Journal of Harbin Engineering University
ISSN: 1006-7043

Vol 46 No. 1
January 2025

Improved YOLOv10 Deployment on Edge Devices for Tomato Grading

Jules Alexandre HABA[1,] *, Lawrence Chege Ngugi[2], John Adebisi[3]
[1]Pan African University, Institute for Basic Sciences Technology and Innovation (PAUSTI), Nairobi, Kenya
[2]Jomo Kenyatta University of Agriculture and Technology (JKUAT), Nairobi, Kenya
[3] University of Namibia, Windhoek, Namibia
*Corresponding author: jules.haba@students.jkuat.ac.ke

Abstract: Maintaining high tomato quality is crucial for consistent marketing standards and consumer satisfaction, necessitating the development of efficient automated tomato grading methods. Existing tomato grading systems are manual, labour-intensive, and time-consuming making them unsuitable for large-scale processing. Modern Deep Learning models are computationally expensive and not suitable for deployment on edges devices. Grading tomatoes is challenging because of the similarity in colour between some classes, such as unripe (green) and semi-ripe (yellow-green) tomatoes. This study proposes integrating a Convolutional Block Attention Module (CBAM) into YOLOv10 to improve feature extraction for enhanced accurate grading of tomatoes. Furthermore, the proposed model is deployed on an NVIDIA Jetson Nano board to demonstrate its performance on a resource-constrained edge device. For this study, a dataset of 7058 tomato images was collected from tomato farms in Kenya. The dataset represents tomatoes in four stages of maturity: unripe, semi-ripe, ripe, and damaged and the model was trained to distinguish between these categories. On the Personal Computer, the model achieved a mean Average Precision (mAP) of 73%, an inference speed of 38.31 frames per second (FPS), and a model size of 5.812 MB. In the Jetson board, it achieved a mAP of 72.3% with an inference speed of 25 FPS using 640x640 image resolution, which is better compared to the competitors.
Keywords: Convolutional Block Attention Module; Edge Devices; Precision Agriculture; Real-Time Image processing; Tomato grading; YOLOv10

1. Introduction

Agriculture profoundly impacts economic sustainability and promotes long-term development [1]. Tomatoes are essential for human nutrition and the agricultural economy [2]. With the growing human population, tomato production must be increased while maintaining high-quality standards. Existing tomato grading systems are manual, time-consuming, and labour-intensive. Moreover, they are prone to inaccuracies due to human errors and fatigue. On the other hand, classical image processing techniques, such as colour-based segmentation and edge detection, suffer from poor feature extraction and generalizability.

Modern Deep Learning (DL) algorithms for tomato grading face challenges [4,5], such as large model size, computational complexity, and low inference speed on edge devices. These challenges hinder their deployment on resource-constrained edge devices for real-time applications. Many studies [6,7] have shown that Convolutional Neural Networks (CNN) achieve high performance in fruit grading in Personal Computers (PCs). However, the performance decreases when tested in a resource-constrained edge device for real-time scenarios. This can be attributed to the large model size and high computational complexity [6].

You Only Look Once (YOLO) [7] and other object detection algorithms were initially developed as general-purpose object detectors. However, they have been customized for niche applications such as autonomous driving, facial recognition, agriculture, etc.… Researchers have trained the YOLO algorithms to recognize and grade fruits using field-captured images [3]. High-diversity datasets with varying lighting, posing, and background conditions are crucial for training and validating YOLO models.

Grading tomatoes is challenging because of the difficulties in accurately distinguishing similarities in colour between some classes, such as unripe (green) and semi-ripen (yellow-green) tomatoes. This could potentially reduce the overall performance of the

algorithm. Several studies [6,11] have highlighted the need to develop new models that are smaller, more accurate, and faster in inference speed for accurate grading of tomatoes. However, a tomato-grading algorithm that fully meets these requirements has not yet been developed.

To address these challenges, this study proposes the following main contributions: (1) The presentation of a custom dataset of 7,058 tomato well-annotated images with natural backgrounds collected directly from farms in Kenya using an RGB smartphone camera. The dataset categories tomatoes in four stages of maturity: unripe, semi-ripe, ripe, and damaged. (2) Development of an optimized lightweight YOLOv10 model for accurate grading of tomatoes: The use of Convolutional Block Attention Module (CBAM) in YOLOv10 backbone to tackle the difficulties in accurately recognizing similarity in colour between some classes such as unripe (green) and semi-ripe (yellow-green) tomatoes. The proposed improved algorithm is designed to be a lightweight architecture suitable for edge device implementation. (3) The proposed improved model is deployed on an NVIDIA Jetson Nano board to demonstrate its performance on a resource-constrained edge device. This is significant because such edge devices are best suited for deployment on sorting machines or agricultural robots. This work contributes to the goal of Agriculture 4.0, by providing a viable solution for the real-world grading of tomato systems.

The remainder of this paper is organized as follows: Section 2 presents the related work, Section 3 describes the methodology, Section 4 discusses the results, and Section 5 concludes the paper and highlights future work.

2. Related work

DL algorithms have been deployed in various agricultural domains, such as fruit recognition, harvesting, and grading [7,14]. The study by authors in [11] demonstrated that integrating DL models with the Internet of Things (IoT) can enhance fruit grading and reduce resource waste in agriculture. Over the past few years, significant progress has been made in object detection and image recognition algorithms. These advancements span from the Viola-Jones detector [12] to more sophisticated models, such as the Faster Region-Convolutional Neural Network (Faster R-CNN) [13], YOLO [7], and the Single-Shot Multibox Detector (SSD) [14].

Although the Faster R-CNN [13] algorithm introduced region proposal networks for enhanced accuracy, they were limited by low detection speed, making them unsuitable for real-time applications. SSD [14] algorithm achieved a balance between speed and accuracy but was constrained by its large model size, making it less suitable for resource-limited hardware. on the other hand, the YOLO [7] family of algorithm, employs a single-step detection process that reduces model size while maintaining high accuracy and speed, making it an ideal choice for real-time applications.

The YOLO family of algorithms [7,15-23] has revolutionized real-time object detection in various fields. You Only Look Once version five (YOLOv5) [22], You Only Look Once version eight (YOLOv8) [23], and You Only Look Once version ten (YOLOv10) [16] have emerged as suitable options for real-time applications due to their high detection accuracy, lightweight design, and faster inference speeds.

In the YOLOv5 [22] algorithm, the authors introduced the Cross Stage Partial Darknet *(*CSPDarknet) backbone and Mosaic Augmentation to enhance speed and accuracy. Its growing popularity is attributed to its modular design, which allows for the customization and export of trained models into multiple formats such as Open Neural Network Exchange *(*ONNX), and TensorFlow Lite (TFLite), simplifying deployment across diverse platforms [24]. However, it faces limitations, particularly in terms of relatively low inference speed.

YOLOv8 [23] algorithm, building on the success of YOLOv5 [22], incorporates enhanced feature extraction and anchor-free detection. This offers improved accuracy for small-object detection, a unified framework for diverse tasks, and faster inference speed for real-time applications, where rapid processing is important [25]. However, these improvements come at the cost of a larger model size than YOLOv5. The large model size limits its deployment in resource-constrained edge devices. These limitations open the door for the YOLOv10 algorithm.

In the YOLOv10 [16] algorithm, the authors introduced innovative methods to minimize computational complexity while achieving high

accuracy and faster inference speed than its predecessors. It introduces dual assignments for non-maximum suppression (NMS)-free training of YOLOs. This eliminates reliance on non-maximum suppression (NMS) for post-processing. This leads to a reduction in computational complexity and offers faster inference speed. This novel design makes YOLOv10 better-suited for edge device deployment than its predecessors. Table 1 presents a comparative analysis of the performance metrics for YOLOv5n (nano), YOLOv8n (nano), and YOLOv10n (nano) variants, evaluated on CPU using the COCO dataset with 640x640 image resolution. The nano variants are especially relevant to this study because they were developed with edge devices in mind.

Table 1. Performance Comparison of YOLOv5n, YOLOv8n, and YOLOv10n [16]

	mAP 50 (%)	Inference speed (FPS)	Model Size (MB)
YOLOv5	45.7	22.22	7.6
YOLOv8	37.3	162.33	12.21
YOLOv8	39.5	543.47	8.77

Several studies [25,26] have investigated the integration of the Convolutional Block Attention Module (CBAM) in YOLOv5 and YOLOv8. However, the resulting architectures are often too complex for deployment on resource-constrained edge devices. The models are large and slow. Despite this, CBAM has demonstrated significant capability in enhancing CNN accuracy performance. The YOLOv10 algorithm, one of the latest releases in the YOLO series, is more suitable for deployment on resource-constrained edge devices. This study proposes a new lightweight architecture integrating CBAM into the YOLOv10 backbone to address key factors making tomato grading more challenging.

Guan et al. [27] presented an improved YOLOv10 algorithm for small wheat ear detection and obstruction handling. Their approach integrated a Bidirectional Feature Pyramid Network, a separated and enhanced attention module, and a global context network into the YOLOv10 framework. It achieved an mAP of 95.10 % compared to the original YOLOv10, which had a mAP of 93.54%. However, their proposed algorithm was only evaluated on a Personal Computer (PC).

Qiu et al. [28] developed a Lightweight Detection YOLOv10 for drone object detection. Their approach integrated a Re-parameterized Grouped Extended Learning and Automation Network (RGELAN) and an Attention-Incorporated Feature Interaction (AIFI) module. This improved semantic information and small-object detection. It achieved an mAP of 39.4 %, a model size of 13.6 MB, and an inference speed of 25 FPS compared to the original YOLOv10, which had a mAP of 38.6%, a model size of 32.16 MB and an inference speed of 23 FPS. They evaluated their model on PC and Jetson Orin Nano, demonstrating its suitability for deployment on resource-constrained edge devices.

According to the literature, several challenges remain in the field of automated tomato grading. Grading tomatoes is particularly challenging due to the similarity in colour between certain classes, such as green (unripe) and yellow-green (semi-ripe) tomatoes. Furthermore, few studies have explored tomato grading using YOLOv10. This study proposes a solution for precise feature extraction of unripe and semi-ripe tomatoes, ensuring a small model size, high speed, and accuracy in real-time.

3. Methodology and materials

The proposed research methodology is presented in Figure 1.

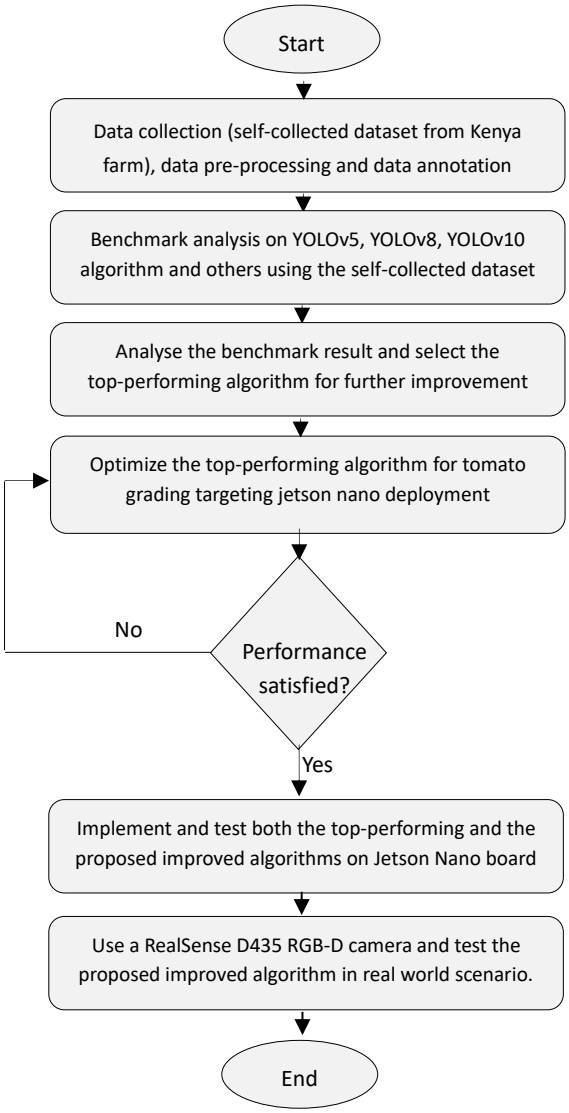


Figure 1. Detailed description of the proposed methodology used to conduct the research study

3.1. Dataset collection and preparation processes

For this study a dataset of 7,058 tomato images was collected from farms in Juja, Kiambu County, Kenya. High-resolution RGB smartphone cameras (Infinix X6851B, iPhone 13 Pro Max, Samsung Galaxy S10e) were used to capture images with natural backgrounds directly from the farm. This approach is critical, as training a CNN model solely on images with uniform backgrounds may limit its ability to generalise to real-world applications. All images were captured under various challenging conditions, including different lighting scenarios (morning, midday, and evening) and diverse backgrounds. They were taken at distances ranging from 0.5 to 1 meter to simulate practical use cases in robotic systems, such as automated harvesting and quality inspection.

The dataset represents tomatoes in four stages of maturity: unripe, semi-ripe, ripe, and damaged and the model is trained to distinguish between these categories. The composition of the self-collected dataset can be seen in Figure 1, which highlights the different stages of tomato maturation.

(a) (b) (c) (d)

Figure 2. Dataset classes: (a) unripe, (b) semi-ripe, (c) ripe and unripe, (d) unripe and damaged

After successfully collecting the images, all images were converted to JPG format and resized to 640×640 pixels. Redundant images were removed to enhance the dataset's effectiveness. Then, the dataset was split into three sets: 70% for training, 15% for validation, and 15% for testing. The training set images were not augmented.

After data preprocessing, the images were labeled using Label Studio software with four classes: unripe, semi-ripe, ripe, and damaged. The distribution of tomato classes exhibited a predominance of unripe tomatoes, exceeding 8,000. Ripe tomatoes numbered 4,400, semi-ripe tomatoes 2,300, and damaged tomatoes 1,600. The lower occurrence of damaged tomatoes is due to farmers' interventions before tomatoes reach this state. The class distributions after annotation are shown in Figure 3.

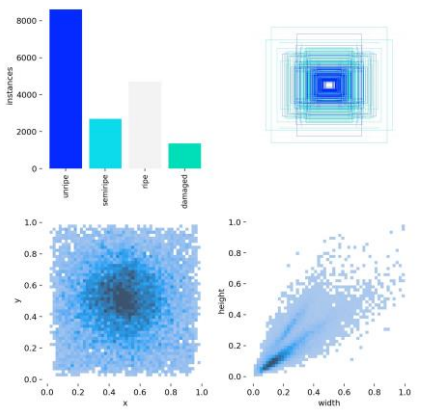


Figure 3. Tomato class distribution and bounding box analysis

3.2. Benchmark analysis of YOLOv5, YOLOv8, and YOLOv10 algorithms

The YOLOv5 [22], YOLOv8 [23], and YOLOv10 [16] are suitable for real-time applications due to their improved balance of inference speed, accuracy, and size. The question that can be asked is which algorithm is best for tomato grading under real-time conditions? To answer this question, a comparative analysis was conducted utilizing the self-collected tomato dataset. The hyperparameters used in the experiments are listed in Table 2.

Table 2. Training hyperparameters

Hyperparameters	Values
Variants	YOLOv5n.pt, YOLOv8n.pt, YOLOv10n.pt
Image size	640*640
Batch size	16
Epochs	1000
Initial learning rate	0.01
Optimizer	SGD
Weight Decay	0.0005
Momentum	0.937

During training evaluation and testing on the PC, this study used Anaconda Prompt, a command-line interface for Anaconda distribution, Python 3.9.19, PyTorch 2.4.0, and CUDA 11.8. To accelerate training and maintain consistency, the pre-trained YOLOv10n.pt, YOLOv8n.pt, and YOLOv5n.pt models were trained on the dataset using the transfer learning method. The number of epochs was set to 1000 with a patience threshold of 100, Stochastic Gradient Descent (SGD) was used as the optimizer, and the batch size was set to 16.

This comparative benchmark analysis aimed to identify a suitable model to form the basis for further enhancements. After comprehensive analysis and evaluation, the YOLOv10 [16] algorithm was identified as the best-performing model due to its optimal balance between mAP, size, inference speed, and classification accuracy. The results of these analyses are presented and discussed in Section 4.

3.3. Proposed detector network

The architecture of YOLOv10 is illustrated in Figure 4. It consists of three major components: Backbone, Neck and Head. The backbone consists of CBS (composed of Convolution, Batch Normalization, and SiLu activation functions), SCDown (Spatial-Channel decoupled downsampling module), C2f, C2fCIB, and PSA self-attention mechanisms. It extracts essential features from the input image (colours, size, and anomalies) and produces multi-scale feature maps [29]. The Neck module serves as a bridge between the feature extraction (backbone) and object detection (head) stages. It uses a Path Aggregation Network (PANet) layer to merge the features extracted by the backbone network and improves the capability of the model to detect objects of varying sizes, colours, and defects [27]. The Head layer predicts the object classes with both single-head and multi-head configurations. A single head generates the best prediction per object, whereas multiple heads provide several predictions [27].

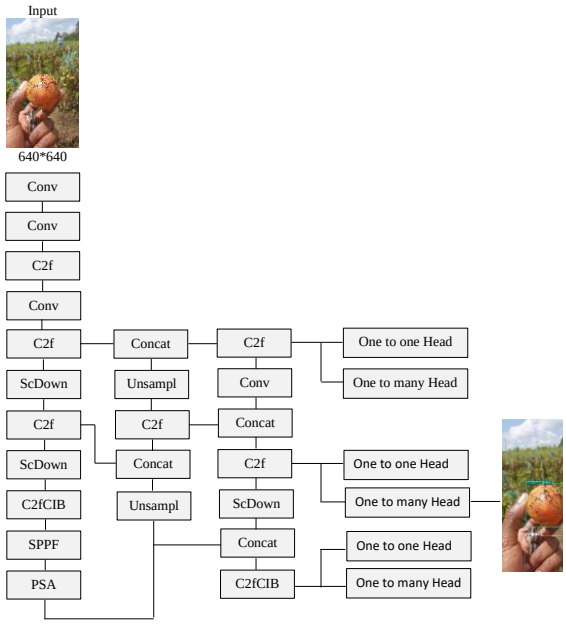


Figure 4. Architecture of original YOLOv10

This study proposes an improved YOLOv10 algorithm for accurate tomato grading under field conditions. CBAM [30] is used to enhance the performance of YOLOv10 for accurately distinguishing minute differences in colour between certain tomato classes, such as green (unripe) and yellow-green (semi-ripe) tomatoes.

CBAM is an attention mechanism module that improves CNN performance without increasing complexity. It has two main parts: channel attention and spatial attention modules. The channel attention module helps the model identify important features in the images, such as shapes, textures and colours, whereas the spatial attention module focuses on significant areas within the images (location). The structural diagram of the CBAM attention mechanism, is presented in Figure 5. σ is sigmoid function.

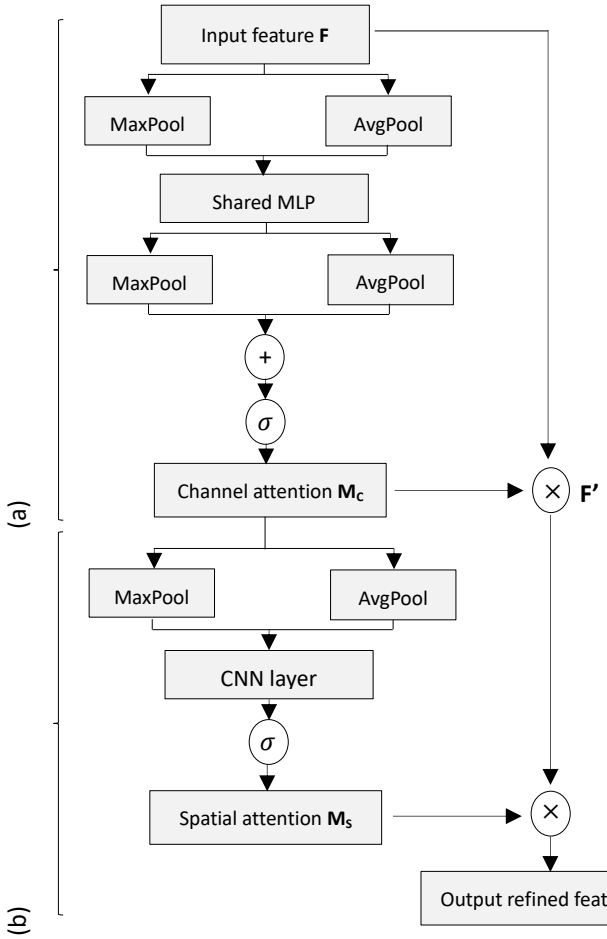


Figure 5. Structure diagram of CBAM attention mechanism: (a) Chanel Attention Module, and (b) Spatial Attention module.

The channel attention module enhances feature representation for each channel using average pooling (AvgPool), which reduces feature count and prevents overfitting. Average pooling suppresses noise while preserving colours and background details, whereas maximum pooling (MaxPool) eliminates redundant information and emphasises edge details. The Multilayer Perceptron (MLP) processes the pooled feature vectors, assigning channel weights through a sigmoid activation function. These weights are then multiplied with the original image features to adjust the importance of each channel.

The feature map is passed to the spatial attention module, which identifies important regions by applying average pooling and max pooling to the output of the channel attention module, generating two channels. These channels are then combined, compressed, and convolved with weights derived from the channel attention module. A sigmoid activation is applied to the result, creating spatial weights. Finally, these spatial weights are multiplied by the original image features to emphasize significant spatial details. The overall attention process can be summarized as follows: Equation 1 represents the Channel Attention process, whereas Equation 2 illustrates the Spatial Attention module.

$$F' = M_C(F) \times F \quad (1)$$

$$F'' = M_S(F') \times F' \quad (2)$$

The proposed method maintains the YOLOv10 neck and head intact. This study's approach consists of integrating CBAM into the YOLOv10 backbone. The architecture of the proposed algorithm is shown in Figure 6.

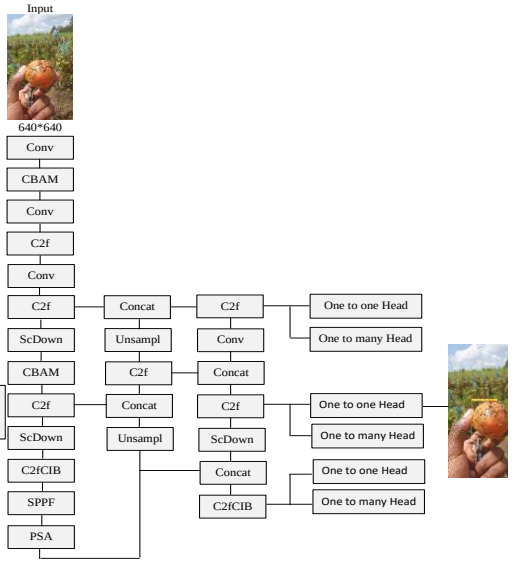


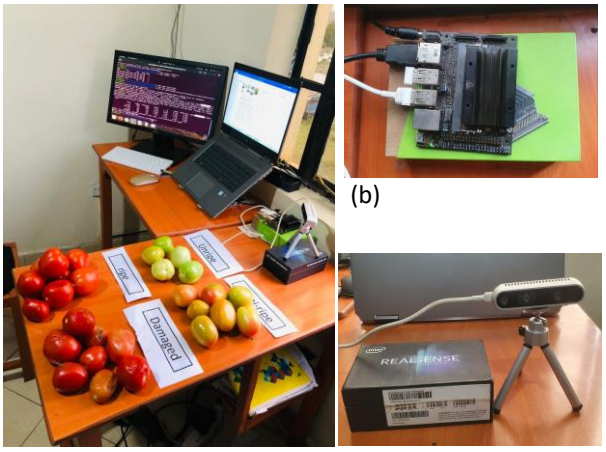
Figure 6. Architecture of the proposed improved YOLOv10

The first Convolutional layer in original YOLOv10 backbone (Conv1) initiates the model's ability to identify key features such as colour, shape or edges. Grading of tomatoes relies on colour and physical defects. However, the similarity in colour between some classes such as green (unripe) and yellow-green (semi-ripe) tomatoes, makes the detection process more challenging, increasing the likelihood of misclassification. Therefore, it is crucial to prioritise colour feature extraction early in the model. Integrating CBAM [30] after the Conv1 could enhance the model's ability to accurately grade tomatoes by improving feature attention.

The first Spatial-Channel decoupled downsampling layer in the original YOLOv10 backbone (SCDown1) adjusts channel dimensions using pointwise convolution and depthwise convolution for spatial downsampling to reduce parameter count. However, experiments indicate that some information related to features is lost during the downsampling process [31]. Given the importance of colour features in improving model performance, integrating CBAM could help the model retain colour-related information more effectively.

3.4. Deployment on Jetson nano environment

The real-time test setup, as shown in Figure 7, comprised a RealSense D435 RGB-D camera, HP ZBOOK Studio 8360 G5 (16 GB RAM, Intel Core i7, 9th Gen, 500 GB SSD, and Nvidia Quadro P1000 4 GB GPU), and NVIDIA Jetson Nano kit (4 GB RAM, 128 core Maxwell GPU, LINUX OS). Tomatoes in four stages of maturity: unripe, semi-ripe, ripe, and damaged were used.



(a) (b) (c)

Figure 7. Experimentation setup: (a) real-time testing environment (b) Nvidia jetson nano kit, (c) RealSense D435 RGB-D camera

3.5. Performance evaluation metrics

The performance metrics used in this study are: mean Average Precision (mAP) at a 0.5 IoU threshold, inference speed in FPS, model size in MB, and confusion matrix. The mAP combines precision and recall to calculate the models' detection accuracy. The weight file size measured the model size. The inference speed evaluates the detection speed of the network. The confusion matrix shows the classification accuracy for each class (unripe, semi-ripe, ripe, and damaged) across the model. Equations (3) to (7) represent precision (P), recall (R), Average Precision (AP), mAP, and inference speed, respectively. (n) denotes the total number of tomatoes in the test set.

$$P\ (\%) = \frac{TP}{TP + FP} * 100 \quad (3)$$

$$R\ (\%) = \frac{TP}{TP + FN} * 100 \quad (4)$$

$$AP\ (\%) = \int_0^1 P(R)d(R) \quad (5)$$

$$mAP = \frac{\sum_{i=1}^{n} AP_i}{n} \quad (6)$$

$$Inference\ speed = \frac{1000}{Latency\ (ms)} \quad (7)$$

4. Results and discussion

This section presents the evaluation results of the trained models in the two environments: (1) the Personal Computer (PC) and (2) the Jetson Nano board.

4.1. Performance comparison on PC environment

This section discusses the performance of YOLOv5 [22], YOLOv5-CBAM [26], YOLOv8 [23], YOLOv8-CBAM [25], YOLOv10 [16], and the proposed improved YOLOv10 using the self-collected tomato dataset on a PC. Figure 8 shows that the training mAP curves for all six models converged around 150 epochs.

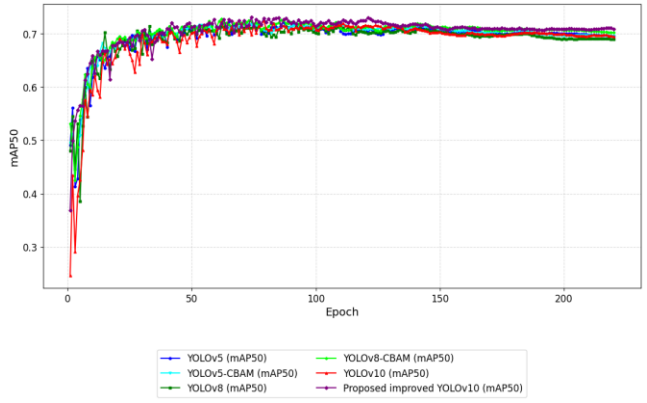


Figure 8. Training mAP curves comparison

After successful training and validation, all six models were tested on 15% of the dataset. Table 3 presents the mAP (detection accuracy), inference speed, and model size of the six models during testing.

Table 3. Performance comparison across the six models on PC using the self-collected tomato dataset.

	mAP 50 (%)	Inference speed (FPS)	Model Size (MB)
YOLOv5 [22]	72.2	23.25	5.828
YOLOv5-CBAM [26]	72.6	25	8.905
YOLOv8 [23]	71.2	28.62	6.589
YOLOv8-CBAM [25]	69.9	31.70	10.756
YOLOv10 [16]	72.5	32.05	**5.602**
Proposed improved YOLOv10	**73**	**37.31**	5.812

Figures 10 to 15 show the confusion matrices (classification accuracy) of YOLOv5 [22], YOLOv5-CBAM [26], YOLOv8 [23], YOLOv8-CBAM [25], YOLOv10 [16], and the proposed improved YOLOv10.

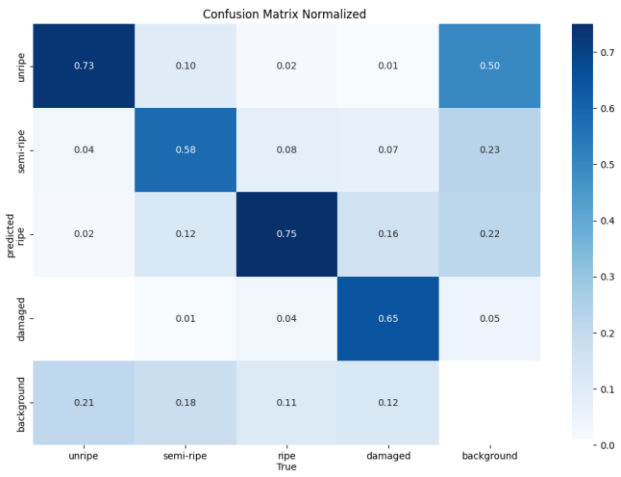


Figure 9. YOLOv5 confusion matrix normalized

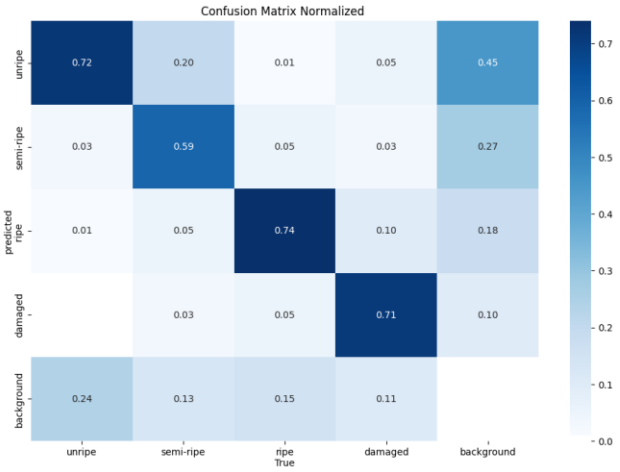


Figure 10. YOLOv5-CBAM confusion matrix normalized

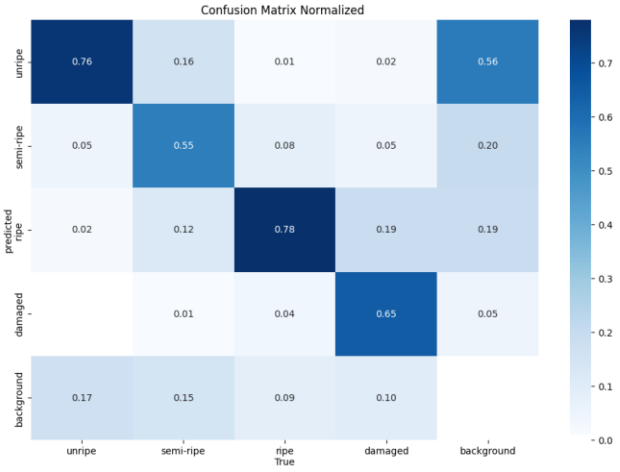


Figure 11. YOLOv8 confusion matrix normalized

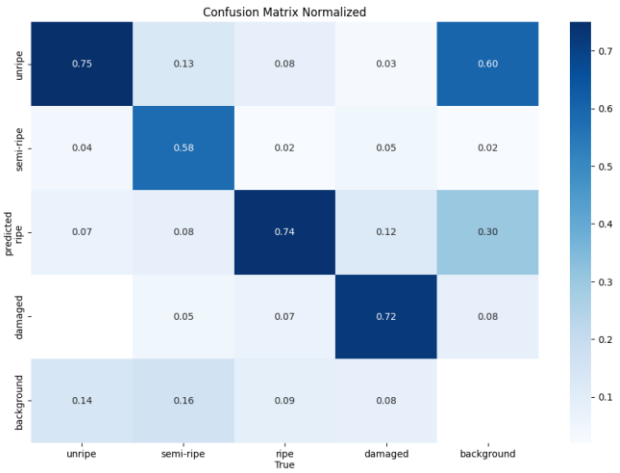


Figure 12. YOLOv8-CBAM confusion matrix normalized

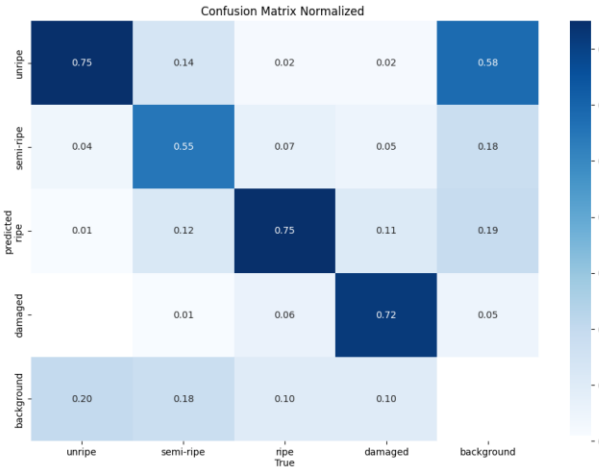


Figure 13. YOLOv10 confusion matrix normalized

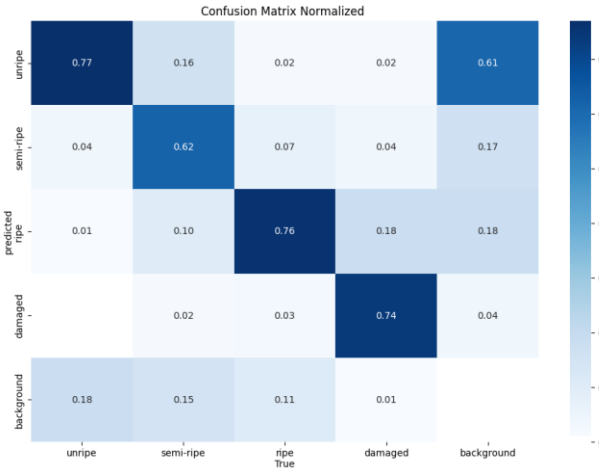


Figure 14. Improved proposed YOLOv10 confusion matrix normalized

The proposed improved YOLOv10 model achieved an enhanced output mAP of 73% compared to the original YOLOv10 (72.5%), YOLOv5 (72.2%), YOLOv8 (71.2%), YOLOv5-CBAM (72.6) and YOLOv8-CBAM (69.9%). These improvements can be attributed to the integration of CBAM in the YOLOv10 backbone. This indicates that the CBAM block enhances the CNN's feature extraction capability. This agrees with the previous studies [25,26] in which CBAM improved the performance of the YOLOv5 and YOLOv8 networks. However, their method involved integrating CBAM into the neck or building a bridge between the backbone and the neck using CBAM. This study provides a simple architecture compared to those models. The novelty in this study is the finding that the CBAM can be used after the first Convolution layer and the first Spatial-Channel decoupled downsampling layer in the Conventional YOLOv10 backbone.

In terms of speed, the proposed improved YOLOv10 has a faster inference speed of 37.31 FPS compared to its competitors, YOLOv5 (23.25 FPS), YOLOv5-CBAM (25 FPS), YOLOv8 (28.62 FPS), YOLOv8-CBAM (31.7 FPS) and YOLOv10 (32.05 FPS). This increase in inference speed compared to its competitors can be attributed to proposed improved feature extraction, which allows faster processing in accurately recognizing similarity in colour between some classes, such as green (unripe) and yellow-green (semi-ripe) tomatoes of input images without sacrificing accuracy. Further, the conventional YOLOv10 is already lightweight because of its efficient network architecture, which uses dual assignments to eliminate the need for NMS training, reducing inference speed. These design choices reduce the number of parameters and computational complexity while maintaining high performance. The proposed improved YOLOv10 builds upon this foundation by optimizing the feature extraction process, resulting in even faster inference speeds without compromising accuracy. The proposed improved YOLOv10 is more suitable for real-time applications in edge devices where rapid execution is crucial.

The proposed improved YOLOv10 model has a slightly larger size of 5.812 MB compared to the original YOLOv10 with 5.602 MB. This small size difference can be attributed to the integration of CBAM. But it remains more compact than the YOLOv5 with 5.828 MB, YOLOv5-CBAM with 8.905 MB, YOLOv8 with 6.089 MB, and YOLOv8-CBAM with 10.756 MB. The reason why the proposed improved YOLOv10 algorithm is small in size compared to YOLOv5-CBAM and YOLOv80-CBAM is because of integration techniques used to enhance YOLOv10 feature extraction. The previous study in [26] used three blocks of CBAM in the YOLOv5 neck for precise localization. Another study in [25] uses three blocks of CBAM as a bridge between the YOLOv8 backbone and neck to address the complexity in accurately recognizing the difference between fire and smoke. This study uses the two CBAMs in the YOLOv10 backbone to address the difficulties in accurately recognizing the difference between unripe and semi-ripe tomatoes among all tomato maturity stages.

The normalized confusion matrix for each model is presented in Figures 9-14. The proposed improved YOLOv10 shows the highest balanced classification accuracy across all categories in distinguishing

between Unripe, Semi-ripe, Ripe, and Damaged tomatoes. It achieves 77% accuracy for Unripe, 62% for Semi-ripe, 76% for Ripe, and 74% for damaged tomatoes, showing notable improvements over the other algorithms. The improvement in semi-ripe classification (62%) is particularly valuable, because semi-ripe (yellow-green) tomatoes often show visual characteristics overlapping with unripe (green) and ripe (red) categories. This valuable improvement compared to its competitors using the self-collected dataset shows the effectiveness of the proposed algorithm.

The proposed improved YOLOv10 model demonstrates superior performance in mAP, inference speed, and classification accuracy for tomato grading. The results confirm that the proposed model is viable for real-time agricultural applications.

4.2. Performance comparison on Jetson nano board

The performance of YOLOv10 and the proposed improved YOLOv10 on the jetson nano board are discussed in this section. Further, the objective is to evaluate their performance on a resource-constrained edge device. A Linux operating system was used to set up the Jetson environment. The software environment included Ubuntu 20.04, CUDA 10.2, cuDNN 8.2.1, Python 3.9.0, JetPack 32.6.1, and TensorRT 8.0.1. Table 4 presents the mAP, and inference speed of the conventional YOLOv10 and the proposed improved YOLOv10 in the Jetson Nano platform

Table 4. Performance comparison between conventional YOLOv10 and the proposed improved YOLOv10

	mAP 50 (%)	Inference speed (FPS)
YOLOv10	71.8	19.93
Proposed improved YOLOv10	**72.3**	**25**

Figures 16 and 17 show the confusion matrices generated during testing on the Jetson Nano board for the original YOLOv10 and the proposed improved YOLOv10, respectively.

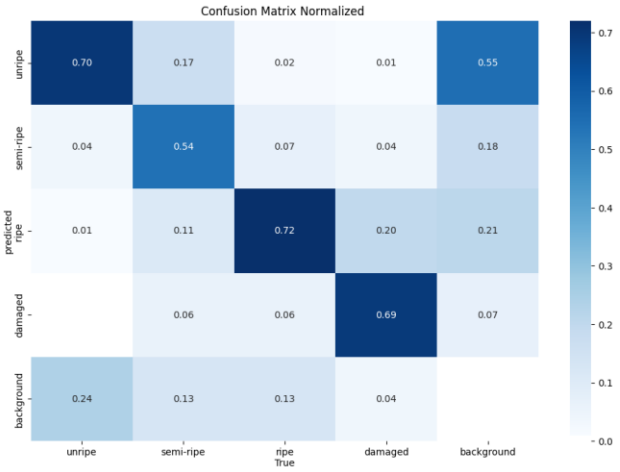


Figure 15. YOLOv10 confusion matrix normalized

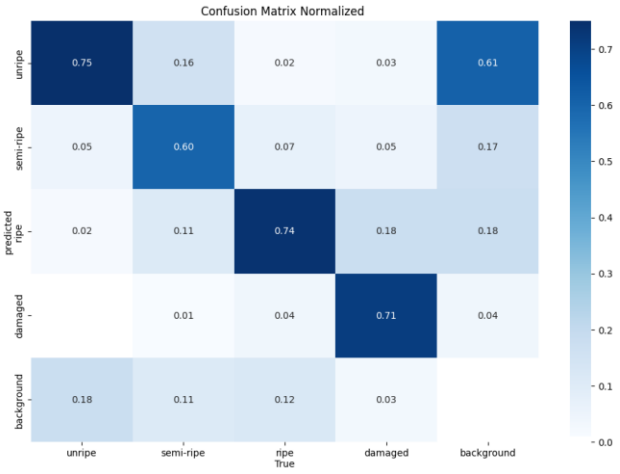


Figure 16. Improved proposed YOLOv10 confusion matrix normalized

Figure 17 presents a comparative analysis of the predicted images obtained during testing of YOLOv10 and the proposed improved algorithm using the test dataset on jetson nano platform.

(a) (b)

Figure 17. Detection performance of (a) YOLOv10 and (b) the proposed algorithm.

The proposed improved YOLOv10 achieved an improved mAP of 72.3% on the Jetson Nano board compared to the original YOLOv10, which achieved 71.8%. A slight reduction in mAP values is observed compared to the performance in the PC environment. This can be attributed to the Jetson nano board's limited memory (only 4 GB of RAM), processing power capabilities, and thermal constraints. The reduced computational resources on the Jetson Nano board impact the model's ability to process complex features and maintain high mAP. Despite this limitation, the proposed improved YOLOv10 outperforms the original YOLOv10 algorithm.

The proposed improved YOLOv10 achieved an inference speed of 25 FPS, outperforming YOLOv10, which achieved 19.93 FPS. An algorithm is unsuitable for real-time applications when the inference speed is less than 20 FPS [32]. Therefore, the proposed model satisfied the requirements for real-time performance. However, a drop in performance is observed in the Jetson nano board compared to the performance on the PC. This can be attributed to the limited computational resources available on the Jetson Nano compared to a high-performance PC. Despite this performance drop, the improved YOLOv10 maintains acceptable real-time performance on the embedded platform.

The proposed improved YOLOv10 demonstrates highly balanced classification accuracy (confusion matrix) across all tomato categories in distinguishing between Unripe, Semi-ripe, Ripe, and Damaged tomatoes. It achieves 75% accuracy for Unripe, 60% for Semi-ripe, 74% for Ripe, and 71% for Damaged tomatoes, showing notable improvements over the original YOLOv10. The improved proposed YOLOv10 provided an improvement of 3.8 % in overall classification accuracy in the jetson nano board. In particular, the semi-ripe category had a 6% improvement in classification accuracy. This demonstrates that the proposed improved YOLOv10 is more efficient than its competitors. This new algorithm can help farmers become efficient and productive.

As shown in Figure 18, the proposed improved algorithm is able to detect occluded tomatoes accurately, demonstrating its enhanced capability in handling challenging scenarios compared to conventional YOLOv10. This advancement is particularly beneficial for real-world applications where tomatoes may be partially hidden or overlapping, such as in dense foliage or crowded harvesting bins. The model's ability to accurately identify and localize tomatoes even when they are occluded showcases its robustness and adaptability to complex environments. This feature is especially valuable in automated harvesting systems, where precisely detecting partially obscured fruits is essential for efficient picking. Furthermore, the model's superior detection of small tomatoes indicates its increased sensitivity to fine details, which is crucial for accurate grading across various tomato sizes and growth stages. This improved capability allows for more precise monitoring of crop development, enabling farmers and researchers to track the growth progression of individual fruits from early formation to full maturity. The enhanced detection of small tomatoes also facilitates early identification of potential issues, such as pest infestations or nutrient deficiencies, which may affect fruit development.

The proposed model improves detection accuracy, classification accuracy, rapid processing performance, low computation demand, occlusion handling, and small object detection, which contribute significantly to tomato grading systems' overall accuracy and reliability. These improvements can lead to more efficient sorting and grading processes in post-harvest operations, reducing labour costs and minimizing errors in quality control.

4.3. Real-word testing of the proposed model on Jetson

Nano platform

The real-time testing environment presents several challenges, including a background different from what is in the dataset, the motion of the object during image capture, and camera latency. Differences in background will severely test the model's ability to generalize to new scenes. The target object's (tomato's) movement can introduce motion blur into the images. Whereas this can be countered by increasing the shutter speed of a camera, it in turn reduces the amount of light falling on the camera's complementary metal-oxide semiconductor (CMOS) sensor. The resulting images would appear darker than expected, thereby obscuring important features. The latency introduced by the camera, being the picture acquisition time and time taken to transmit the image to the Jetson, could reduce the system's frame rate.

Moreover, other factors also influence the performance of tomato detection systems in challenging image conditions. When tomatoes occupy a small area in the image, detection accuracy may decrease due to limited visual information and reduced distinguishing features. This can lead to misclassifications or missed detections, especially if the model was not specifically trained on small-scale objects. In scenarios where multiple tomatoes are clustered together, the system may struggle to differentiate and classify individual fruits accurately. This can result in merged detections or incorrect count estimations, potentially affecting the overall performance of the detection algorithm. Dark or poorly lit images present another challenge, as insufficient lighting can obscure important visual cues, making it difficult for the model to identify and classify tomatoes accurately. This may increase false negatives or misclassifications, particularly for tomatoes with subtle colour variations or those in shadow areas. Despite these challenges, the proposed improved YOLOv10 performs well in these scenarios, indicating its robustness and effectiveness in handling diverse and complex image conditions. Figure 18 shows that the detection system using a RealSense D435 RGB-D camera performs well when dealing with the aforementioned challenging image conditions

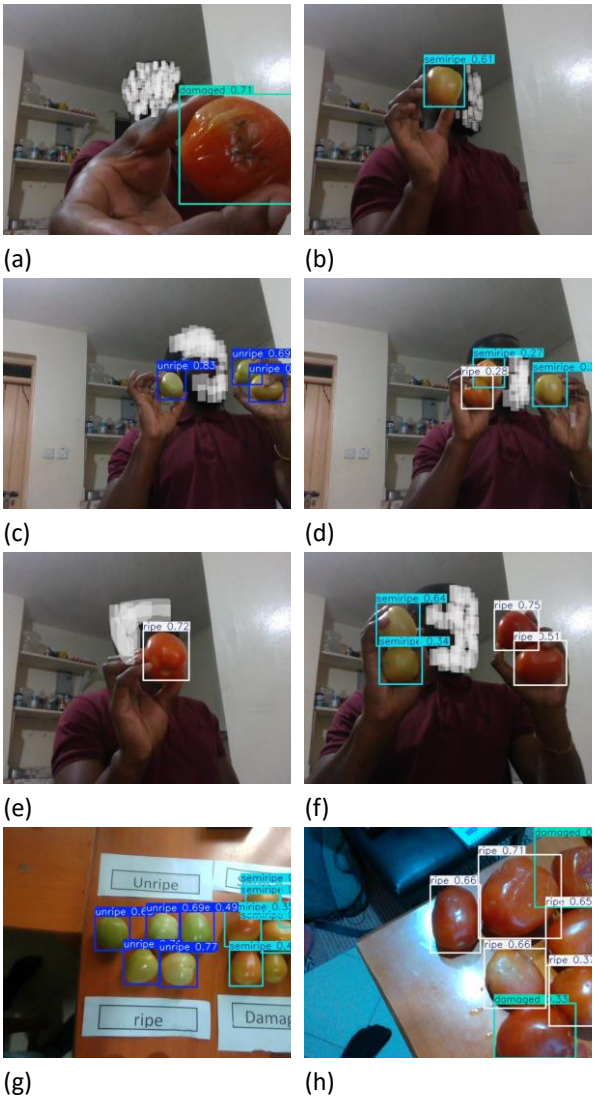


Figure 18. Real-time tomato detection results: (a), (b), (e) tomatoes occupying small areas in the image; (c), (d), (f) (g), (h) clustered tomatoes; (h), (f) tomatoes in dark conditions.

5.

6. Conclusion

This study presents a real-time system based on a deep-learning algorithm for grading tomatoes in real-world applications. The results obtained in this study are novel for several reasons: (1) The presentation of a dataset of 7,058 tomato images with natural backgrounds collected directly from farms in Kenya using an RGB smartphone camera. The dataset represents tomatoes in four stages of maturity: unripe, semi-ripe, ripe, and damaged. (2) The development of a new optimized lightweight YOLOv10 model for accurately grading tomatoes. This is achieved by integrating a CBAM block in the backbone of the original YOLOv10 model. The proposed improved algorithm tackles the challenges

facing the actual tomato grading system. It improves accuracy in distinguishing slight differences in colour between unripe (green) and semi-ripe (yellow-green) tomatoes in both the PC environment and jetson nano board. (3) The proposed improved model was deployed on an NVIDIA Jetson Nano board to demonstrate its performance on a resource-constrained edge device. This is significant because such edge devices are best suited for deployment on sorting machines or agricultural robots. Such advancements are crucial towards meeting the goal of Agriculture 4.0.

The proposed enhanced algorithm is designed to address quality control, market efficiency, waste reduction, consumer health and safety, supply chain management, and large-scale agricultural practices. It was evaluated on the PC and an NVIDIA Jetson Nano Kit. On the PC, the model achieved a mAP of 73%, an inference speed of 37.31 FPS, and a memory usage of 5.812 MB. The Jetson achieved a mAP of 72.3% with an inference speed of 25 FPS using 640x640 image resolution, which is better than the competitors.

Despite these improvements, the live test results indicated decreased model performance. This can be attributed to several factors arising in the live test scenario, such as challenging lighting conditions, the tomato's orientation pose to the camera, and the quality of the camera used for real-time testing. Future studies should address these challenges by collecting larger datasets for real-world field conditions to capture more poses under diverse lighting conditions and propose further enhancements to the detection model.

Acknowledgment

This work was funded and supported by the African Union Commission and Pan African University Institute for Basic Sciences, Technology and Innovation (PAUSTI).

References

[1] A. DEVLET, “Modern agriculture and challenges,” *Front. Life Sci. Relat. Technol.*, vol. 2, no. 1, pp. 21–29, 2021, doi: 10.51753/flsrt.856349.

[2] Food and Agriculture Organization of the United Nations, *The State of Food and Agriculture - Revealing the true cost of food to transform agrifood systems*. 2023.

[3] W. Zhang, “Automated Fruit Grading in Precise Agriculture using You Only Look Once Algorithm,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 10, pp. 1136–1144, 2023, doi: 10.14569/IJACSA.2023.01410119.

[4] Z. Zhou *et al.*, “Advancement in artificial intelligence for on-farm fruit sorting and transportation,” *Front. Plant Sci.*, vol. 14, no. April, pp. 1–11, 2023, doi: 10.3389/fpls.2023.1082860.

[5] H. S. Mputu, A. Abdel-mawgood, and A. Shimada, “Tomato Quality Classification Based on Transfer Learning Feature Extraction and Machine Learning Algorithm Classifiers,” *IEEE Access*, vol. 12, no. January, pp. 8283–8295, 2024, doi: 10.1109/ACCESS.2024.3352745.

[6] R. G. de Luna, E. P. Dadios, A. A. Bandala, and R. R. P. Vicerra, “Tomato growth stage monitoring for smart farm using deep transfer learning with machine learning-based maturity grading,” *Agrivita*, vol. 42, no. 1, pp. 24–36, 2020, doi: 10.17503/agrivita.v42i1.2499.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.

[8] V. Meshram, K. Patil, V. Meshram, D. Hanchate, and S. D. Ramkteke, “Machine learning in agriculture domain: A state-of-art survey,” *Artif. Intell. Life Sci.*, vol. 1, no. August, p. 100010, 2021, doi: 10.1016/j.ailsci.2021.100010.

[9] V. G. Dhanya *et al.*, “Deep learning based computer vision approaches for smart agricultural applications,” *Artif. Intell. Agric.*, vol. 6, pp. 211–229, 2022, doi: 10.1016/j.aiia.2022.09.007.

[10] M. Wakchaure, B. K. Patle, and A. K. Mahindrakar, “Application of AI techniques and robotics in agriculture: A review,” *Artif. Intell. Life Sci.*, vol. 3, no. January, p. 100057, 2023, doi: 10.1016/j.ailsci.2023.100057.

[11] E. M. B. M. Karunathilake, A. T. Le, S. Heo, Y. S. Chung, and S. Mansoor, “The Path to Smart Farming: Innovations and Opportunities in Precision Agriculture,” *Agric.*, vol. 13, no. 8, pp. 1–26, 2023, doi: 10.3390/agriculture13081593.

[12] P. Viola and M. Jones, “Managing work role performance: Challenges for twenty-first century organizations and their employees.,” *Rapid*

Object Detect. using a Boost. Cascade Simple Featur., pp. 511–518, 2001.

[13] R. Girshick, "Fast R-CNN," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 Inter, pp. 1440–1448, 2015, doi: 10.1109/ICCV.2015.169.

[14] Y. Konishi, Y. Hanzawa, M. Kawade, and M. Hashimoto, "Fast 6D Pose Estimation Using Hierarchical Pose Trees," *Eccv*, vol. 1, no. January 2017, pp. 398–413, 2016, doi: 10.1007/978-3-319-46448-0.

[15] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," no. May, 2020, [Online]. Available: http://arxiv.org/abs/2004.10934

[16] A. Wang *et al.*, "YOLOv10: Real-Time End-to-End Object Detection," pp. 1–18, 2024, [Online]. Available: http://arxiv.org/abs/2405.14458

[17] J. Redmon, "YOLOv3: An Incremental Improvement".

[18] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information," 2024, [Online]. Available: http://arxiv.org/abs/2402.13616

[19] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," pp. 7464–7475, 2023, doi: 10.1109/cvpr52729.2023.00721.

[20] C. Li *et al.*, "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," 2022, [Online]. Available: http://arxiv.org/abs/2209.02976

[21] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 6517–6525, 2017, doi: 10.1109/CVPR.2017.690.

[22] Glenn Jocher et al., "Yolov5," Ultralytics. [Online]. Available: https://github.com/ultralytics/yolov5

[23] G. J. et al. 2023, "YOLOv8," ultralytics. [Online]. Available: https://github.com/ultralytics/ultralytics

[24] C. V Jul, C. Science, and H. Hd, "YOLO V 5 , YOLO V 8 AND YOLO V 10 : T HE G O -T O D ETECTORS FOR R EAL - TIME V ISION," pp. 1–12, 2024.

[25] Z. Liu, R. Zhang, H. Zhong, and Y. Sun, "YOLOv8 for Fire and Smoke Recognition Algorithm Integrated with the Convolutional Block Attention Module," *Open J. Appl. Sci.*, vol. 14, no. 01, pp. 159–170, 2024, doi: 10.4236/ojapps.2024.141012.

[26] M. Lv and W. H. Su, "YOLOV5-CBAM-C3TR: an optimized model based on transformer module and attention mechanism for apple leaf disease detection," *Front. Plant Sci.*, vol. 14, no. January, pp. 1–13, 2023, doi: 10.3389/fpls.2023.1323301.

[27] Z. Li, Y. Zhu, S. Sui, Y. Zhao, P. Liu, and X. Li, "Real-time detection and counting of wheat ears based on improved YOLOv7," *Comput. Electron. Agric.*, vol. 218, pp. 1–19, 2024, doi: 10.1016/j.compag.2024.108670.

[28] X. Qiu, Y. Chen, W. Cai, M. Niu, and J. Li, "LD-YOLOv10: A Lightweight Target Detection Algorithm for Drone Scenarios Based on YOLOv10," *Electron.*, vol. 13, no. 16, 2024, doi: 10.3390/electronics13163269.

[29] G. Yang, J. Wang, Z. Nie, H. Yang, and S. Yu, "A Lightweight YOLOv8 Tomato Detection Algorithm Combining Feature Enhancement and Attention," *Agronomy*, vol. 13, no. 7, 2023, doi: 10.3390/agronomy13071824.

[30] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11211 LNCS, pp. 3–19, 2018, doi: 10.1007/978-3-030-01234-2_1.

[31] L. Tan, S. Liu, J. Gao, X. Liu, L. Chu, and H. Jiang, "Enhanced Self-Checkout System for Retail Based on Improved YOLOv10," *arXiv Prepr. arXiv*, 2024, [Online]. Available: http://arxiv.org/abs/2407.21308

[32] Z. Jiao, K. Huang, Q. Wang, Z. Zhong, and Y. Cai, "Real-time litchi detection in complex orchard environments: A portable, low-energy edge computing approach for enhanced automated harvesting," *Artif. Intell. Agric.*, vol. 11, pp. 13–22, 2024, doi: 10.1016/j.aiia.2023.12.002.