image captured, such as in geographic information systems (GIS) and unmanned aerial vehicle (UAV)-based image detection. It would be of interest to explore the possibilities of further detection improvements by integrating a hybrid backbone and leveraging the advancements in vision transformers to replace the current YOLO backbone for enhanced feature extraction and improved scalability.

ACKNOWLEDGMENT

The authors would like to thank all the industry partners that were involved in this project.

REFERENCES

[1] G. K. A. Parveez, "Oil palm economic performance in Malaysia and R&D progress in 2022," *J. Oil Palm Res.*, vol. 35, no. 2, pp. 193–216, 2022.

[2] *Technologies for Oil Palm Harvesting, Evacuation and Loose Fruit Collection*, MPOB, Bandar Baru Bangi, Selangor, 2009.
[3] I. E. Henson, "Ripening, harvesting, and transport of oil palm bunches," in *Palm Oil*. Amsterdam, The Netherlands: Elsevier, 2012, pp. 137–162.
[4] D. Feng, M. Liang, and G. Wang, "Improved YOLOv4 based on dilated convolution and focal loss," in *Proc. IEEE Int. Conf. Adv. Electr. Eng. Comput. Appl. (AEECA)*, Aug. 2021, pp. 966–971.
[5] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 936–944.
[6] D. Hoiem, S. K. Divvala, and J. H. Hays, "PASCAL VOC 2008 challenge," *World Lit. Today*, vol. 24, no. 1, pp. 1–4, 2008.
[7] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.
[8] W. Liu, "SSD: Single shot multibox detector," in *Proc. 14th Eur. Conf. Comput. Vis. (ECCV)*. The Netherlands: Springer, Oct. 2016, pp. 21–37.
[9] Q. Fan, L. Brown, and J. Smith, "A closer look at faster R-CNN for vehicle detection," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2016, pp. 124–129.
[10] F. Gao, L. Fu, X. Zhang, Y. Majeed, R. Li, M. Karkee, and Q. Zhang, "Multi-class fruit-on-plant detection for apple in SNAP system using faster R-CNN," *Comput. Electron. Agricult.*, vol. 176, Sep. 2020, Art. no. 105634.
[11] D. Xiao, F. Shan, Z. Li, B. T. Le, X. Liu, and X. Li, "A target detection model based on improved tiny-Yolov3 under the environment of mining truck," *IEEE Access*, vol. 7, pp. 123757–123764, 2019.
[12] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.
[13] G. Ghiasi, T.-Y. Lin, and Q. V Le, "DropBlock: A regularization method for convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–11.
[14] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 658–666.
[15] M. Kulshreshtha, S. S. Chandra, P. Randhawa, G. Tsaramirsis, A. Khadidos, and A. O. Khadidos, "OATCR: Outdoor autonomous trash-collecting robot design using YOLOv4-tiny," *Electronics*, vol. 10, no. 18, p. 2292, Sep. 2021.
[16] J. Wang, Z. Gao, Y. Zhang, J. Zhou, J. Wu, and P. Li, "Real-time detection and location of potted flowers based on a ZED camera and a YOLO V4-tiny deep learning algorithm," *Horticulturae*, vol. 8, no. 1, p. 21, Dec. 2021.
[17] M. H. Junos, A. S. M. Khairuddin, S. Thannirmalai, and M. Dahari, "Automatic detection of oil palm fruits from UAV images using an improved YOLO model," *Vis. Comput.*, vol. 38, no. 7, pp. 2341–2355, Jul. 2022, doi: 10.1007/s00371-021-02116-3.
[18] M. M. Daud, Z. Kadim, and H. H. Woon, "Detection of oil palm tree and loose fruitlets for fresh fruit Bunch's ready-to-harvest prediction via deep learning approach," *IAENG Int. J. Comput. Sci.*, vol. 50, no. 4, pp. 1183–1193, 2023.
[19] G. Dai, J. Fan, S. Yan, and R. Li, "Research on detecting potato sprouting based on improved YOLOV5," *IEEE Access*, vol. 10, pp. 85416–85428, 2022, doi: 10.1109/ACCESS.2022.3192406.
[20] S. Ke, G. Dai, H. Pan, and B. Jin, "Intelligent vineyard blade density measurement method incorporating a lightweight vision transformer," *Egyptian Informat. J.*, vol. 26, Jun. 2024, Art. no. 100456, doi: 10.1016/j.eij.2024.100456.
[21] D. Tzutalin. (2024). *LabelImg*. Github. [Online]. Available: https://github.com/HumanSignal/labelImg
[22] C. R. Harris, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.
[23] F. A. Shewajo and K. A. Fante, "Tile-based microscopic image processing for malaria screening using a deep learning approach," *BMC Med. Imag.*, vol. 23, no. 1, pp. 1–14, Mar. 2023, doi: 10.1186/s12880-023-00993-9.
[24] F. C. Akyon, S. O. Altinuc, and A. Temizel, "Slicing aided hyper inference and fine-tuning for small object detection," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2022, pp. 966–970.
[25] M. Quigley, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, 2009, p. 5.
[26] M. Helmke, *Ubuntu Linux Unleashed 2021 Edition*. Reading, MA, USA: Addison-Wesley, 2020.
[27] G. Jocher, A. Chaurasia, and J. Qiu. (2023). *Ultralytics YOLO (Version 8.0. 0)[Computer software]*. [Online]. Available: https//github.com/ultralytics/ultralytics
[28] G. Jocher, *Ultralytics/YOLOv5: v6. 0-YOLOv5n'Nano'Models, Roboflow Integration, TensorFlow Export, OpenCV DNN Support*. Genève, Switzerland: Zenodo, 2021.
[29] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 764–773.

AHMED FAREED JAPAR received the B.Eng. degree (Hons.) in electrical and electronic engineering from Universiti Putra Malaysia, in 2020, where he is currently pursuing the master's degree in robotics and automation. His research interests include robotics, image processing, machine learning, and deep learning.

HAFIZ RASHIDI RAMLI (Member, IEEE) received the B.Eng. degree in electrical and electronic engineering and the M.Sc. degree in control and automation engineering from Universiti Putra Malaysia (UPM), in 2007 and 2010, respectively, and the Ph.D. degree in biomedical engineering from Imperial College London, in 2017. Currently, he is a Professional Engineer and a Senior Lecturer with UPM, where his research interests include artificial intelligence, computer vision, robotics, and haptics.

NOR MOHD HAZIQ NORSAHPERI (Member, IEEE) received the B.Eng. and M.Sc. degrees in mechatronics engineering from International Islamic University Malaysia (IIUM), in 2015 and 2017, respectively, and the Ph.D. degree in electrical engineering from Universiti Teknologi Malaysia, in 2020. He is currently a Senior Lecturer with the Department of Electrical and Electronic Engineering, Universiti Putra Malaysia (UPM). His research interests include nonlinear control, robotics, and artificial intelligence.

WAN ZUHA WAN HASAN (Senior Member, IEEE) received the degree in electrical and electronic engineering from Universiti Putra Malaysia, in 1997, and the Ph.D. degree in microelectronic engineering from Universiti Kebangsaan Malaysia, in 2010. He is currently a Professor with the Department of Electrical and Electronic Engineering, Universiti Putra Malaysia. His research interests include robotics, automation, and artificial intelligence in the fields of agriculture and biomedical engineering.