

Article

A Lightweight Deep Learning Framework for Real-Time Brinjal Detection Under Field Conditions

Abhishek Pandey ¹, Pramod Kumar Sahoo ^{1,*}, Tapan Kumar Khura ¹, Dilip Kumar Kushwaha ^{1,*}, Roaf Ahmad Parray ¹, Jeetendra Kumar Ranjan ², Md. Ashraful Haque ³, Susheel Kumar Sarkar ⁴, Rohit Gaddamwar ¹ and Nrusingh Charan Pradhan ^{1,5,*}

¹ Division of Agricultural Engineering, ICAR-Indian Agricultural Research Institute, New Delhi 110012, India; iariabhishekfmp@gmail.com (A.P.); tapankhura2020@gmail.com (T.K.K.); rouf.engg@gmail.com (R.A.P.); grohit11.rg@gmail.com (R.G.)

² Division of Vegetable Science, ICAR-Indian Agricultural Research Institute, New Delhi 110012, India; jkranjan2001@yahoo.co.in

³ Division of Computer Application, ICAR-Indian Agricultural Statistics Research Institute, New Delhi 110012, India; ashrafulhaque.iasri@icar.org.in

⁴ Division of Design of Experiments, ICAR-Indian Agricultural Statistics Research Institute, New Delhi 110012, India; susheel.sarkar@icar.org.in

⁵ Department of Renewable Energy Engineering, College of Agricultural Engineering and Technology, Odisha University of Agriculture and Technology, Bhubaneswar 751003, India

* Correspondence: sahoopk1965@gmail.com (P.K.S.); sddkushwaha@gmail.com (D.K.K.); ncpradhan@ouat.ac.in (N.C.P.)

Abstract

Brinjal (*Solanum melongena* L.) is an important vegetable crop worldwide, but its cultivation faces challenges from pests, diseases, and variable environmental conditions that negatively affect quality and yield. Accurate fruit detection in natural field conditions is essential for yield estimation and perception module of automated harvesting, but existing deep learning techniques often require substantial computational support, restricting their deployment on edge devices. This study addresses this gap by evaluating the Faster Objects, More Objects (FOMO) model—a lightweight architecture for resource-constrained platforms—for brinjal detection under diverse field conditions. A dataset of 1500 images was captured under varying illumination and growth stages, annotated using a bounding box-based approach, and used to train FOMO models with 25, 50, and 100 epochs via transfer learning. Post-training quantization to INT8 format was applied to assess improvements in computational efficiency. The Float32 model achieved a precision of 0.827, recall of 0.915, and F1-score of 0.869 at 100 epochs. The INT8-quantized model maintained comparable accuracy (precision 0.829, recall 0.908, F1-score 0.866) while reducing model size by 63.33% (from 0.30 MB to 0.11 MB), inference time by 62.02% (from 65.3 ms to 24.8 ms), and RAM usage by 73.02% (from 887.2 KB to 239.4 KB). These results demonstrate that FOMO combined with INT8 quantization provides an efficient, accurate solution for real-time brinjal detection on edge platforms, supporting the advancement of precision agriculture through intelligent crop monitoring and serving as a perception module for future robotic harvesting systems.

Keywords: brinjal; automated agriculture; real-time detection; lightweight architecture; quantization

Academic Editors: Shuai (Steven) Li, Qingting Liu, Tao Wu and Zhigang Zhang

Received: 22 May 2026

Revised: 6 July 2026

Accepted: 23 July 2026

Published: 1 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Harvesting of horticulture produce presents significant challenges due to non-uniformity in maturity cycle, making it labour-intensive and requiring frequent intervention. Manual harvesting of horticultural crops (fruits, vegetables, and flowers) is highly labour-intensive, often requiring between 50% and 80% of the total labour hours and about one-third to half of the total production expenses for the entire production cycle. Timely harvesting of horticultural crops is essential, as delays result in over-ripening/over-maturity, which leads to a consistently extensive labour requirement. Due to the recently faced shortage of labour in agricultural operations, excess work hours in such physically draining tasks that are marked by ergonomic strain and repetitive motion, contribute to documented health hazards for agricultural workers, including elevated rates of musculoskeletal disorders, heat stress, dermatitis, infections, and occupational injuries. With the globally escalating horticultural produce needs, and in spite of such constraints, there is a requirement for development in technological advancement and automation in the harvesting field, such as fruit detection and localization technologies and robotic harvesters, with the perspective of improving operational precision, resource management, and produce quality.

Brinjal (*Solanum melongena* L.), also known as eggplant, is an economically valuable vegetable commodity that is produced on a global scale. As of 2023, global brinjal production reached approximately 59.3 M million tonnes, with China, India, and Egypt being the leading producers [1]. In India, brinjal is the third-largest vegetable crop in terms of production, cultivated over approximately 0.7 million hectares with an annual production of about 13.4 million tonnes [2]. Besides its economic importance, brinjal is nutritionally valuable due to its rich content of vitamins, minerals, and dietary fibre, contributing significantly to food security and nutritional diversity in many regions [3]. However, brinjal harvesting faces several challenges adversely affecting crop yield and quality [4,5]. Accurate detection and monitoring of brinjal fruits in field environments can support yield estimation, optimized harvesting schedules, and automated harvesting systems, thereby improving agricultural productivity [6].

The accelerated advancement in computer vision technologies and artificial intelligence (AI) modern agriculture is increasingly adopting automated solutions to enhance productivity and efficiency [7]. Deep learning-assisted object recognition and detection techniques have become impactful tools for identifying crops, weeds, pests, and diseases in complex agricultural environments. Traditional agricultural monitoring methods often depend on manual inspections, which are time-consuming, manpower-intensive, and sensitive to human error [8]. The adoption of computer vision systems can significantly improve the accuracy and efficiency of agricultural monitoring tasks [9].

Recent advances in deep learning-based object detection, particularly Convolutional Neural Network (CNN) models, have shown promising results in agricultural applications [10,11]. Models such as Mask R-CNN, Faster R-CNN, and YOLO have been widely used for fruit finding in natural, complex agricultural conditions. Zu et al. [12] reported an F1-score of 92% using Mask R-CNN for tomato detection, while Chen et al. [13] utilizing Faster R-CNN with ResNet-50, achieved an average precision of 0.83 for strawberry detection. Adoption of YOLO-based models has also been extensively explored due to their fast inference capability [14], with improved variants such as YOLO-Tomato, YOLOv3-Litchi, and SE-YOLOv5 achieving recognition accuracies above 87–94% [15–17]. However, these models typically require large architectures and high computational resources, limiting their deployment on edge devices used in agricultural automation. To address this limitation, lightweight models such as FOMO (Faster Objects, More Objects) have been evaluated for efficient real-time inference. FOMO has demonstrated strong performance on embedded platforms, achieving up to 60 FPS on Raspberry Pi [18], and has

been applied to applications including pothole detection, water-level monitoring, mosquito egg detection, and borehole detection [19]. In agricultural applications, FOMO achieved 90.2% detection accuracy for strawberry detection on embedded hardware [20]. Additional studies have reported its use in currency recognition, railway signal detection, and wildlife monitoring, highlighting its versatility for edge-based vision systems [21–23]. Furthermore, improvements such as adaptive tiling techniques have enhanced FOMO performance by reducing object counting errors by 76% and increasing F1-scores by over 225% [24].

Although deep learning-based fruit detection has been extensively investigated, many existing approaches rely on computationally intensive architectures that require GPU-class hardware for deployment. Recent advances in TinyML have highlighted the importance of deploying machine-learning models directly on resource-constrained edge devices and microcontrollers for real-time applications [25]. FOMO (Faster Objects, More Objects) was specifically developed to enable object detection on highly constrained embedded platforms through centroid-based localization rather than computationally expensive bounding-box regression. As a result, FOMO can achieve object localization using substantially lower memory and processing resources than conventional MobileNet SSD and YOLO-based detectors [26]. As on the authors' awareness and knowledge, not much research has been conducted in precision agriculture specifically for brinjal crops, although several approaches have been explored for detection and maturity assessment. Early work by Hayashi et al. [27,28] utilized RGB-based channel subtraction techniques to isolate brinjal fruits, achieving a harvesting rate of 62.5% with a processing time of approximately 64.1 s per fruit; however, the method was highly dependent on controlled lighting conditions and exhibited slow processing speed, limiting its applicability in real-time field environments. Subsequent studies, such as Jian [29], applied grayscale-based segmentation and contour analysis for fruit localization, but these approaches were sensitive to background complexity and lacked robustness under varying field conditions. Miraei et al. [30] focused on improving mechanical harvesting through end-effector design; however, their work did not address the vision-based detection challenges required for autonomous harvesting systems. More recently, deep learning-based approaches have been introduced, with Kahya et al. [31] employing YOLOv5 models to achieve high detection precision of 95.65%; nevertheless, such models require substantial computational resources and larger model sizes, limiting their deployment on resource-constrained edge devices. In addition, machine vision-based maturity estimation methods using pixel clustering and spatial features have reported accuracies of up to 96% [32], but these methods rely on handcrafted features and lack scalability under diverse field conditions. Edge-based implementations such as the Brinjal Harvesting Decision System (BHDS) have demonstrated practical applicability under field conditions, achieving a detection precision of 87.48% and an F1-score of 87.53% [33]; however, the reliance on conventional image processing techniques results in slower inference speeds and limited real-time capability. Similarly, recent lightweight deep learning improvements such as YOLOv11s-Brinjal have achieved high detection performance with precision of 94% and F1-score of 95.3% [34], but the model still involves relatively higher computational complexity compared to ultra-lightweight architectures, which may restrict its deployment on low-power edge platforms.

To address the research gap of achieving good detection performance, low computational cost, and real-time use on low-power devices at the same time, this study evaluates the Faster Objects, More Objects (FOMO) model-based lightweight object detection approach for brinjal detection under natural field conditions, as shown in Figure 1. The key objective is to evaluate the detection accuracy of the model in diverse environmental settings. Additionally, to enable deployment on resource-limited edge devices, model

optimization techniques such as quantization were employed. The optimization technique reduces the computational and the model size requirements while keeping the admissible detection accuracy. The study evaluates the trade-offs between Float32 and INT8 model formats, quantifies their effects on accuracy, memory consumption, inference speed, and deployment performance, and validates the resulting models under natural field conditions using the OpenMV Cam H7 platform. By focusing on a lightweight and edge-optimized detection approach, this research aims to illuminate the applicability of the FOMO model for practical agricultural applications. The findings are expected to expand the understanding of computer vision applications in precision agriculture, particularly in the context of brinjal detection in natural field environments. Autonomous harvesting is a multi-stage process involving perception, localization, maturity assessment, manipulation planning, end-effector interaction, and harvesting verification. Therefore, the contribution of the present study focuses specifically on adapting, optimizing, quantizing, deploying, and validating a TinyML-oriented FOMO framework for the perception component by developing a lightweight-embedded vision framework capable of real-time brinjal detection under field conditions, using embedded hardware suitable for precision agriculture applications.

Automated Fruit Detection System Workflow

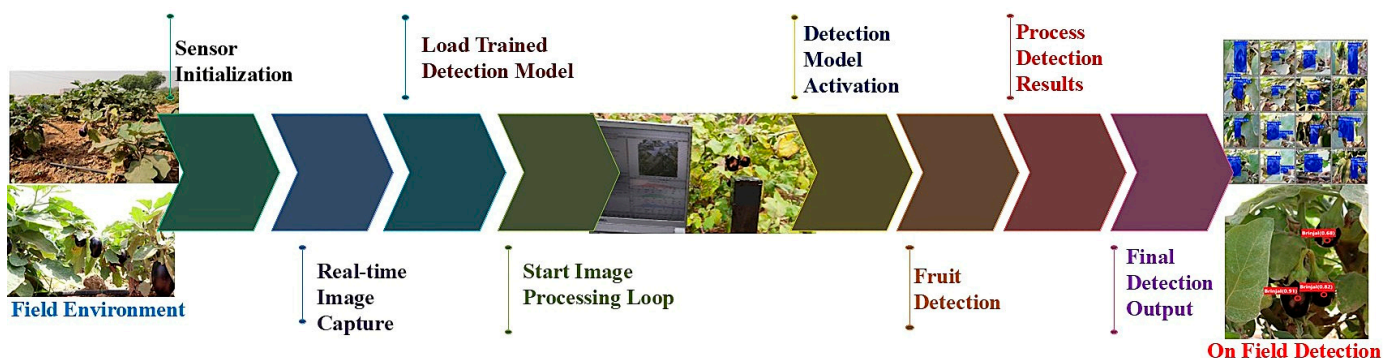


Figure 1. Workflow of the automated brinjal fruit detection system using the lightweight detection model FOMO.

2. Material and Methods

The materials and methodologies employed in the proposed approach describes the dataset acquisition settings and experimental setup. This is followed by an overview of the detection algorithms used, specifically FOMO (Faster Objects, More Objects). Subsequently, the application of learning techniques is discussed, including transfer learning and hyperparameter fine-tuning. The model optimization strategies adopted in the study are then outlined. Finally, the evaluation metrics used to assess the performance of the proposed approach are briefly explained.

2.1. Dataset Acquisition Setting and Experimental Setup

The dataset employed in this experiment comprises 4500 images (1500 original field images + 3000 pre-processed and augmented images) of brinjal fruits captured using a 24.1-megapixel digital camera equipped with an APS-CMOS sensor and DIGIC-8 image processor [30]. The sensor has physical dimensions of 22.3 × 14.9 mm and recorded images with a resolution of 2112 × 1600 pixels. The camera was fitted with an 18–45 mm focal length lens with an aperture range of f/4.5–6.3. No publicly available datasets were used in this study. The image dataset was collected from experimental brinjal fields at ICAR–

Indian Agricultural Research Institute (ICAR-IARI), New Delhi, India. The original images were captured across multiple acquisition dates, different experimental field plots, varying illumination conditions (morning, afternoon, and evening), different camera-to-fruit distances, and multiple viewing angles. Furthermore, the dataset included variations in fruit orientation, canopy structure, leaf occlusion, and natural field backgrounds as illustrated in Figure 2. Such diversity was incorporated to improve model robustness under realistic agricultural operating conditions. Images were acquired under non-uniform lighting conditions, incorporating variations in natural sunlight during morning, afternoon, evening, and shaded environments to strengthen the detecting model's stability. Approximately 150 images were captured per day for five days, with image acquisition sessions conducted at 12-day intervals to incorporate variations in plant growth and environmental conditions. All images were captured during the maturity stage of brinjal fruits, which corresponds to the stage suitable for human consumption. To enhance the robustness and generalization capability of the model, a comprehensive data preprocessing and augmentation pipeline was implemented mentioned in Table 1, using an online cloud-based platform Edge Impulse with student access. For model development, the dataset was divided into training and validation subsets, where 80% of the images (3600 images) were used for training and 20% (900 images) were used for validation. Image annotation was performed manually using the Roboflow platform for labelling interface. Each visible brinjal fruit instance was annotated using rectangular bounding boxes that enclosed the fruit region. These annotations served as the object-location labels for model development. During training, the FOMO framework automatically converted the bounding-box annotations into centroid-based localization targets represented on the output heatmap. All annotations were visually inspected and verified prior to model training to ensure consistency and labelling accuracy. All images were resized to 256×256 pixels before training and evaluation. This resolution was selected to balance detection performance and computational efficiency for deployment on the OpenMV Cam H7 platform. While higher resolutions may preserve additional spatial detail for very small or partially occluded fruits, they also increase computational and memory requirements, which are critical constraints for TinyML-based embedded systems. Experiments were performed on a workstation equipped with an Intel Core i7-11900HK CPU, 32 GB RAM, and an NVIDIA GeForce RTX 3050 Ti GPU with 12 GB VRAM, enabling efficient training and evaluation of both Float32 and INT8 versions of the model.

Table 1. Data preprocessing and augmentation pipeline techniques and their settings.

Technique	Type	Setting	Purpose
Image resizing	Preprocessing	256×256 pixels (resize-fit)	Standardizes input size; ensures computational efficiency while retaining sufficient spatial resolution for feature extraction
Random rotation	Geometric	-15° to $+15^\circ$	Simulates natural plant orientations and varying camera angles during data acquisition
Adaptive histogram equalization (CLAHE)	Enhancement	Local contrast enhancement	Emphasizes subtle surface texture; highlights glossiness and skin quality indicators critical for detection assessment
Gaussian blur	Photometric	Up to 0.5 pixels	Simulates natural camera motion, focus variations, and environmental factors, such as wind movement, during image capture
Scaling and shifting	Geometric	Scale: 50–150%; shift: up to 10%	Improves generalization to object size and location variability

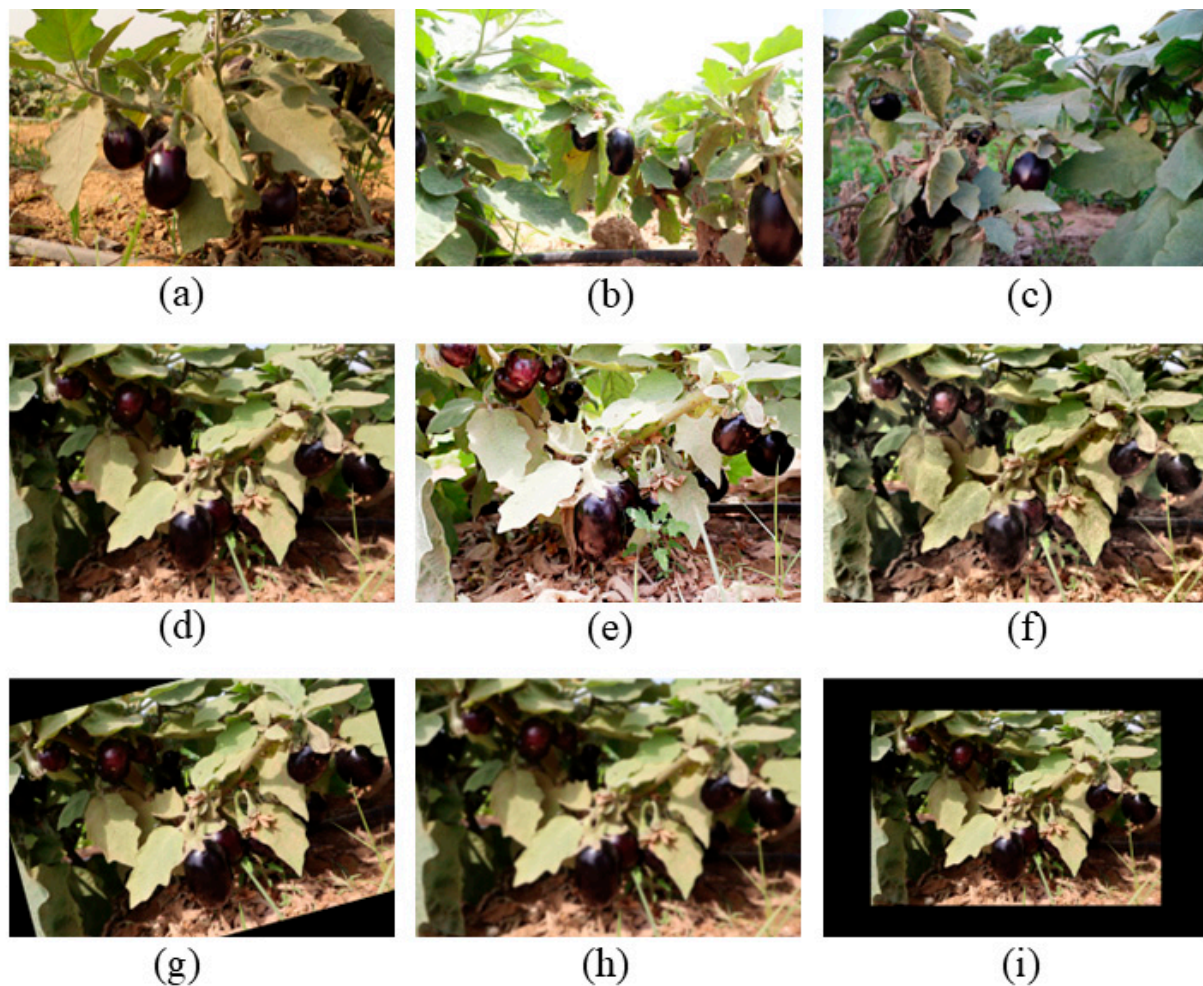


Figure 2. Sample of original acquired dataset under natural sunlight environment and enhanced images; (a) morning, (b) afternoon, (c) evening, (d) original image before enhancement, (e) brightness adjusted image, (f) CLAHE adjusted image, (g) random rotation applied image, (h) Gaussian blur-added image, (i) scale-adjusted image.

To prevent information leakage between training and validation datasets, all augmented images generated from a given original image were maintained within the same dataset partition. Consequently, augmented derivatives originating from a particular source image were assigned exclusively either to the training dataset or to the validation dataset and were never distributed across both subsets. This grouping strategy ensured that validation samples remained independent of training samples and prevented artificially inflated performance estimates caused by highly correlated augmented images [35].

2.2. FOMO Architectures

The object detection framework proposed in this study is built upon the FOMO (Faster Objects, More Objects) architecture [27], specifically adapted for real-time brinjal fruit detection under uncontrolled natural field conditions. The framework is built around a truncated MobileNetV2 backbone, modified to serve as a spatial feature extractor from conventional anchor-based detection paradigms by replacing the bounding-box regression head with a heatmap-based object localization mechanism illustrated in Figure 3. The design prioritizes low computational overhead and deployment feasibility on microcontroller-class hardware without the post-processing overhead of non-maximum suppression, while maintaining reliable detection performance under varying illumination, occlusion, and background conditions typical of agricultural settings.

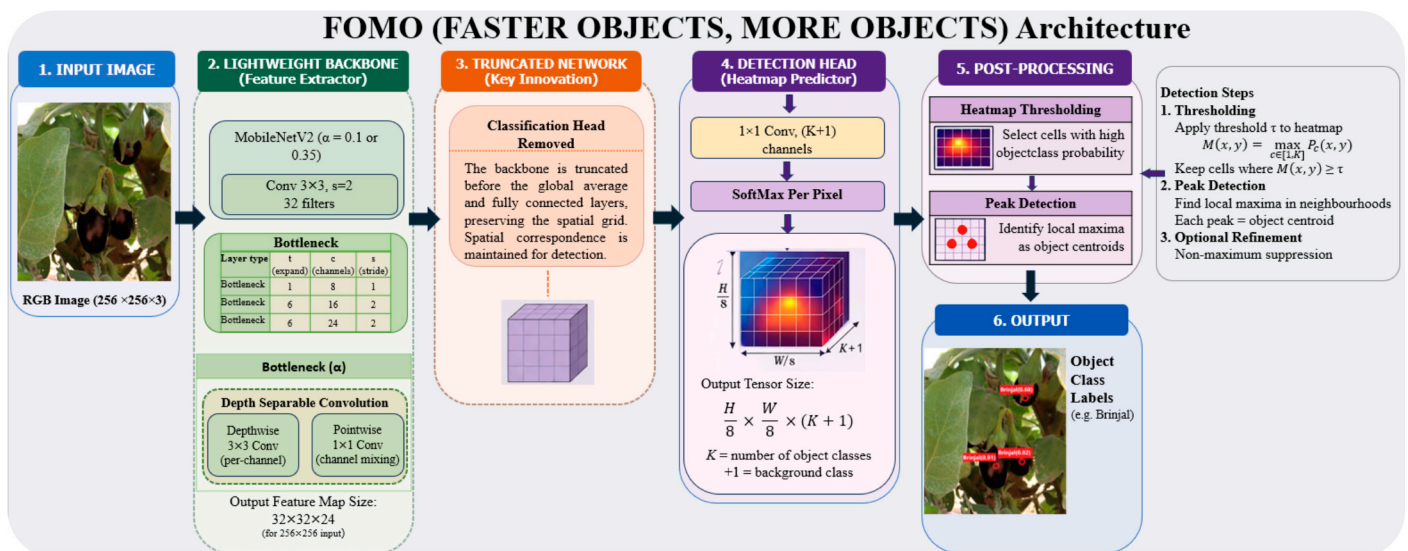


Figure 3. Detailed architecture of the object detection framework for brinjal fruit detection.

The proposed pipeline follows a two-stage design: (i) a truncated MobileNetV2 backbone that performs hierarchical spatial feature extraction, and (ii) a compact heatmap prediction head that localizes brinjal fruit within the resulting feature space as elucidated in Figure 4. The network processes input images of resolution $256 \times 256 \times 3$ and progressively reduces spatial dimensions while increasing channel depth, culminating in a $32 \times 32 \times 96$ cut point feature map. The input resolution of 256×256 pixels was selected to balance localization performance and computational efficiency for deployment on the OpenMV Cam H7 platform. This spatial representation, at an effective stride of 8 relative to the input, is passed directly to the detection head without global pooling or dense classification layers—a design choice that preserves the full spatial awareness required for object localisation in complex canopy backgrounds.

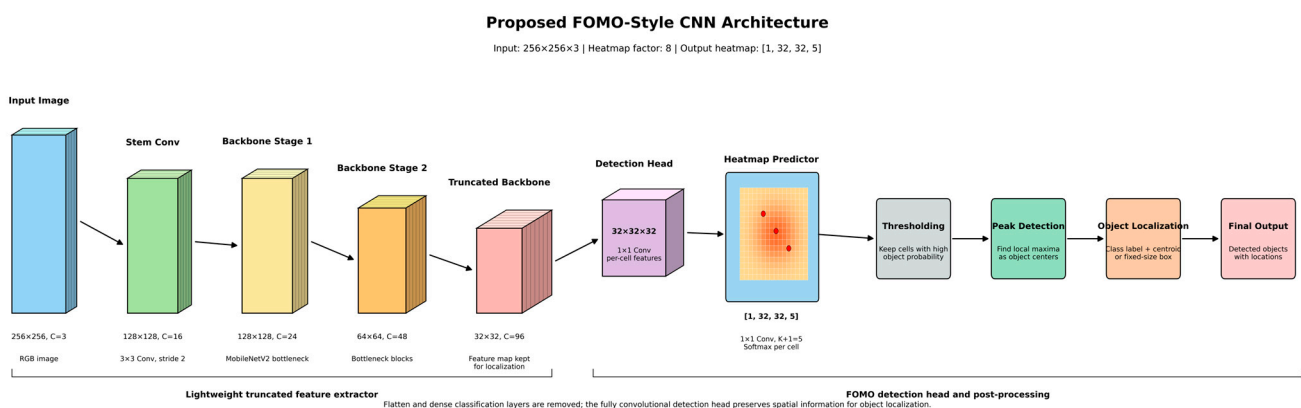


Figure 4. The final FOMO MobileNetV2 network architecture.

2.2.1. Backbone

To meet the computational constraints of edge deployment in agricultural settings, the backbone is derived from MobileNetV2 [30], a compact architecture that employs inverted residual bottleneck blocks with depthwise-separable convolutions to reduce multiply accumulate operations without a commensurate loss in representational capacity as illustrated in Figure 5. The network is truncated at an intermediate-expanded activation layer, yielding a spatially rich feature volume suited for dense heatmap prediction.

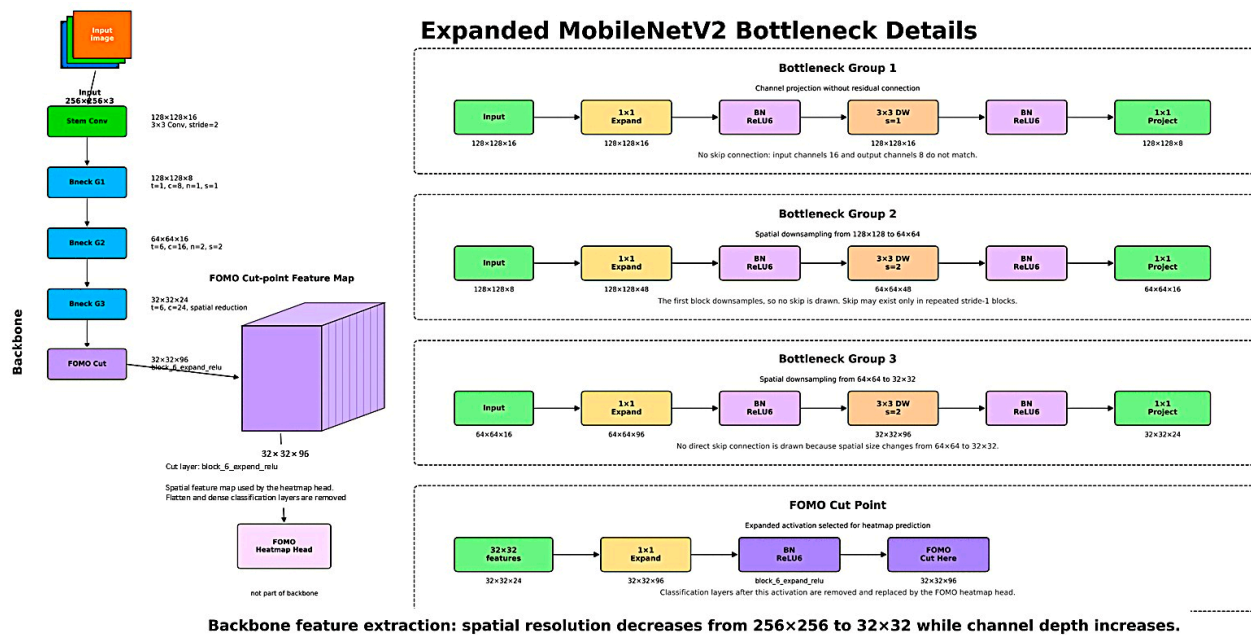


Figure 5. Backbone bottleneck structure diagram.

2.2.2. Stem Layer

A 3×3 convolutional layer with stride 2 processes the $256 \times 256 \times 3$ input, producing an initial $128 \times 128 \times 16$ feature map. This stage establishes low-level visual features—edges, colour gradients, and texture patterns, relevant to brinjal fruit identification while halving spatial resolution to reduce subsequent computational load.

2.2.3. Inverted Residual Bottleneck Groups

Three sequential bottleneck groups progressively reduce spatial resolution from 128×128 to 32×32 while expanding channel depth. Group 1 ($t = 1, c = 8, n = 1, s = 1$) compresses channels at full resolution. Group 2 ($t = 6, c = 16, n = 2, s = 2$) performs the first stride-2 downsampling from 128×128 to 64×64 . Group 3 ($t = 6, c = 24, n = 2, s = 2$) applies a second stride-2 downsampling from 64×64 to 32×32 , expanding intermediate feature maps to 96 channels. Residual skip connections are retained only in stride-1 blocks where input and output dimensions match; strided blocks with mismatched spatial sizes do not carry skip connections.

2.2.4. FOMO Cut Point

The backbone is truncated at the expanded activation tensor `block_6_expand_relu`—the post-expansion, pre-projection feature volume of shape $32 \times 32 \times 96$. This intermediate representation is preferred over the final projected output ($32 \times 32 \times 24$) because it retains a higher-dimensional activation space with greater discriminative capacity for spatial localisation. All subsequent global average pooling and dense classification layers of the original MobileNetV2 architecture are discarded.

2.3. Detection Head and Output Decoding

The $32 \times 32 \times 96$ feature map is passed to a lightweight heatmap prediction head consisting of a single 1×1 convolutional layer followed by a per-pixel softmax activation. This produces an output tensor of shape $32 \times 32 \times (K + 1)$, where K denotes the number of target object classes ($K = 1$ for brinjal) and the additional channel encodes background probability. Each spatial cell in the 32×32 grid represents an 8×8 -pixel region in the

original 256×256 input image; brinjal fruit detections are identified by locating peak activations in the foreground class channel that exceed a predefined confidence threshold.

Neighbouring high-confidence predictions within a spatial proximity are grouped through a centroid clustering process to consolidate activations arising from the same fruit instance into a single detection event. This post-processing step replaces the bounding-box regression and non-maximum suppression routines of conventional detectors, substantially reducing inference time overhead and making the model suitable for deployment on devices with limited processing capability.

2.4. *Lightweight Design and Computational Efficiency*

The FOMO architecture employs a truncated MobileNetV2 backbone followed by a lightweight fully convolutional detection head. The total number of trainable parameters depends on the selected MobileNetV2 width multiplier (alpha), network cut point, input resolution, and number of output classes. Consequently, parameter count and memory footprint vary according to the deployment configuration adopted for a particular application. The systematic use of depthwise-separable convolutions reduces multiply accumulate operations by a factor of approximately 8–9 relative to standard convolutions of equivalent spatial coverage [36], making the model well-suited for deployment on embedded platforms, such as the Arduino Nicla Vision and Raspberry Pi series, commonly employed in precision agriculture applications.

The heatmap-based output formulation further reduces inference overhead by eliminating anchor decoding, bounding-box regression, and non-maximum suppression—operations that collectively account for a significant proportion of post-processing latency in region-based and anchor-based detectors. Quantitative inference benchmarks, including memory footprint and per-frame latency for both Float32 and INT8 model representations.

2.5. *Application of Learning Techniques*

2.5.1. Transfer Learning

Deep learning models are composed of a layered architecture, in which various layers are arranged in a specific order. These layers can extract attributes from the input data images, learn, and subsequently predict the output [37]. Transfer learning is a machine-learning process that employs a previously trained model as the foundation for the new model on a closely linked second task. Pre-trained models are typically trained using a vast and varied dataset, such as ImageNet, which has millions of photos and thousands of object types [38]. This model was trained to extract necessary image attributes such as textures, edges, and shapes. Following attributes can be employed to initialize the weights of a new model, which is subsequently refined on a smaller dataset, offering numerous benefits. The model for the second job can be trained more rapidly, as fundamental features have already been acquired. Transfer learning can yield superior performance compared to training a model from the ground up, particularly when the new dataset resembles the pre-trained dataset [39]. It can improve the generalization of the new model. The pre-trained model has learned features that are useful to a variety of tasks. Transfer learning enables accurate prediction with less data.

2.5.2. Fine-Tuning of Hyperparameters

Fine-tuning of hyperparameters performs critically in optimizing models to achieve improved detection accuracy and generalization performance. Hyperparameters regulate the learning process, and their selection depends on both the model architecture and the characteristics of the training dataset [36,40,41]. In this study, key hyperparameters were configured as the number of batch size epochs and learning rate to obtain optimal detection performance. The models were trained with epoch values of 25, 50, and 100, while a

batch size of 8 was used for updating model weights during each training iteration. To strengthen model generalization and minimize overfitting, data augmenting methods, including operations such as rescaling, zooming, shearing and horizontal flipping of the input images, were applied. The model had been trained using the Adam optimizer and the categorical cross-entropy loss function with a 0.001 learning rate, which is the default setting in the Keras library [42]. The final hyperparameters were configured based on the model achieving the optimum detection accuracy. The set of hyperparameters used during the training process is summarized in Table 2.

Table 2. Hyperparameters used for training the FOMO model.

Serial	Parameter	Value/Range
1	Model Architecture	Faster Objects, More Objects (FOMO)
2	Number of Epochs	25, 50, 100
3	Batch Size	8
4	Learning Rate	0.001
5	Optimizer	Adam
6	Loss Function	Categorical Cross Entropy
7	Data Augmentation	Rescaling, Shearing, Zooming, Horizontal Flip, CLAHE, Gaussian Blur

2.6. Model Optimization

Aiming improve computational efficiency and reduce model size; post-training quantization was applied to convert the model weights and activations to 8-bit integer (INT8) format [43]. Quantization reduces the memory footprint and accelerates inference by enabling integer arithmetic operations, making the model more suitable for deployment on edge-AI hardware and embedded systems [44,45]. For the evaluation of the model optimization impact on detection accuracy and computational efficiency, two versions of the trained FOMO model were trained and evaluated: a Float32 model and an INT8-quantized model. The Float32 model represents the baseline configuration, where model activations and weights are stored in 32-bit floating-point format. This format provides high numerical precision but requires greater memory and computational resources. Model quantization was applied to convert the trained Float32 model into an INT8 representation to reduce memory usage and computational complexity. To investigate the influence of quantization at different stages of model convergence, three independent Float32 model checkpoints were retained after 25, 50, and 100 training epochs. Post-training quantization (PTQ) was then applied separately to each checkpoint to generate the corresponding INT8 models. Consequently, the reported INT8 results do not represent models trained in integer precision; rather, they represent quantized versions of previously trained Float32 checkpoints evaluated after the PTQ conversion process. Quantization maps floating-point values X to integer values X_q using a linear transformation defined by a scale factor S and zero-point Z . The quantized value is computed as per Equations (1) and (2) [46],

$$X_q = \text{round} \left(\frac{X}{S} \right) + Z \quad (1)$$

The scale factor was calculated using the maximum and minimum values of the tensor, i.e., floating-point values range (X_{max} , X_{min}) and integer limits (q_{max} , q_{min})

$$S = \frac{X_{max} - X_{min}}{q_{max} - q_{min}} \quad (2)$$

This process enables efficient integer arithmetic during inference while maintaining comparable model accuracy.

The INT8 model was generated using the post-training quantization (PTQ) workflow and exported as a TensorFlow Lite Micro (TFLM) model for deployment on the OpenMV Cam H7 platform. The Float32 model was converted to an INT8 model using the TensorFlow Lite full-integer post-training quantization (PTQ) framework, which employs linear affine quantization for efficient embedded inference. In the exported TensorFlow Lite model, the convolutional weights are represented using per-channel (per-axis) INT8 quantization, whereby each output channel is assigned an independent quantization scale, while the input tensor, intermediate activation tensors, and output tensors are represented using per-tensor INT8 quantization with a single scale and zero-point. The convolution biases are stored as INT32 values with scaling factors derived from the corresponding input and weight quantization parameters, in accordance with the TensorFlow Lite quantization specification. The quantization parameters are generated automatically during model conversion and embedded within the exported TensorFlow Lite model. The resulting quantized model was deployed on the OpenMV Cam H7 using the TensorFlow Lite Micro inference engine together with ARM CMSIS-NN optimized kernels and TensorFlow Lite Micro components for efficient inference on the STM32H743 Cortex-M7 microcontroller [47], enabling efficient low-memory inference while maintaining detection performance.

Inference was executed using the TensorFlow Lite Micro runtime with static tensor-arena memory allocation. This memory-management strategy preallocates inference buffers at compile time and avoids dynamic memory allocation during execution, improving memory predictability and deployment reliability on resource-constrained embedded systems.

Although post-training quantization is conventionally applied only to the final converged model for deployment, the present study additionally quantized intermediate Float32 checkpoints obtained after 25, 50, and 100 training epochs. The purpose of this analysis was to systematically investigate how model convergence influences post-training quantization performance and to evaluate the evolution of Float32–INT8 performance differences throughout the training process. These intermediate INT8 models were generated exclusively for comparative analysis and were not intended to represent the deployment workflow. The final embedded implementation on the OpenMV Cam H7 employed the INT8 model obtained from the fully converged Float32 network after 100 training epochs.

2.7. Evaluation Matrices

The performance of the proposed FOMO-based brinjal detection framework was evaluated using classification-, localization-, and counting-oriented metrics. Unlike conventional object detectors such as YOLO, Faster R-CNN, and Mask R-CNN that perform bounding-box regression, FOMO employs centroid-based localization using probability heatmaps and therefore does not directly generate bounding boxes. Consequently, evaluation was performed using metrics consistent with the operating principle of the FOMO architecture, including precision, recall, F1-score, miss rate, missed-fruit rate, and counting-performance metrics [48]. Here are some key metrics (Equations (3)–(8)) used for evaluating object detection models:

2.7.1. Precision

Determines the proportion of true positive (TP) detections across all positive detections, including false positives (FP), produced by the model.

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

2.7.2. Recall

Defines the ratio of real positive detections across all actual positive detections obtained through the model.

$$Recall = \frac{TP}{TP + FN} \quad (4)$$

where FN is false negatives.

2.7.3. F1-Score

The harmonic mean of precision and recall yields a single statistic that balances both.

$$F1\ Score = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \quad (5)$$

2.7.4. Miss Rate

Miss rate represents the proportion of actual fruit instances that were not detected by the model.

$$Miss\ Rate = 1 - Recall \quad (6)$$

2.7.5. Missed-Fruit Rate

Missed-fruit rate quantifies the percentage of fruit instances that were missed during detection.

$$Missed\ Fruit\ Rate\ (\%) = \frac{FN}{(TP + FN)} \times 100 \quad (7)$$

2.7.6. Mean Absolute Error (MAE)

To evaluate the counting capability of the proposed framework, Mean Absolute Error (MAE) was computed between the predicted fruit count and the ground-truth fruit count.

$$MAE = \frac{1}{(N)} \times \sum_{i=1}^N |Pi - Gi| \quad (8)$$

where Pi is the predicted fruit count, Gi is the ground-truth fruit count, and N is the total number of test images.

2.7.7. Root Mean Square Error (RMSE)

RMSE was used to assess counting accuracy while giving greater weight to larger counting errors.

$$RMSE = \sqrt{\frac{1}{N} \times \sum_i^N (Pi - Gi)^2} \quad (9)$$

These metrics collectively provide a comprehensive assessment of the detection and counting capabilities of the proposed lightweight-embedded vision framework. Precision, recall, F1-score, and miss rate characterize detection performance, whereas MAE and RMSE evaluate the counting capability enabled by the centroid-based localization mechanism of the FOMO architecture.

The confusion matrix reported by the FOMO evaluation framework differs from that of conventional image-classification systems. FOMO performs object localization through heatmap-based centroid prediction, where the input image is internally represented as a grid of localization cells. Consequently, confusion-matrix statistics are derived from centroid-matching and grid-level localization outcomes rather than image-level classification decisions. Because background cells substantially outnumber object cells, background true-negative values are typically very large and should be interpreted with caution.

Therefore, precision, recall, F1-score, missed-fruit rate, and counting-performance metrics were considered the primary indicators of model performance.

The model generated a 32×32 output heatmap from 256×256 input images, corresponding to an effective spatial resolution of 8×8 pixels per heatmap cell. Each heatmap cell therefore represented a localized region of the original image and functioned as an independent classifier. Performance metrics were computed using the heatmap-based evaluation framework, where object presence and localization were evaluated on the output feature map rather than through IoU-based overlap measurements. Consequently, the reported mAP values represent heatmap/centroid-localization performance and are not directly comparable to IoU-based mAP values reported for bounding-box detectors.

2.8. Independent Test Dataset Evaluation

To obtain an unbiased estimate of model generalization performance, an independent testing dataset was maintained separately from the training and validation processes. The testing dataset was not used during model training, hyperparameter selection, or model optimization. Following completion of training, the final Float32 and INT8 models were evaluated using the held-out testing dataset provided through the Edge Impulse evaluation framework. Performance metrics including precision, recall, F1-score, confusion matrix, and detection accuracy were computed using this unseen dataset.

3. Result and Discussion

3.1. Training Convergence and Performance Progression of the FOMO Brinjal Detection Model

The potential of the proposed FOMO-based brinjal detection model was first analyzed by examining the learning behaviour throughout the training process. The model training was carried out using the dataset containing 4500 brinjal images captured under natural field conditions, incorporating variations in illumination, occlusion, and background complexity. Training was conducted using the hyperparameters described in Table 2. Figure 6 presents the training dynamics of the proposed model across 100 epochs, including training loss, precision, recall, F1-score, and centroid-localization performance indicators generated by the FOMO framework. The training loss decreased sharply from an initial value of approximately 2.0, stabilizing below 0.20 after epoch 40, indicating effective gradient descent and model convergence without evidence of divergence or oscillatory behaviour. Precision and recall exhibited monotonic improvement throughout training, with recall reaching approximately 0.92 and precision stabilizing at approximately 0.83 by epoch 100. The F1-score improved from 0.55 at epoch 0 to 0.87 at epoch 100, reflecting consistent improvement in the balance between detection sensitivity and specificity. The centroid-localization performance indicator (mAP) generated by the Edge Impulse evaluation framework increased progressively during training and reached approximately 0.30 at epoch 100. The reduced rate of improvement beyond epoch 50 suggests that additional training yielded diminishing performance gains, indicating near-convergence of the model under the selected dataset and network configuration.

Figure 7 compares the F1-scores of Float32 model checkpoints obtained after 25, 50, and 100 training epochs with the corresponding INT8 models generated through post-training quantization (PTQ). For each training checkpoint, the Float32 model was first trained to convergence for the specified number of epochs and subsequently converted into an INT8 representation using PTQ. The resulting INT8 performance therefore reflects the effect of quantization on the corresponding Float32 checkpoint rather than direct INT8 training. This comparison was performed to evaluate the evolution of quantization robustness during network convergence rather than to represent the standard PTQ deployment procedure, which is typically applied only after completion of model training. The

Float32 model exhibits consistent and substantial improvement across all checkpoints, progressing from a training F1-score of 0.821 at epoch 25 to 0.865 at epoch 50, reaching 0.892 at epoch 100—an absolute gain of 0.071. The INT8 variant follows a comparable trajectory (0.794 → 0.822 → 0.863), achieving a total gain of 0.069. The absence of a performance plateau between epochs 50 and 100—Float32: +0.027, INT8: +0.041—confirms that neither format reaches saturation within the training budget, and that a full 100-epoch run is necessary to realize peak performance.

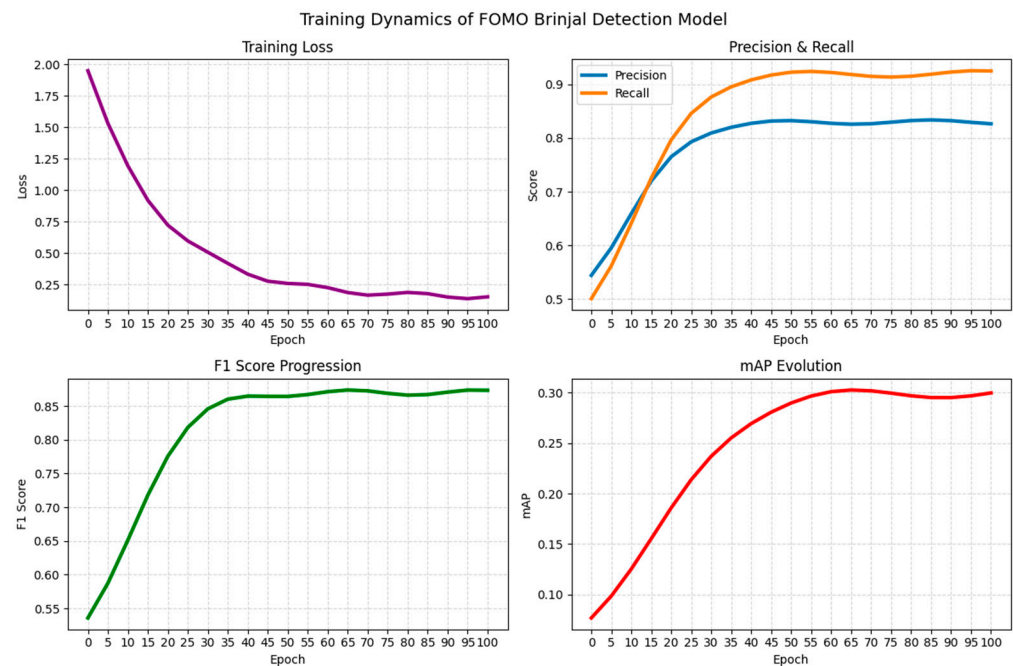


Figure 6. Training dynamics illustrating the evolution of training loss, precision, recall, F1-score, and centroid-localization performance indicators (mAP) during the 100-epoch training process.

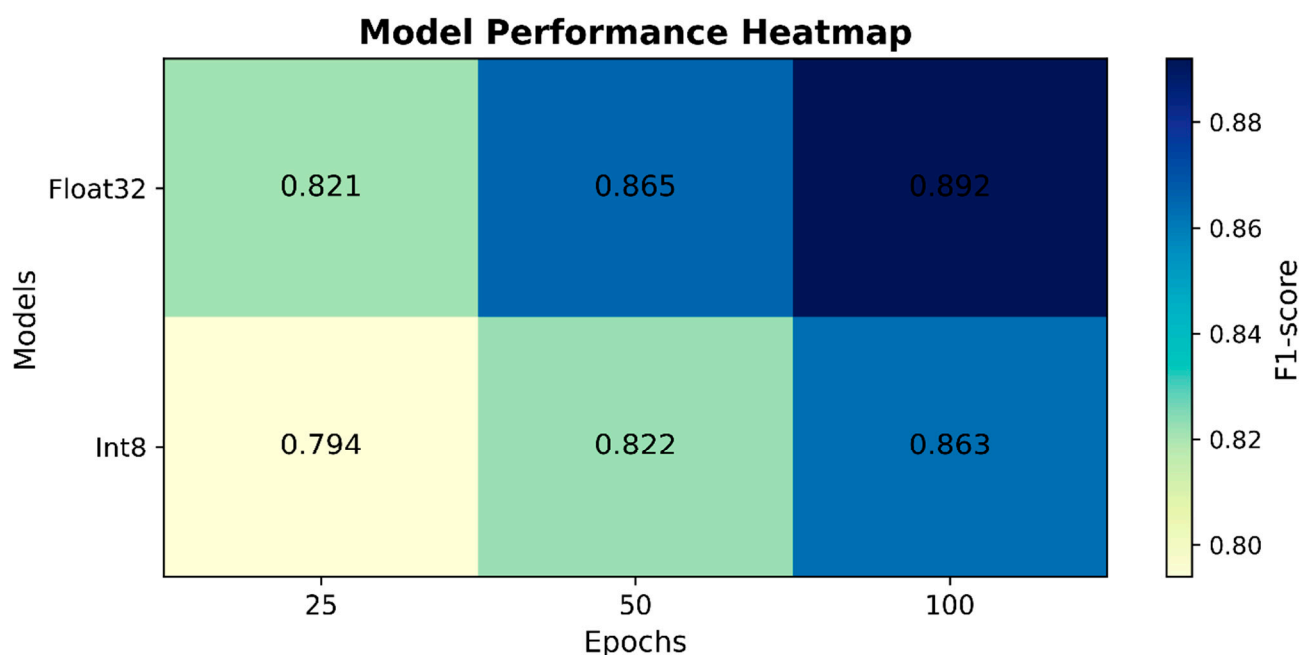


Figure 7. Effect of post-training quantization at different stages of model convergence.

The inter-format training F1 gap is smallest at epoch 25 ($\Delta = 0.027$), widens to its maximum at epoch 50 ($\Delta = 0.043$), and narrows to 0.029 at epoch 100. This pattern indicates

that the INT8 model is most sensitive to quantisation-induced precision loss during the mid-training phase and partially recovers as weight distributions stabilize in later epochs.

The localization metric reported by framework requires careful interpretation before comparison with published object detection benchmarks. Conventional mAP calculations are based on Intersection-over-Union (IoU) overlap between predicted and ground-truth bounding boxes. In contrast, FOMO performs object localization through heatmap prediction based on object centroids and does not directly regress bounding-box coordinates. The deployed model generated a 32×32 output heatmap from 256×256 input images, corresponding to an effective spatial resolution of 8×8 pixels per heatmap cell. Each heatmap cell functions as an independent classifier, and localization performance is evaluated on the output heatmap representation rather than through IoU-based bounding-box overlap. Consequently, the localization metric reported during training reflects heatmap-based centroid-localization performance rather than conventional IoU-based localization accuracy. The progressive increase of this metric during training therefore indicates improved localization of brinjal fruit instances on the output feature map. Direct numerical comparison with YOLO, Faster R-CNN, Mask R-CNN, or other bounding-box detectors is therefore inappropriate. For FOMO-based systems, precision, recall, F1-score, counting accuracy, missed-fruit rate, and counting-error metrics provide a more meaningful assessment of practical field performance.

The validation F1-score of 0.869 (Float32) and the precision and recall values are the primary accuracy indicators for this model class, as they assess detection at the object instance level without dependence on bounding box geometry. The relatively small performance degradation observed after post-training quantization is noteworthy because MobileNet-based architectures employing depthwise-separable convolutions have frequently been reported to exhibit increased sensitivity to INT8 quantization. Recent studies have shown that quantization-induced degradation in MobileNet architectures is strongly influenced by activation-range distributions, network depth, task complexity, and deployment configuration. While substantial performance degradation has been reported in large-scale benchmark datasets, lightweight edge-AI studies have also demonstrated that carefully optimized MobileNetV2-based models can retain competitive accuracy following INT8 quantization while achieving substantial reductions in model size and computational requirements [49,50]. However, the present study exhibited only a minor reduction in validation F1-score (0.869 to 0.866), suggesting that the combination of a single-class detection problem, centroid-based localization, and the MobileNetV2 inverted residual architecture contributed to quantization robustness under the investigated deployment conditions. A further plausible explanation is the quantization strategy adopted during model conversion. Inspection of the exported TensorFlow Lite model confirmed that convolutional weights were represented using per-channel (per-axis) INT8 quantization, whereas the input, activation, and output tensors employed per-tensor quantization, consistent with the TensorFlow Lite full-integer quantization framework. Per-channel quantization allows each output channel to retain an independent dynamic range, thereby reducing channel-wise quantization error accumulation that has been identified as a major source of performance degradation in MobileNet-based depthwise-separable convolutional networks. Consequently, the combined effects of per-channel weight quantization, the relatively simple single-class centroid-prediction task, and the MobileNetV2 inverted residual architecture may have collectively contributed to the minimal loss in validation performance observed after post-training quantization.

3.2. Validation Performance Across Epochs and Precision Formats

The effect of training epochs on the outcomes of the stated detection FOMO model was evaluated using three different epoch settings: 25, 50, and 100 epochs. The validation

performance of both Float32 and INT8 model representations in terms of precision, recall, and F1-score is illustrated in Figure 8. At 25 epochs, the Float32 model achieved a precision of 0.843, recall of 0.888, and F1-score of 0.864, while the INT8 model achieved precision of 0.840, recall of 0.883, and F1-score of 0.861. Validation F1 exceeds training F1 for both formats, Float32: 0.864 (val) vs. 0.821 (train) and INT8: 0.861 (val) vs. 0.794 (train) such a pattern attributable to data augmentation applied during training, which suppresses training-phase scores in early epochs by artificially increasing input diversity.

When the training duration was increased to 50 epochs, detection performance improved further. The Float32 model obtained a precision of 0.826, recall of 0.907, and F1-score of 0.865, while the INT8 model achieved a precision of 0.835, recall of 0.893, and F1-score of 0.863. The highest performance was observed at 100 epochs, where the Float32 model achieved a precision of 0.827, recall of 0.915, and F1-score of 0.869, whereas the INT8 model achieved a precision of 0.829, recall of 0.908, and F1-score of 0.866. By epoch 100, the Float32 training F1 (0.892) marginally exceeds validation F1 (0.869), yielding a generalization gap of 0.023, indicating mild but well-controlled overfitting. The consistently higher recall relative to precision across both formats (Float32: 0.915 vs. 0.827; INT8: 0.908 vs. 0.829) reflects the recall-oriented operating profile of the FOMO heatmap detection head. INT8 quantisation introduced a validation F1 degradation of only 0.003 (0.3%) relative to Float32, confirming that the quantisation process preserves the model's learned detection boundary with negligible accuracy loss.

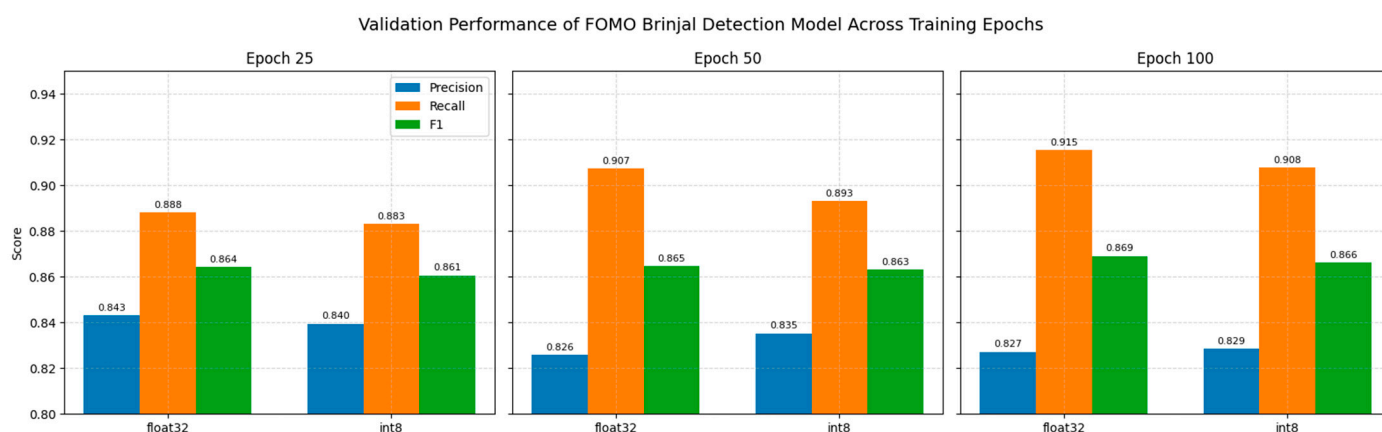


Figure 8. Comparative validation performance of the FOMO brinjal detection Float32 and INT8 model performance across training epochs.

These outcomes indicated that the increase in training epoch numbers allows the model to learn more discriminative features for detecting brinjal fruits under natural field conditions. A comparison between the Float32 and INT8 models shows that both configurations achieved comparable detection performance obtained across all epoch settings. Although the Float32 model exhibited slightly higher recall values, the INT8 model maintained similar precision and F1-score values, indicating that model quantization has minimal impact on detection accuracy.

3.3. Precision–Recall Relationship Analysis

Figure 9 illustrates the precision–recall trade-off obtained by varying the confidence threshold (0.4–0.7) of the FOMO detector. As expected, decreasing the confidence threshold increased recall at the expense of precision, whereas higher thresholds reduced false detections but increased the likelihood of missed-fruit instances. For the Float32 model, precision varied from approximately 0.878 to 0.827 while recall increased from approximately 0.85 to 0.924. A similar trend was observed for the INT8 model, where precision

decreased from approximately 0.863 to 0.822 as recall increased from approximately 0.867 to 0.923. Both model formats occupied nearly identical regions of the precision–recall space, corroborating the 0.3% validation F1 degradation reported in Section 3.2 and indicating that post-training INT8 quantisation did not substantially alter the operating characteristics of the detector. The small separation between corresponding Float32 and INT8 operating points corroborates the negligible validation-performance degradation observed after quantisation.

The deployment configuration employed a confidence threshold of 0.50, which provided the best compromise between detection sensitivity and false-positive suppression under field conditions. At this operating point, the detector achieved high recall while maintaining acceptable precision, thereby minimizing missed mature fruits without introducing excessive false detections. This threshold was therefore selected for OpenMV deployment and all subsequent field evaluations.

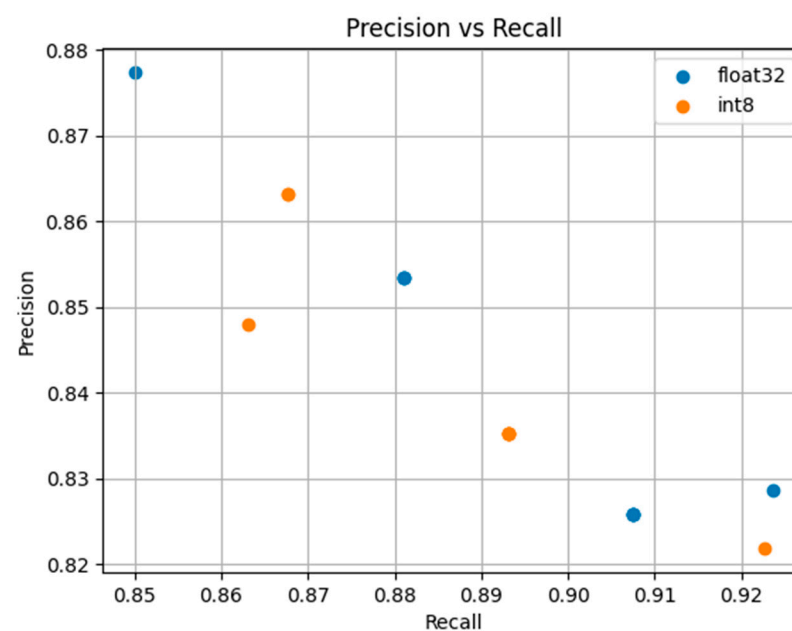


Figure 9. Precision–recall relationship of Float32 and INT8 models of the proposed FOMO-based brinjal detector.

3.4. Confusion Matrix Analysis

The detection potential of the trained model was further examined using a normalized confusion matrix, as shown in Figure 10.

In FOMO, each heatmap cell acts as an independent classifier and is assigned either a target class or background class. Consequently, background cells substantially outnumber fruit-containing cells, resulting in very large true-negative counts for the background category. Therefore, the near-unity background true-negative rate primarily reflects the dominance of background localization cells and should not be interpreted as a standalone indicator of detector performance.

For brinjal fruit (Class 1), the model achieved a true positive rate of 0.85 and a false negative rate of 0.15, indicating that approximately 15% of fruit instances were missed during detection. More informative indicators of detection performance include precision, recall, F1-score, missed-fruit rate, and counting-performance metrics, which directly evaluate object-level localization outcomes. Figure 11 illustrates representative detection results showing successful localization of brinjal fruits under field conditions with minimal false detections in non-fruit regions.

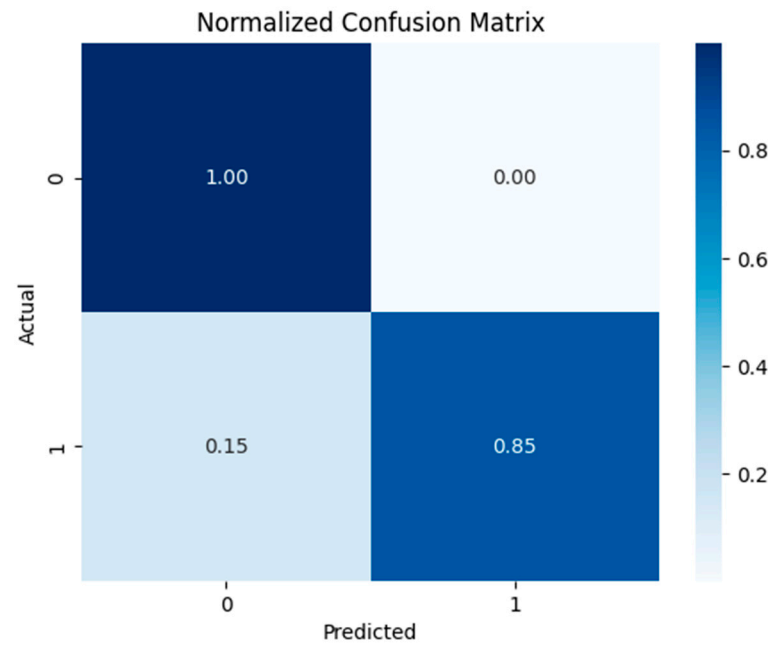


Figure 10. Normalized confusion matrix representing the detection potential of the model.

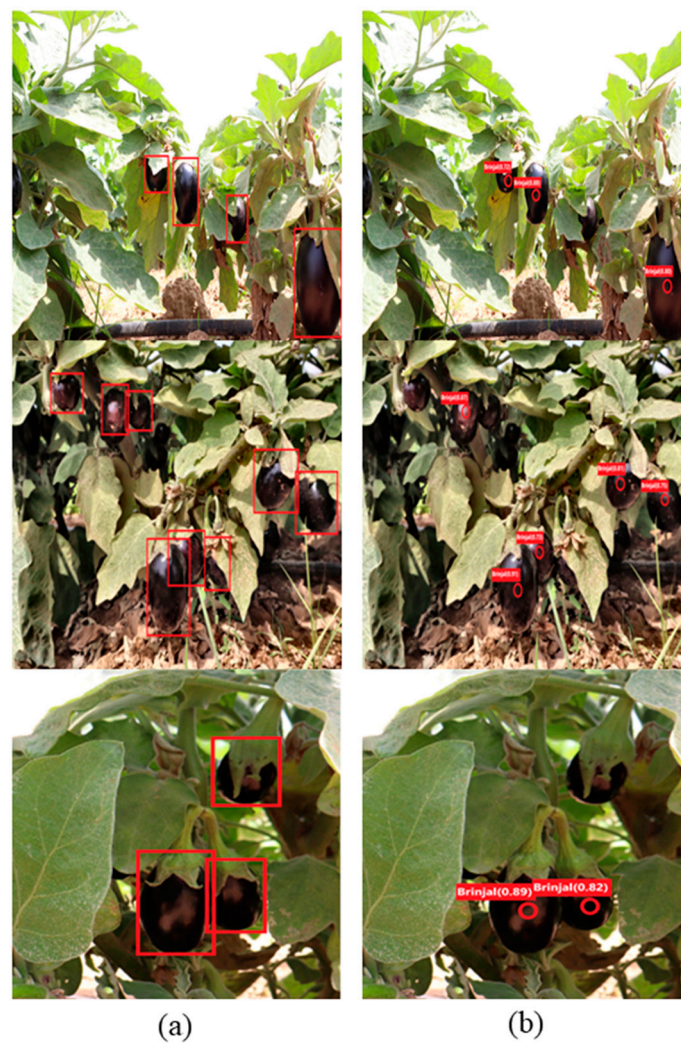


Figure 11. Detection output of the proposed FOMO model on purple brinjal plants under natural field conditions: (a) ground-truth bounding box annotations and (b) predicted centroid-based

detections, demonstrating the model's ability to identify multiple brinjals including occluded and clustered instances.

The missed-detection rate is primarily attributable to severe fruit-foliage overlap and dense fruit clustering under natural canopy conditions. Because FOMO performs localization through a low-resolution heatmap, i.e., 32×32 heatmap grid (8×8 -pixel resolution per cell), neighbouring fruits occupying adjacent heatmap cells may generate merged activation regions. With the selected image-to-heatmap scaling factor, the resulting localization grid provides limited spatial granularity for separating closely spaced fruit instances. This limitation becomes more pronounced under dense canopy conditions where fruit visibility is reduced. The FOMO architecture relies on centroid prediction rather than bounding-box regression, and overlapping centroids may become difficult to separate when multiple objects occupy neighbouring localization cells. Previous studies on FOMO have similarly reported reduced performance when objects are closely spaced or when centroid overlap occurs, suggesting that heatmap resolution is a critical factor governing localization performance with the known limitations of stride-8 detection heads for occluded targets [51]. Increasing the heatmap resolution or using larger input images can improve object separation and partially mitigate this limitation.

An additional factor contributing to missed detections is the image-resolution trade-off inherent to embedded deployment. All images were resized to 256×256 pixels before training and inference to satisfy the computational and memory constraints of the OpenMV Cam H7 platform. While this resolution enabled efficient real-time execution, small fruits occupying only a limited number of pixels may lose discriminative spatial information after resizing. Furthermore, FOMO performs localization through a lower-resolution heatmap representation rather than pixel-level object boundaries. Consequently, small, distant, or heavily occluded fruits may be represented by only a few heatmap activations, increasing the probability of missed detections. Edge Impulse documentation similarly notes that increasing image resolution or heatmap resolution can improve localization performance for small or closely spaced objects, albeit at the cost of increased computational requirements.

3.5. Inference Performance and Memory Footprint

Table 3 presents deployment metrics for both model formats measured on OpenMV Cam H7. The Float32 model requires 0.30 MB of storage and 887.2 KB of RAM, achieving 65.3 ms inference (15.3 FPS). The INT8 model reduces storage to 0.11 MB (2.73 times reduction), RAM to 239.4 KB (3.7 times reduction), and inference time to 24.8 ms, yielding 40.3 FPS, a speedup of 2.63 times over Float32. The deployment validation F1 cost of quantisation is 0.02 (2.25%).

Table 3. Measured deployment metrics for Float32 and INT8 FOMO variants on OpenMV Cam H7.

Metric	Float32	INT8 (Quantised)	Improvement
Model Size (MB)	~0.30	~0.11	Reduced 2.73 times
RAM Usage (KB)	887.2	239.4	Reduced 3.70 times
Inference Time (ms)	65.3	24.8	Reduced 2.63 times
Throughput (FPS)	15.3	40.3	Improved 2.63 times
Precision	0.80	0.81	+0.01
Recall	0.91	0.88	−0.03
F1-Score (Val.)	0.83	0.81	−0.02

The 2% F1 degradation is substantially in range of 1% to 5% drop typically reported for convolutional detection models in the quantisation literature [52], attributable to the

compact weight space of the depthwise-separable convolution backbone, which is particularly amenable to 8-bit approximation. The precision–recall scatter plot (Figure 9) corroborates this, with Float32 and INT8 occupying near-identical positions across the full recall range.

In addition to the quantitative deployment metrics presented in Table 3, the proposed detector was evaluated under continuous live-video operation using the OpenMV Cam H7 in a natural brinjal field environment. The embedded system successfully processed real-time image streams and maintained processing rates of approximately 33 FPS during field operation. Representative deployment screenshots are presented in Figure 12, demonstrating successful fruit localization under natural canopy conditions. No execution interruptions or memory-overflow events were observed during the deployment trials, confirming the compatibility of the proposed framework with the hardware constraints of the OpenMV platform.



Figure 12. Real-time field deployment of the proposed FOMO detector on OpenMV Cam H7.

Although the deployment experiments demonstrated successful real-time execution under field conditions, several deployment-oriented characteristics were beyond the scope of the present study. Specifically, dedicated measurements of camera-to-decision latency, power consumption, thermal behaviour, and long-duration operational stability were not performed. Future work will focus on comprehensive hardware characterization and extended field testing to further assess the suitability of the proposed framework for autonomous agricultural applications.

3.6. Independent Test Set Performance

The final models were evaluated using an independent testing dataset that remained unseen during training and validation. The Float32 implementation achieved a precision of 0.85, recall of 0.88, and F1-score of 0.87. Similarly, the INT8 implementation achieved a precision of 0.86, recall of 0.87, and F1-score of 0.87. The close agreement between validation and testing performance indicates that the model generalizes well to previously unseen samples and that post-training quantization did not adversely affect detection

capability. Furthermore, real-time deployment experiments conducted on the OpenMV-embedded vision platform under natural field conditions confirmed the practical applicability of the proposed framework for embedded agricultural sensing applications.

The use of an independent testing dataset is important for assessing the generalization capability of machine-learning models because validation data may indirectly influence model development through hyperparameter tuning and training decisions. The consistent performance observed across validation, testing, and real-time deployment experiments suggests that the proposed framework did not suffer from substantial overfitting and is capable of maintaining reliable detection performance on previously unseen field images.

3.7. Fruit Counting and Missed-Fruit Analysis

In addition to detection performance, the fruit-counting capability of the proposed FOMO framework was evaluated using 43 independent test images. Because each activation cluster within the generated heatmap corresponds to a detected fruit instance, the framework inherently supports simultaneous localization and counting as presented in Figure 13. The counting analysis yielded a Mean Absolute Error (MAE) of 1.65 fruits per image and a Root Mean Square Error (RMSE) of 2.16 fruits per image as presented in Table 4. Counting errors increased primarily in densely clustered fruit scenes and under severe foliage occlusion, where neighbouring fruits occasionally generated merged activation regions within the heatmap. Such behaviour is a known limitation of centroid-based localization approaches when multiple objects occupy adjacent heatmap cells.

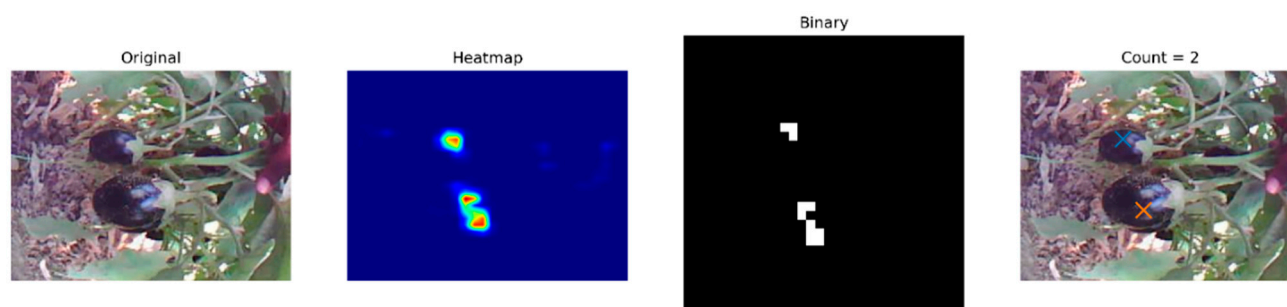


Figure 13. Illustration of the FOMO-based fruit localization and counting process. The original image is transformed into a probability heatmap followed by binary thresholding and centroid extraction. Each connected activation region corresponds to an individual fruit instance. In the illustrated example, two brinjal fruits were successfully localized and counted. FOMO performs object localization and counting through centroid prediction rather than conventional bounding-box regression.

Table 4. Quantitative evaluation of fruit-counting performance and missed-fruit detection characteristics of the proposed FOMO model using independent test images under natural field conditions.

Metric	Value
Test Images	43
MAE (fruits/image)	1.65
RMSE (fruits/image)	2.16
Float32 Missed-Fruit Rate (%)	7.64
INT8 Missed-Fruit Rate (%)	7.74

The counting analysis demonstrated that the proposed FOMO framework can simultaneously localize and enumerate brinjal fruits through centroid prediction. However, counting performance decreased in densely clustered fruit scenes where neighbouring fruits occasionally produced overlapping activation regions. This limitation is inherent to

centroid-based localization frameworks and has been previously reported for FOMO systems. Increasing input image resolution or heatmap resolution may further improve object separation and counting performance in future studies.

Occlusion remains one of the principal challenges for machine-vision-based harvesting systems operating in natural crop canopies. In the present study, missed detections were predominantly associated with partial or severe fruit occlusion by leaves, stems, and neighbouring fruits. Such behaviour is not unique to the proposed framework and has been widely recognized as a fundamental challenge in object detection systems operating under field conditions. Heatmap-based localization architectures such as FOMO are particularly sensitive to overlapping object centroids because localization is performed through spatial activation maps rather than explicit object-boundary estimation. Nevertheless, the proposed framework maintained high precision and low false-positive rates under field conditions while achieving real-time performance on highly resource-constrained embedded hardware. Future work will focus on improving occlusion robustness through higher-resolution heatmaps, multi-view image acquisition, temporal information fusion, and RGB-D sensing approaches that can exploit depth cues to distinguish partially hidden fruit instances.

3.8. Suitability for Field Deployment

At 40.3 FPS and 239.4 KB RAM, the INT8 model is directly deployable on microcontroller-class platforms widely used in precision agriculture (e.g., Arduino Nicla Vision, STM32-based vision modules) without companion processing hardware, providing substantial headroom for preprocessing, wireless transmission, and pipeline latency. The primary field limitation is sensitivity to severe occlusion. In densely planted rows where adjacent fruit clusters overlap at the scale of the 32×32 heatmap grid, multiple fruit may be resolved as a single centroid activation, causing systematic under-counting in high-density configurations. Future work should investigate higher-resolution FOMO variants (e.g., 320×320 or 416×416 input) or attention-augmented backbone modifications to improve spatial resolution without exceeding the memory and latency constraints of the target-embedded platform.

The significance of the present work lies in demonstrating the feasibility of agricultural object detection on microcontroller-class hardware. Unlike conventional detectors that are typically deployed on GPU-equipped edge computers, the proposed framework was successfully deployed on an OpenMV-embedded vision platform. Deployment profiling showed that INT8 quantization reduced peak RAM requirements from 887.2 KB to 235.6 KB while maintaining comparable detection performance.

The observed reductions in memory utilization and inference latency can be attributed not only to parameter quantization but also to the optimized inference stack provided by TensorFlow Lite Micro and ARM CMSIS-NN. CMSIS-NN provides highly optimized fixed-point neural network kernels for Cortex-M processors, enabling efficient execution of quantized convolutional and depthwise convolutional operations while minimizing memory overhead. Although the deployment framework incorporates TensorFlow Lite Micro and ARM CMSIS-NN acceleration libraries, low-level implementation details such as quantization granularity (per-tensor versus per-channel quantization), kernel selection, and processor-specific SIMD utilization are internally managed by the deployment toolchain and were not independently controlled within the scope of the present study. Future work will investigate these hardware-level optimization mechanisms in greater detail to further characterize their contribution to inference efficiency.

Such characteristics are particularly important for agricultural sensing systems operating under strict power, memory, and cost constraints. FOMO was specifically designed for these deployment scenarios and can operate using substantially lower computational

resources than conventional object detection architectures [26]. A limitation of the present study is that detection performance decreases under severe fruit–foliage occlusion and dense fruit clustering, which may result in missed detections. Although this limitation is inherent to centroid-based heatmap localization approaches, future integration of higher-resolution localization maps and depth-assisted sensing is expected to improve robustness in complex canopy environments.

The selection of a 256×256 input resolution reflects a deliberate trade-off between localization fidelity and computational efficiency. Higher image resolutions can preserve additional spatial detail and improve the detection of small or partially occluded fruits; however, they also increase memory consumption and inference latency, potentially compromising deployment feasibility on microcontroller-class hardware. The proposed configuration was therefore optimized for real-time-embedded operation, although future work will investigate higher-resolution FOMO configurations and alternative lightweight architectures to further improve robustness under severe occlusion and small-object conditions.

The selection of the proposed FOMO architecture was primarily governed by the hardware constraints of the OpenMV Cam H7 platform based on the STM32H743 microcontroller, which provides 2 MB Flash memory and 1 MB SRAM for the complete-embedded application, including the TensorFlow Lite Micro inference engine, tensor arena, image buffers, firmware, and neural network parameters. Embedded deep learning deployment on microcontrollers is fundamentally constrained by limited memory, computational capability, and power availability, requiring model architectures to be selected according to hardware resources rather than detection accuracy alone, while lightweight inference frameworks such as TensorFlow Lite Micro are specifically designed to address these constraints [25]. Recent benchmarking studies have further demonstrated that deploying modern YOLO-based object detectors on MCU-class platforms remains challenging because model selection must simultaneously satisfy stringent constraints on Flash memory, SRAM utilization, inference latency, and detection accuracy. In particular, the MCUBench benchmark evaluated more than 100 YOLO-based object detection models across seven microcontroller platforms, highlighting that even highly optimized YOLO architectures require careful architecture scaling and memory optimization to satisfy MCU resource limitations [53,54]. Likewise, TinyissimoYOLO, specifically developed for ultra-low-power microcontrollers, employs approximately 422 k trainable parameters, requires less than 0.5 MB memory for INT8 weights, and achieves deployment feasibility through quantization-aware training and dedicated network optimization [55]. Similarly, microYOLO demonstrated that YOLO-based object detection on Cortex-M microcontrollers is feasible only after substantial architectural simplification, requiring less than 800 KB Flash, less than 350 KB RAM, and achieving approximately 3.5 FPS during inference [56]. In comparison, the proposed INT8 FOMO model occupied only 0.11 MB of Flash storage with a runtime memory requirement of 239.4 KB, leaving sufficient on-chip memory for image acquisition, TensorFlow Lite Micro inference, and application logic while maintaining satisfactory field detection performance. Therefore, the selection of the FOMO architecture in the present study was primarily motivated by its compatibility with the stringent memory and computational constraints of the STM32H743 microcontroller rather than by claims of superior detection accuracy over alternative lightweight object detection architectures.

3.9. Positioning of the Present Work Relative to Existing Literature

To better position the contribution of the present work relative to existing literature, Table 5 compares representative agricultural object detection and TinyML deployment studies. Unlike most agricultural detection studies that primarily emphasize detection

accuracy using computationally intensive architectures, the present work focuses on deployment feasibility, memory utilization, inference latency, quantization effects, and field validation on microcontroller-class hardware. The comparison highlights that the contribution of this study lies in deployment-oriented evaluation and embedded implementation rather than the development of a new detection architecture.

Table 5. Positioning of the present study relative to representative agricultural object detection studies, TinyML object detection research, and embedded deployment frameworks.

Study	Domain	Detection Framework	Embedded Deployment	Quantization	Hardware Validation	Main Focus
Badgujar et al. (2024) [57]	Agricultural object detection review	YOLO family	Not primary focus	Not primary focus	Not primary focus	Comprehensive review of YOLO applications in agriculture.
Khan et al. (2025) [58]	Agricultural object detection review	Multiple architectures (YOLO, Faster R-CNN, SSD, etc.)	Limited discussion	Limited discussion	Not primary focus	Comprehensive review of object detection in precision agriculture.
Moosmann et al. (2024) [59]	TinyML object detection	TinyissimoYOLO	Yes	QAT + INT8	Multiple microcontrollers	Ultra-lightweight quantized object detector for microcontrollers.
Deutel et al. (2024) [60]	TinyML object detection	microYOLO	Yes	Lightweight optimization	OpenMV H7-class MCU	Feasibility of YOLO-based object detection on microcontrollers.
Edge Impulse FOMO Documentation (2024) [61]	TinyML object detection	FOMO	Yes	INT8 supported	MCU deployment	Centroid-based object detection for highly constrained devices.
Present Study	Agricultural TinyML deployment	FOMO + MobileNetV2	Yes	Post-training INT8 quantization	OpenMV Cam H7	Brinjal detection under field conditions with systematic Float32–INT8 trade-off analysis and real-world deployment validation

The novelty of the present study should be interpreted from a deployment-oriented perspective rather than an architectural perspective. The FOMO architecture and post-training quantization techniques employed in this work are existing technologies within the TinyML ecosystem. However, recent reviews of agricultural object detection indicate that research is dominated by accuracy-oriented studies using computationally intensive architectures such as YOLO, Faster R-CNN, and SSD, whereas deployment on microcontroller-class hardware remains comparatively underexplored. Table 5 positions the present work relative to representative agricultural object detection and TinyML deployment studies. Unlike conventional agricultural detection studies that primarily emphasize detection accuracy, the present work focuses on practical deployment constraints including model size, memory utilization, inference latency, quantization effects, and field validation on the OpenMV Cam H7 platform. Furthermore, while recent TinyML object detection studies have investigated microcontroller deployment using architectures such as microYOLO and TinyissimoYOLO, agricultural applications of embedded object detection

remain relatively limited. Therefore, the contribution of the present study lies in demonstrating the feasibility of deploying a lightweight brinjal detection framework on highly resource-constrained hardware and systematically quantifying the associated accuracy–efficiency trade-offs under real field conditions.

4. Conclusions

To address the practical challenge of deploying accurate object detection on resource-constrained-embedded hardware in agricultural environments, this study demonstrated the feasibility of adapting and deploying a FOMO-based TinyML lightweight deep learning object detection framework for real-time brinjal (*Solanum melongena*) detection under natural field conditions, built upon a truncated MobileNetV2 backbone adapted within the FOMO architecture. It can process 256×256 input images and produce spatially aware heatmap-based object predictions without bounding-box regression or non-maximum suppression. The proposed framework achieved a validation precision of 0.827, recall of 0.915, and F1-score of 0.869, with a perfect background true-negative rate of 1.00, confirming that all produced detections correspond to actual fruit instances with no false activations on non-fruit canopy regions. Post-training INT8 quantisation reduced inference time from 65.3 ms to 24.8 ms (40.3 FPS), while reducing RAM usage from 887.2 KB to 239.4 KB and model storage from 0.30 MB to 0.11 MB, at a validation F1 cost of only 2.0%. These results collectively demonstrate that the proposed framework satisfies both the lightweight and real-time objectives stated in this work, offering a deployment-ready solution for brinjal fruit monitoring on microcontroller-class-embedded platforms widely used in precision agriculture without requiring GPU-equipped field hardware. Although a 15% false negative rate under severe occlusion conditions and limited cross-variety generalization remain open challenges, future work will address these through higher-resolution backbone variants, attention-augmented feature extraction, and expanded multi-variety, multi-season datasets to further strengthen the practical applicability of the framework. The proposed framework addresses the perception stage of robotic harvesting by providing lightweight fruit localization on embedded hardware. However, additional modules including 3D localization, maturity assessment, grasp planning, obstacle avoidance, and end-effector validation are required for complete autonomous harvesting. Forthcoming research will investigate higher-resolution FOMO backbone variants, attention-augmented feature extraction mechanisms to improve occlusion handling, and expanded multi-variety, multi-season datasets to strengthen the generalisability of the proposed framework. Integration of the framework with autonomous harvesting platforms and deployment evaluation under fully uncontrolled field conditions will constitute the primary direction for applied validation.

Author Contributions: Conceptualization of research, writing—original draft, and visualization were performed by (A.P., P.K.S., T.K.K., D.K.K., M.A.H., R.G.). Investigation and Data curation (A.P., D.K.K., J.K.R., R.G.). Analysis of data and interpretation (A.P., D.K.K., J.K.R., M.A.H., R.A.P., N.C.P.). Reviewing and editing of manuscript (P.K.S., R.A.P., J.K.R., S.K.S., N.C.P.). All authors have read and agreed to the published version of the manuscript.

Funding: The author did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors for conducting this study.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to the data are part of an ongoing study.

Conflicts of Interest: The authors declare that they have no conflict of interest.

Nomenclatures/Abbreviations

Symbol	Description	Symbol	Description
X	Floating-point value	$q_{\max}$	Maximum integer limit
X_q	Quantized integer value	$q_{\min}$	Minimum integer limit
S	Scale factor used in quantization	TP	True Positive
Z	Zero-point in quantization	FP	False Positive
$X_{\max}$	Maximum floating-point value	FN	False Negative
$X_{\min}$	Minimum floating-point value	AP	Average Precision
N	Number of classes	mAP	Mean Average Precision
Abbreviation	Full Form	Abbreviation	Full Form
FOMO	Faster Objects, More Objects	FPS	Frames Per Second
CNN	Convolutional Neural Network	GPU	Graphics Processing Unit
YOLO	You Only Look Once	CPU	Central Processing Unit
Mask R-CNN	Mask Region-based Convolutional Neural Network	VRAM	Video Random Access Memory
Faster R-CNN	Faster Region-based Convolutional Neural Network	RGB	Red, Green, Blue
MobileNetV2	Lightweight Convolutional Neural Network Architecture	CLAHE	Contrast Limited Adaptive Histogram Equalization
SSD	Single Shot Detector	IoT	Internet of Things
AI	Artificial Intelligence	BHDS	Brinjal Harvesting Decision System
INT8	8-bit Integer Representation	ReLU	Rectified Linear Unit
Float32	32-bit Floating-Point Representation	Conv2D	2-Dimensional Convolution Layer
RAM	Random Access Memory	KB	Kilobyte
ms	Milli-seconds	MB	Megabyte

References

1. FAO. *FAOSTAT Statistical Database: Crops and Livestock Products*; Food and Agriculture Organization of the United Nations: Rome, Italy, 2024.
2. National Horticulture Board. *Horticultural Statistics at a Glance 2023*; National Horticulture Board: Gurugram, India, 2023.
3. Raigón, M.D.; Prohens, J.; Muñoz-Falcón, J.E.; Nuez, F. Comparison of eggplant landraces and commercial varieties for fruit content of phenolics, minerals, dry matter and protein. *J. Food Compos. Anal.* **2008**, *21*, 370–376. <https://doi.org/10.1016/j.jfca.2008.03.006>.
4. Hazra, P. Antioxidants and health benefits of brinjal. In *Vegetables for Nutrition and Entrepreneurship*; Springer Nature: Singapore, 2023; pp. 203–216.
5. Nahar, N.; Islam, M.R.; Uddin, M.M.; Jong, P.; Struik, P.C.; Stomph, T.J. Disease management in eggplant (*Solanum melongena* L.) nurseries also reduces wilt and fruit rot in subsequent plantings: A participatory testing in Bangladesh. *Crop Prot.* **2019**, *120*, 113–124.
6. Bhargava, A.; Bansal, A. Fruits and vegetables quality evaluation using computer vision: A review. *J. King Saud Univ.-Comput. Inf. Sci.* **2021**, *33*, 243–257.
7. Liakos, K.G.; Busato, P.; Moshou, D.; Pearson, S.; Bochtis, D. Machine Learning in Agriculture: A Review. *Sensors* **2018**, *18*, 2674. <https://doi.org/10.3390/s18082674>.
8. Ghosal, S.; Zheng, B.; Chapman, S.C.; Potgieter, A.B.; Jordan, D.R.; Wang, X.; Singh, A.K.; Singh, A.; Hirafuji, M.; Ninomiya, S.; et al. A Weakly Supervised Deep Learning Framework for Sorghum Head Detection and Counting. *Plant Phenomics* **2019**, 2019, 1525874. <https://doi.org/10.34133/2019/1525874>.
9. Bhargava, A.; Bansal, A.; Goyal, V. Machine learning-based detection and sorting of multiple vegetables and fruits. *Food Anal. Methods* **2022**, *15*, 228–242.
10. Karki, S.; Basak, J.K.; Tamrakar, N.; Deb, N.C.; Paudel, B.; Kook, J.H.; Kim, H.T. Strawberry disease detection using transfer learning of deep convolutional neural networks. *Sci. Hortic.* **2024**, *332*, 113241.
11. Koirala, A.; Walsh, K.B.; Wang, Z.; McCarthy, C. Deep learning—Method overview and review of use for fruit detection and yield estimation. *Comput. Electron. Agric.* **2019**, *162*, 219–234.

12. Zu, L.; Zhao, Y.; Liu, J.; Su, F.; Zhang, Y.; Liu, P. Detection and segmentation of mature green tomatoes based on mask R-CNN with automatic image acquisition approach. *Sensors* **2021**, *21*, 7842.
13. Chen, Y.; Lee, W.S.; Gan, H.; Peres, N.; Fraisse, C.; Zhang, Y.; He, Y. Strawberry yield prediction based on a deep neural network using high-resolution aerial orthoimages. *Remote Sens.* **2019**, *11*, 1584.
14. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
15. Liu, G.; Nouaze, J.C.; Touko Mbouembe, P.L.; Kim, J.H. YOLO-tomato: A robust algorithm for tomato detection based on YOLOv3. *Sensors* **2020**, *20*, 2145.
16. Qi, J.; Liu, X.; Liu, K.; Xu, F.; Guo, H.; Tian, X.; Li, Y. An improved YOLOv5 model based on visual attention mechanism: Application to recognition of tomato virus disease. *Comput. Electron. Agric.* **2022**, *194*, 106780.
17. Wang, H.; Dong, L.; Zhou, H.; Luo, L.; Lin, G.; Wu, J.; Tang, Y. YOLOv3-Litchi Detection Method of Densely Distributed Litchi in Large Vision Scenes. *Math. Probl. Eng.* **2021**, *2021*, 8883015.
18. Alaa, R.; Hussein, E.A.; Al-libawy, H. OBJECT DETECTION ALGORITHMS IMPLEMENTATION ON EMBEDDED DEVICES: CHALLENGES AND SUGGESTED SOLUTIONS. *Kufa J. Eng.* **2024**, *15*, 148–169.
19. Aira, J.; Olivares, T.; Delicado, F.M.; Vezzani, D. MosquitoIoT: A system based on IoT and machine learning for the monitoring of *Aedes aegypti* (Diptera: Culicidae). *IEEE Trans. Instrum. Meas.* **2023**, *72*, 1–13.
20. Nicolas, C.; Naila, B.; Amar, R.C. Tinyml smart sensor for energy saving in internet of things precision agriculture platform. In *2022 Thirteenth International Conference on Ubiquitous and Future Networks (ICUFN)*; IEEE: Piscataway, NJ, USA, 2022; pp. 256–259.
21. Darras, K.F.; Balle, M.; Xu, W.; Yan, Y.; Zakka, V.G.; Toledo-Hernández, M.; Wanger, T.C. Eyes on nature: Embedded vision cameras for multidisciplinary biodiversity monitoring. *bioRxiv* **2023**. <https://doi.org/10.1101/2023.07.26.550656>.
22. Andhe, D.; Kumar, S.A.; Patil, P.N.. Object detection at edge using TinyML models. *SN Comput. Sci.* **2023**, *5*, 11.
23. Farooq, A.; Imran, M. Innovating vending machine through object detection for currency recognition and enhanced cash handling. In *2023 7th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*; IEEE: Piscataway, NJ, USA, 2023; pp. 1–6.
24. Boyle, L.; Moosmann, J.; Baumann, N.; Heo, S.; Magno, M. DSORT-MCU: Detecting small objects in real time on microcontroller units. *IEEE Sens. J.* **2024**, *24*, 40231–40239.
25. Lin, J.; Chen, W.M.; Lin, Y.; Gan, C.; Han, S. MCUNET: Tiny deep learning on IoT devices. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 11711–11722.
26. Jongboom, J.; Impulse, E. FOMO: Real-Time Object Detection on Microcontrollers. **2022**. Available online: https://www.edge-ai-vision.com/wp-content/uploads/2022/05/E3W04_Jongboom_Edge_Impulse_2022.pdf (accessed on 20 January 2026).
27. Hayashi, S.; Ganno, K.; Ishii, Y. Machine vision algorithm of eggplant recognition for robotic harvesting. *J. Soc. High Technol. Agric.* **2000**, *12*, 38–46.
28. Hayashi, S.; Ganno, K.; Ishii, Y.; Tanaka, I. Robotic harvesting system for eggplants. *Jpn. Agric. Res. Q.* **2002**, *36*, 163–168.
29. Jian, S. Research on image-based fuzzy visual servo for picking robot. In *Proceedings of the 2008 International Conference on Computer and Computing Technologies in Agriculture*; Springer: Boston, MA, USA, 2008; pp. 751–760.
30. Miraei Ashtiani, S.H.; Golzarian, M.R.; Baradaran Motie, J.; Emadi, B.; Nikoo Jamal, N.; Mohammadinezhad, H. Effect of loading position and storage duration on the textural properties of eggplant. *Int. J. Food Prop.* **2016**, *19*, 814–825.
31. Kahya, E.; Ozduven, F.F.; Aslan, Y. YOLOv5 model application in real-time robotic eggplant harvesting. *J. Agric. Sci.* **2024**, *16*, 9.
32. Tamilarasi, T.; Muthulakshmi, P. Machine vision algorithm for detection and maturity prediction of Brinjal. *Smart Agric. Technol.* **2024**, *7*, 100402. <https://doi.org/10.1016/j.atech.2024.100402>.
33. Tamilarasi, T.; Muthulakshmi, P.; Ashtiani, S.-H.M. Smart Edge Computing Framework for Real-Time Brinjal Harvest Decision Optimization. *AgriEngineering* **2025**, *7*, 196. <https://doi.org/10.3390/agriengineering7060196>.
34. Tamilarasi, T.; Muthulakshmi, P.; Miraei Ashtiani, S.-H. Improved YOLO-based real-time brinjal detection algorithm for vision modules in harvesting robots. *Eng. Res. Express.* **2025**, *7*, 035234. <https://doi.org/10.1088/2631-8695/ade00>.
35. Tampu, I.E.; Eklund, A.; Haj-Hosseini, N. Inflation of test accuracy due to data leakage in deep learning-based classification of OCT images. *Sci. Data* **2022**, *9*, 580.
36. Sandhi, A.; Kumar, R.; Bhardwaj, R.; Kumar, D.; Rana, A.K.; Ajala, O.; Salau, A.O. Optimized deep learning framework for pomegranate disease detection using nature-inspired algorithms. *Plant Methods* **2025**, *21*, 124.

37. Shou, Y.; Meng, T.; Ai, W.; Xie, C.; Liu, H.; Wang, Y. Object detection in medical images based on hierarchical transformer and mask mechanism. *Comput. Intell. Neurosci.* **2022**, 2022, 5863782.
38. Al Sahili, Z.; Awad, M. The power of transfer learning in agricultural applications: AgriNet. *Front. Plant Sci.* **2022**, *13*, 992700.
39. Bhagat, M.; Kumar, D.; Kumar, S. Optimized transfer learning approach for leaf disease classification in smart agriculture. *Multimed. Tools Appl.* **2024**, *83*, 58103–58123.
40. Goyal, A.; Lakhwani, K. Integrating advanced deep learning techniques for enhanced detection and classification of citrus leaf and fruit diseases. *Sci. Rep.* **2025**, *15*, 12659.
41. Subramanian, M.; Shanmugavadeivel, K.; Nandhini, P.S. On fine-tuning deep learning models using transfer learning and hyper-parameters optimization for disease identification in maize leaves. *Neural Comput. Appl.* **2022**, *34*, 13951–13968.
42. Nigam, S.; Jain, R.; Marwaha, S.; Arora, A.; Singh, V.K.; Singh, A.K.; Immanuelraj, K. Automating yellow rust disease identification in wheat using artificial intelligence. *Indian J. Agric. Sci.* **2021**, *91*, 1391–1395.
43. Wu, H.; Judd, P.; Zhang, X.; Isaev, M.; Micikevicius, P. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv* **2020**, arXiv:2004.09602. <https://doi.org/10.48550/arXiv.2004.09602>.
44. Isenkul, M.E. Energy-aware deep learning for real-time video analysis through pruning, quantization, and hardware optimization. *J. Real-Time Image Process.* **2025**, *22*, 125.
45. Naveen, S.; Kounte, M.R. Memory optimization at edge for distributed convolution neural network. *Trans. Emerg. Telecommun. Technol.* **2022**, *33*, 4648.
46. Dotzel, J.; Wu, G.; Li, A.; Umar, M.; Ni, Y.; Abdelfattah, M.S.; Li, S. Fliqs: One-shot mixed-precision floating-point and integer quantization search. *arXiv* **2023**, arXiv:2308.03290. <https://doi.org/10.48550/arXiv.2308.03290>.
47. Lai, L.; Suda, N.; Chandra, V. CMSIS-NN: Efficient neural network kernels for Arm Cortex-M CPUs. *arXiv* **2018**, arXiv:1801.06601. <https://doi.org/10.48550/arXiv.1801.06601>.
48. Birla, L.; Bharadwaj, A.; Jain, R.; Deb, C.K.; Sehgal, V.K.; Ramasubramanian, R. Mango (*Mangifera indica*) tree detection and counting in mango orchard with satellite images using deep learning model YOLO: A comparative analysis. *Indian J. Agric. Sci.* **2025**, *95*, 678–683.
49. Akhtar, M.H.; Eksheir, I.; Shanableh, T. Edge-optimized deep learning architectures for classification of agricultural insects with mobile deployment. *Information* **2025**, *16*, 348. <https://doi.org/10.3390/info16050348>.
50. Cordova-Cardenas, R.; Amor, D.; Gutiérrez, Á. Edge AI in Practice: A Survey and Deployment Framework for Neural Networks on Embedded Systems. *Electronics* **2025**, *14*, 4877. <https://doi.org/10.3390/electronics14244877>.
51. Wang, Y.; Tian, H.; Zhou, Y.; Xiong, Y.; Li, Y.; Wang, M.; Yin, Y.; Guo, X.; Wu, J.; Zhang, J.; et al. Orchard-YOLO: A Robust Deep Learning Framework for Fruit Detection Complex Optical and Environmental Degradation. *Photonics* **2026**, *13*, 429. <https://doi.org/10.3390/photonics13050429>.
52. Nguyen, P.H.P.; Bui, Q.B.; Hoang, T. Q-CAD: Quantized convolutional accelerated detection via channel concatenation-based quantized inference for faster DDoS attack detection. *Int. J. Mach. Learn. Cyber.* **2025**, *16*, 10621–10642. <https://doi.org/10.1007/s13042-025-02790-y>.
53. David, R.; Duke, J.; Jain, A.; Reddi, V.J.; Jeffries, N.; Li, J.; Kreeger, N.; Nappier, I.; Natraj, M.; Wang, T.; et al. TensorFlow Lite Micro: Embedded Machine Learning for TinyML Systems. *Proc. Mach. Learn. Syst.* **2021**, *3*, 800–811.
54. Banbury, C.; Zhou, C.; Fedorov, I.; Navarro, R.M.; Thakker, U.; Gope, D.; Reddi, V.J.; Mattina, M.; Whatmough, P.N. MicroNets: Neural Network Architectures for Deploying TinyML Applications on Commodity Microcontrollers. *arXiv* **2020**, arXiv:2010.11267.
55. Moosmann, J.; Giordano, M.; Vogt, C.; Magno, M. TinyissimoYOLO: A Quantized, Low-Memory Footprint, TinyML Object Detection Network for Low Power Microcontrollers. In Proceedings of the 2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS), Hangzhou, China, 11–13 June 2023.
56. Sah, S.; Ganji, D.C.; Grimaldi, M.; Kumar, R.; Hoffman, A.; Rohmetra, H.; Saboori, E. MCUBench: A Benchmark of Tiny Object Detectors on MCUs. In Proceedings of the European Conference on Computer Vision (ECCV) Workshops, Milan, Italy, 29 September–4 October 2024.
57. Badgujar, C.M.; Poullose, A.; Gan, H. Agricultural object detection with You Only Look Once (YOLO) Algorithm: A bibliometric and systematic literature review. *Comput. Electron. Agric.* **2024**, *223*, 109090. <https://doi.org/10.1016/j.compag.2024.109090>.
58. Khan, Z.; Shen, Y.; Liu, H. Object Detection in Agriculture: A Comprehensive Review of Methods, Applications, Challenges, and Future Directions. *Agriculture* **2025**, *15*, 1351. <https://doi.org/10.3390/agriculture15131351>.
59. Moosmann, J.; Mueller, H.; Zimmermann, N.; Rutishauser, G.; Benini, L.; Magno, M. Flexible and Fully Quantized Ultra-Light-weight TinyissimoYOLO for Ultra-Low-Power Edge Systems. *arXiv* **2024**, arXiv:2307.05999.

60. Deutel, M.; Mutschler, C.; Teich, J. microYOLO: Towards Single-Shot Object Detection on Microcontrollers. In *ECML-PKDD Workshops Revised Selected Papers*; Springer: Cham, Switzerland, 2025; pp. 163–169.
61. Edge Impulse. FOMO (Faster Objects, More Objects) Object Detection Framework Documentation. Edge Impulse Documentation. 2024. Available online: <https://docs.edgeimpulse.com/studio/projects/learning-blocks/blocks/object-detection/fomo> (accessed on 22 July 2026).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.