

Article

Comparative Evaluation of Deep Learning Object Detectors for Embedded Weed Detection on Resource-Constrained Platforms

Nurtay Albanbay ¹, Yerik Nugman ^{2,3}, Mukhagali Sagyntay ^{2,3}, Azamat Mustafa ^{2,3,*}, Ramona Blanes ⁴, Algazy Zhauyt ⁵, Rustem Kaiyrov ⁶ and Nurgali Nurgozhayev ²

¹ Department of Cybersecurity, Information Processing and Storage, Satbayev University, Satbayev 22, Almaty 050013, Kazakhstan; n.albanbay@satbayev.university

² Department of Mechanical Engineering, Satbayev University, Almaty 050013, Kazakhstan; e.nugman@satbayev.university (Y.N.); mukhagali.sagyntay@gmail.com (M.S.); nurgali.nurgozhaev@gmail.com (N.N.)

³ RnD Center LLP, Almaty 050013, Kazakhstan

⁴ Adam Smith Business School, The University of Glasgow, Glasgow G11 6EY, UK; ramona.blanes@glasgow.ac.uk

⁵ Department of Aerospace and Electronic Engineering, Almaty University of Power Engineering and Telecommunications, Almaty 050013, Kazakhstan; a.zhauyt@aues.kz

⁶ Laboratory of Applied Mechanics and Robotics, Karaganda Buketov University, Karaganda 100001, Kazakhstan; kairov.rustem@mail.ru

* Correspondence: a.mustafa@satbayev.university

Abstract

Computer vision-based weed detection plays a critical role in agricultural robotics, enabling accurate, selective weeding. These systems operate on resource-constrained embedded platforms, which introduces a significant trade-off between accuracy and efficiency. This study presents a comparative evaluation of six detection models (YOLOv11n, YOLOv11s, SSD-Lite, NanoDet, Faster R-CNN, RT-DETR) for agro-robotic applications, measuring precision, recall, mAP@0.5, and runtime on low-power hardware. NanoDet achieved the highest detection accuracy (precision 98.6%, recall 94.2%, mAP@0.5 97.7%). YOLOv11s demonstrated similar performance (mAP@0.5: 96.1%) but required more computation. YOLOv11n provides the most favourable balance between accuracy and throughput (mAP@0.5: 94.6%, 207 FPS on a workstation). On Raspberry Pi 5, light models achieved 3–5 FPS. RT-DETR and Faster R-CNN exhibited high latency (3112–6500 ms/frame), which prevents real-time operation. NanoDet excelled in detection, while YOLOv11n provides the best balance between accuracy and efficiency for limited devices.

Keywords: agro-robotics; edge AI; real-time vision systems; low-power inference; precision agriculture; deep learning benchmarking



Academic Editors: Sotirios K. Goudos, Youyang Qu, Khandakar Ahmed and Zhiyi Tian

Received: 6 February 2026

Revised: 21 April 2026

Accepted: 24 April 2026

Published: 27 April 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

Weed control is still a key challenge in modern agriculture. Weed infestations can reduce crop yields by up to 34% and significantly increase production costs [1]. Recently, more land has been devoted globally to organic agriculture. This trend makes effective and safe weed management increasingly important [2]. Weed plants, as unwanted parts of agroecosystems, compete with crops for sunlight, water, and nutrients. This competition adversely affects crop growth and development. Thus, weed management remains a persistent and complex challenge for farmers [3].

Traditional weed control methods, such as manual labour and chemical herbicides, remain common. However, these methods have several major limitations. They require large time and labour investments. They are also linked to environmental and social risks. Manual weeding is labour-intensive, costly, and time-consuming [4]. Chemical treatments harm the environment and endanger the health of agricultural workers [5]. Recently, conventional agriculture has focused on reducing herbicide use due to its negative environmental effects. This includes soil contamination and ecosystem disruption [6,7]. Such efforts align with the UN Sustainable Development Goals. Relevant goals include Zero Hunger, Responsible Consumption and Production, Life on Land, and Industry, Innovation and Infrastructure [8]. Automated mechanical weed removal shows promise for reducing environmental stress. Labour shortages also remain a major challenge for modern agriculture [9,10]. Because of these limitations, intelligent robotic systems for automated crop weeding have become widespread in agriculture [11,12].

Automated robotic weed-control systems remain costly. Several factors contribute to their price: camera and sensor systems [13]; advanced computing, such as machine learning and deep neural networks [11,14]; precise navigation modules; and special devices for targeted weed removal [15]. Commercial solutions are expensive. Farming GT costs EUR 101,000–200,000 [16]; Ecorobotix Ara is USD 268,000–300,000 [17]; Imperador 4000 Eco Spray is about USD 285,000 [18]; and LaserWeeder G2 600 is USD 1.4 million [19]. As a result, many farms cannot afford these systems. This renders adoption challenging, especially for small and medium-sized producers.

For robotic weed control, models must perform real-time detection tasks with limited onboard computational resources. The effectiveness of site-specific interventions depends on inference latency and system stability. Practical studies have shown that weed detection and segmentation are feasible on embedded edge computing platforms. These studies highlight the need to optimise models under computational constraints [20–22].

On limited computational resources, modern object detectors exhibit substantial differences in architectural design and computational complexity. Single-stage models like the You Only Look Once (YOLO) family and Single Shot MultiBox Detector (SSD) are fast. Two-stage approaches, such as Faster R-CNN, are more accurate but require more computation. Transformer-based detectors, such as Real-Time Detection Transformer (RT-DETR), can balance quality and latency. Some studies compare YOLO and RT-DETR for weed detection and spraying [23,24]. However, most work focuses on individual YOLO variants or stand-alone architectures. Unified comparisons across detector families under a shared “accuracy–speed” criterion are rare [23–25].

Recent studies focus on easy-to-deploy models and weed detection. For example, Jia et al. (2024) proposed ADL-YOLOv8, a YOLOv8 variant that improved small-weed detection with a smaller, faster model [26]. Wang et al. (2025) developed YOLO-Weed Nano, an easier-to-run YOLOv8n model for weed detection in cotton fields [27]. They reported fewer model components and less computational cost, with high accuracy. For soybeans, Sulzbach et al. (2025) tested smaller YOLOv8 models [28]. Tiny versions, such as nano and femto, work well for real-time detection even under strict computational constraints. Studies on transformer-based models have started, too. Jin et al. (2025) proposed PHRF-RTDETR, which used less computational power and fewer components while remaining accurate [29]. Deployment studies stress working with on-device and portable hardware, including NPU speed-ups and models for mobile devices. These findings indicate that practical field deployment requires both high accuracy and low-latency inference, as well as model and hardware fit. However, most recent work focuses on just one type of model or setting at a time. There is still a lack of equal testing across different models under the same conditions.

This study presents a comparative evaluation of lightweight, efficient deep learning models that run in real time. These include YOLOv11n, YOLOv11s, Single Shot MultiBox Detector Lite (SSD-Lite), NanoDet, Faster R-CNN, and RT-DETR. The aim is to find models that achieve an optimal balance between weed detection accuracy and efficiency for use on resource-constrained embedded platforms such as the Raspberry Pi 5. The results provide insight into promising architectures for more practical, cost-effective robotic weed control.

2. Materials and Methods

2.1. Hardware Setup of the Agricultural Robot

In this work, a robotic hardware setup for selective weed removal was developed and investigated. The system hardware consists of a mobile platform, a vision sensing module, an on-board computing unit, and a mechanical weeding actuator based on a Delta-type manipulator.

As illustrated in Figure 1, the hardware architecture follows a hierarchical and sequential data-processing pipeline. The vision sensing module acquires a top-view red–green–blue (RGB) video stream (red, green, and blue), which is processed by the on-board computing unit to perform weed detection and target localisation. Based on the detection results, control commands are generated and transmitted to the corresponding hardware control units, including the Delta manipulator controller and the mobile base motion controller, enabling accurate positioning and selective weed removal.

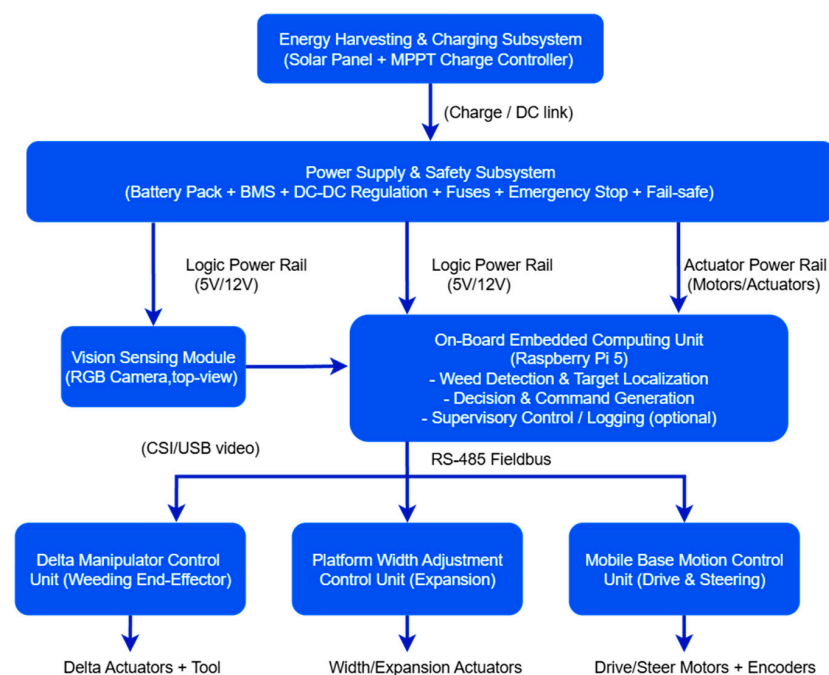


Figure 1. Block diagram of the proposed robotic weeding system.

The power supply, safety, and energy-harvesting subsystems provide electrical power and operational reliability to all hardware modules. The overall hardware block diagram of the proposed robotic system is shown in Figure 1.

2.1.1. Robotic Platform and Mechanical Structure

The agricultural robot was designed as an experimental platform for validating the proposed vision-based weeding system, with emphasis on positioning accuracy and on the integration of sensing, inference, and actuation modules. The chassis is based on a modular

frame with an adjustable wheelbase, allowing adaptation to different row spacings without mechanical reconfiguration.

The proposed agricultural robot (Figure 2) is a wheel-based mobile platform (1) designed for selective weed removal under field conditions. System operation is based on artificial intelligence and computer vision, implemented in an on-board electronic control unit housed within a control cabinet (2), and supported by an RGB camera (3).

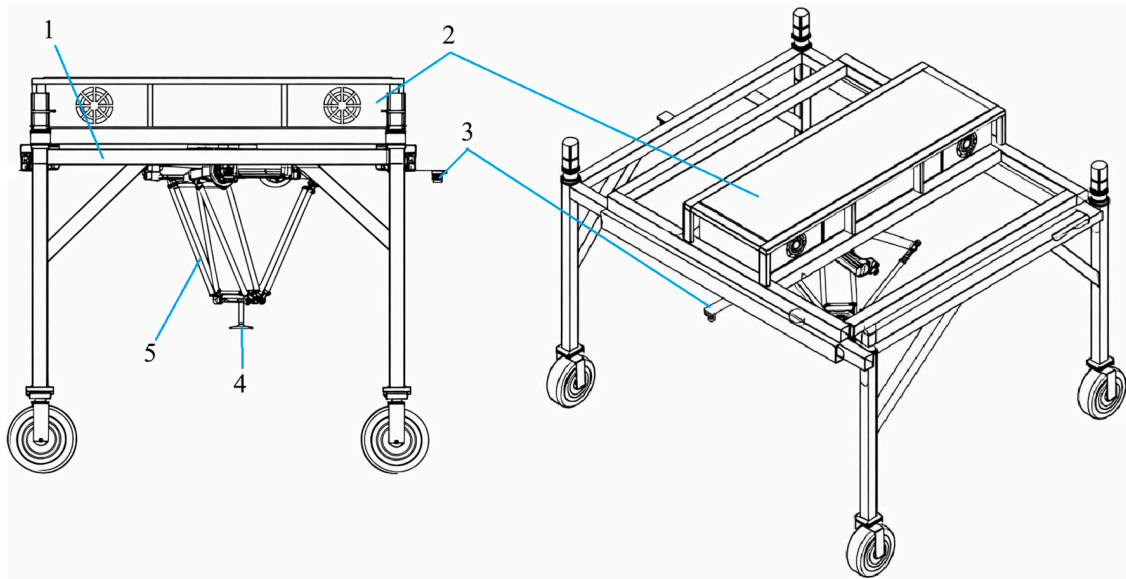


Figure 2. Weeding robot with an adjustable wheelbase width.

The robot is equipped with a mechanical weed-cutting tool (4), which is precisely positioned using a Delta-type robotic manipulator (5), enabling accurate and selective weed removal while minimising mechanical impact on crop plants.

The overall system integrates locomotion, perception, decision-making, and actuation within a unified hardware architecture. Optionally, a solar panel can serve as an auxiliary or primary power source to enhance energy autonomy during field operations. The prototype of the weeding robot is shown in Figure 3.

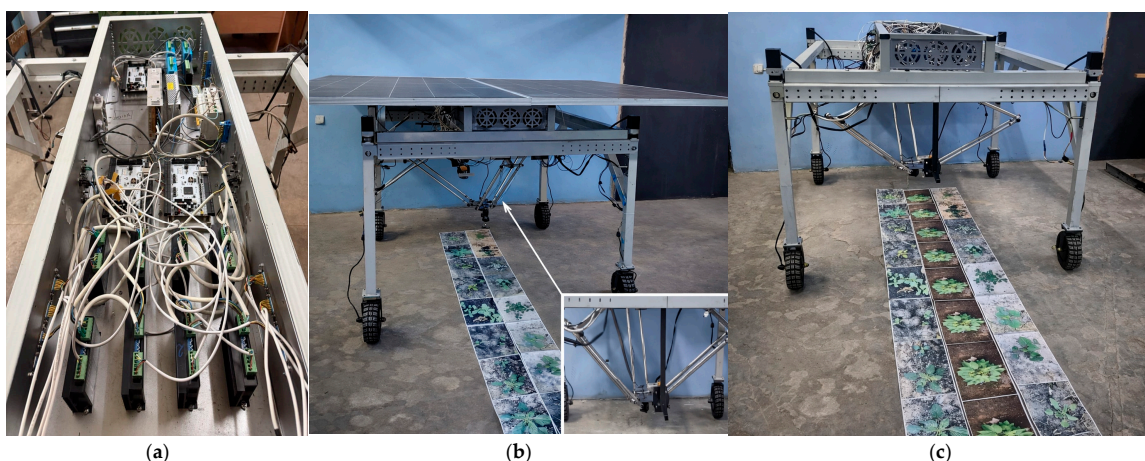


Figure 3. Prototype of the weeding robot and experimental setup of the agricultural robot during laboratory validation. (a) control cabinet housing the onboard electronics and power supply; (b) overall view of the robotic weeding platform; (c) laboratory experimental setup with an artificial weed-crop layout used for detection experiments.

2.1.2. Vision Sensing Module

The vision sensing module captures images of the treatment area and provides input data for weed-detection algorithms. The module uses an RGB camera mounted on the robotic platform and oriented downward, enabling top-view imaging of the soil surface and vegetation (Figure 2). In this study, the RGB camera was the Raspberry Pi Camera Module V2, equipped with an 8-megapixel Sony IMX219 sensor (Raspberry Pi Ltd., Cambridge, UK). This viewing configuration simplifies object localisation and reduces the effect of perspective distortions during robot motion.

Image acquisition is performed with fixed camera mounting parameters (height above the ground and orientation), ensuring stable frame geometry and reproducible measurements. The camera video stream is used for subsequent processing and for extracting frames supplied to the detection model.

2.1.3. Embedded Computing Platform for On-Board Inference

The on-board computing module of the robotic system is implemented using a Raspberry Pi 5 Model B equipped with 8 GB of LPDDR4X memory (Raspberry Pi Ltd., Cambridge, UK), which is installed directly on the robot. It processes data acquired from the vision sensing module locally and executes weed-detection models in autonomous operation mode. The inference results are generated on board and transmitted to the robot control subsystem for subsequent decision-making and actuation. This design supports practical deployment on a mobile robotic platform under constrained computational resources.

2.1.4. Control Command Generation Based on Detection Results

For each incoming video frame, the detection model outputs bounding boxes and confidence scores for the identified weeds. The detections are then filtered using a confidence threshold, and duplicate predictions are removed using non-maximum suppression (NMS). The target point for the weed removal action is defined as the centre of the selected bounding box for the weed.

To guide the end-effector, the target point is transformed from image pixel coordinates to robot coordinates using a camera-to-robot calibration procedure. Based on the resulting coordinates, a control command is generated for the Delta-type manipulator to position the tool and perform mechanical weed removal.

2.2. Deep Learning Models Evaluated

This study evaluates six deep learning architectures selected to represent a broad spectrum of accuracy–efficiency trade-offs relevant to real-time weed detection on resource-constrained agro-robotic platforms. The models include four lightweight detectors—YOLOv11n, YOLOv11s, SSD-Lite, and NanoDet—as well as two computationally heavier baselines, Faster R-CNN and RT-DETR, which serve as benchmarks for classical and transformer-based detection paradigms.

YOLOv11n and YOLOv11s utilise enhanced feature-extraction blocks and optimised detection heads, enabling robust identification of small and visually similar weed classes while maintaining suitability for near-real-time inference. SSD-Lite, a simplified single-stage detector, is included as a lightweight baseline due to its low computational cost and frequent use in mobile computer vision tasks. NanoDet incorporates a compact, highly efficient architecture, making it particularly attractive for deployment on embedded hardware with tight energy and memory constraints.

In contrast, Faster R-CNN is a two-stage detection pipeline designed for high accuracy but known to incur substantial computational overhead, serving as a reference point for understanding the limitations of classical detectors in embedded environments.

RT-DETR introduces transformer-based attention mechanisms for object detection, offering strong representational power but typically requiring significant processing resources, thereby enabling a comparative assessment of contemporary architectures under embedded constraints.

All six models were trained on the same dataset, using shared hyperparameters and a unified augmentation pipeline to ensure a fair comparison. Model performance was evaluated using standard object detection metrics, including precision, recall, F_1 -score, mean Average Precision (mAP@0.5), and inference speed on embedded hardware [30].

2.3. Dataset Description

This study utilises the publicly available CottonWeedDet12 dataset (Zenodo ID: 7535814), which contains real-world imagery collected in commercial cotton fields. Images were captured under natural field illumination using smartphones and hand-held digital cameras between June and September 2021 in cotton production systems in the southern United States. The dataset captures substantial visual variability in lighting, background structure, canopy density, and weed appearance, providing a realistic basis for evaluating weed-detection models in agricultural environments. CottonWeedDet12 comprises 5648 RGB images with 9370 manually annotated bounding boxes and class labels. The dataset contains annotations for 12 weed classes (class names are provided in the dataset documentation).

The 12 weed classes exhibit varying degrees of visual similarity in leaf shape, texture, plant size, and growth stage, which makes inter-class discrimination inherently challenging. In addition, the dataset contains mixed-species scenes, complex field backgrounds, and a considerable proportion of relatively small weed targets, which further increase the difficulty of detection and class separation. These characteristics make the dataset challenging for lightweight detectors and help explain the observed inter-class confusion, especially for visually similar or partially occluded weeds.

For training and evaluation, the dataset was partitioned into training, validation, and test subsets, following common machine learning practice, with proportions of 70%, 20%, and 10%, respectively, to enable robust model development and unbiased performance assessment [31]. This division supports reliable performance estimation while mitigating the risk of overfitting. Prior to model training, all images were resized to 640×640 pixels, and an extensive augmentation pipeline was applied to enhance model robustness under field conditions. The augmentation techniques included random rotation, horizontal flipping, scaling, contrast and brightness adjustments, and Gaussian noise injection. These transformations simulate natural photogrammetric variability, improving the system's resilience to illumination changes, occlusions, background clutter, and fine-grained textural differences.

Overall, the characteristics of the CottonWeedDet12 dataset provide a rigorous and representative benchmark for evaluating object detection models in agricultural settings. Its diversity and annotation quality enable objective comparison of algorithmic performance and support reliable assessment of model suitability for deployment on embedded agrobotic platforms.

2.4. Training Procedure

All models are trained with the same dataset split, input resolution, augmentation pipeline, and optimisation schedule to ensure fair and reproducible performance comparison. Before training, we normalised all images and applied a single augmentation approach: random rotation, horizontal flipping, scaling, brightness and contrast adjustment, and low-intensity Gaussian noise. These steps improve robustness to common changes in illumination, viewpoint, and background complexity in agricultural environments.

The detectors were trained for 300 epochs with AdamW, using an initial learning rate of 0.001 and a cosine schedule. The batch size was 16. Each model applied its standard loss for localisation and classification. Training was consistent across models for fair comparisons. Using 300 epochs ensured convergence in a unified setting. The 0.001 learning rate is the AdamW default for a balance between convergence and speed. The checkpoint with the best validation mAP@0.5 was automatically saved. After each epoch, precision, recall, F_1 -score, and mAP@0.5 were computed to monitor convergence and check for overfitting. All models used identical data splits, augmentation, and optimisation parameters, ensuring unbiased comparison of detection accuracy and efficiency.

Training was conducted on a Windows 11 (64-bit) workstation using an NVIDIA GeForce RTX 3070 GPU, AMD Ryzen 9 5900HX CPU (3.30 GHz), and CUDA 11.x. PyTorch 2.6.0 was the primary framework for model implementation and training.

All models were trained using unified hyperparameters under identical conditions, enabling fair and consistent comparison. While individual hyperparameter optimisation could improve model-specific performance, it would introduce variability and reduce comparability across architectures.

The evaluated models use different architectures: lightweight single-stage detectors (YOLOv11n, SSD-Lite, NanoDet), which make predictions in one step; two-stage detectors (Faster R-CNN), which first generate region proposals and then classify them; and transformer-based methods (RT-DETR), which utilise attention mechanisms for detection. These architectures differ in how they extract features, structure detection heads, and manage computation. By using a unified training setup, we can compare models in a controlled way and isolate the impact of architecture on performance.

2.5. Performance Metrics

Model performance in this study is assessed using standard detection metrics and computational efficiency measures. Precision and recall, defined in Equations (1) and (2), measure the accuracy of positive predictions and the model's sensitivity to target instances. These indicators are combined into the F_1 -score (Equation (3)), which balances both measures into a single metric. To broaden assessment across Intersection over Union (IoU) thresholds, the mean Average Precision (mAP) is used, as defined in Equation (4). Alongside accuracy metrics, computational cost is measured by floating-point operations (FLOPs, Equation (5)) and normalised to giga floating-point operations (GFLOPs, Equation (6)) to represent inference efficiency on embedded hardware. Definitions and mathematical formulations of these metrics are presented below.

Precision indicates the ratio of correctly predicted positive instances among all predicted positives. It is defined as:

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

Recall measures the proportion of correctly detected positive samples out of all actual positives. It is given by:

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

where TP , FP , and FN denote true positives, false positives, and false negatives, respectively.

The F_1 -score represents the harmonic mean of precision and recall, and provides a unified metric that captures both false positives and false negatives:

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$

The mean Average Precision (mAP) integrates both precision and recall across different IoU thresholds. It is computed as the mean of the areas under the precision–recall curves of all classes.

$$\text{mAP} = \frac{1}{T} \sum_{t=1}^T AP_t \quad (4)$$

where T is the total number of classes, and AP_t is the Average Precision for class t .

FLOPs represent the total number of arithmetic operations required for one forward pass of the model. For convolutional layers, FLOPs are computed as:

$$\text{FLOPs} = 2 \cdot C_{in} \cdot C_{out} \cdot K^2 \cdot H_{out} \cdot W_{out} \quad (5)$$

where C_{in} and C_{out} are the numbers of input and output channels, K is the kernel size, and H_{out} and W_{out} are the spatial dimensions of the output feature map.

To enable comparison between models and to reflect computational feasibility on resource-limited platforms, FLOPs are normalised to GFLOPs, defined as:

$$\text{GFLOPs} = \frac{\sum \text{FLOPs}}{10^9} \quad (6)$$

These metrics collectively detail model performance by showing the trade-off between accuracy and computational cost. This balance is crucial for deploying models on edge devices and embedded robotic systems.

3. Results

This section presents and analyses results from four lightweight weed detection models—YOLOv11n, YOLOv11s, SSD-Lite, and NanoDet—and two heavier baseline models: Faster R-CNN and RT-DETR. Models were evaluated on the test dataset under controlled imaging conditions. Performance was evaluated using standard metrics: precision (the proportion of correctly identified weeds among all identified weeds), recall (the proportion of correctly identified weeds among all actual weeds), F_1 -score (the harmonic mean of precision and recall), and mAP@0.5 (mean Average Precision at an Intersection over Union threshold of 0.5). Qualitative evaluation tools included confusion matrices (which show the number of correct and incorrect predictions for each weed class), Precision–Recall curves (which demonstrate the trade-off between precision and recall at varying thresholds), and F_1 -score–Confidence curves (which plot F_1 -scores across different confidence levels). These analyses clarify each architecture’s strengths and weaknesses. The results highlight challenges in detecting similar weed classes, the handling of small objects, and consistent predictions. They also show trade-offs between accuracy and efficiency, model adaptation to limited resources, and suitability for real-time deployment on embedded agricultural platforms. The next subsections detail model-specific outcomes and implications for practical agro-robotic weed detection systems.

3.1. Results of the Confusion Matrix

Figure 4a shows YOLOv11n’s confusion matrix accurately discriminates classes but misclassifies morphologically similar weeds. Designed for low-resource devices, YOLOv11n likely makes these errors due to limited architecture and capacity. Here, architecture refers to the model’s structure and pattern-learning ability.

The YOLOv11s model in Figure 4b demonstrates much higher accuracy and the lowest misclassification rate. The confusion matrix confirms consistent and reliable class discrimination. This advantage is clear in detecting small objects and differentiating between similarly sized weed species.

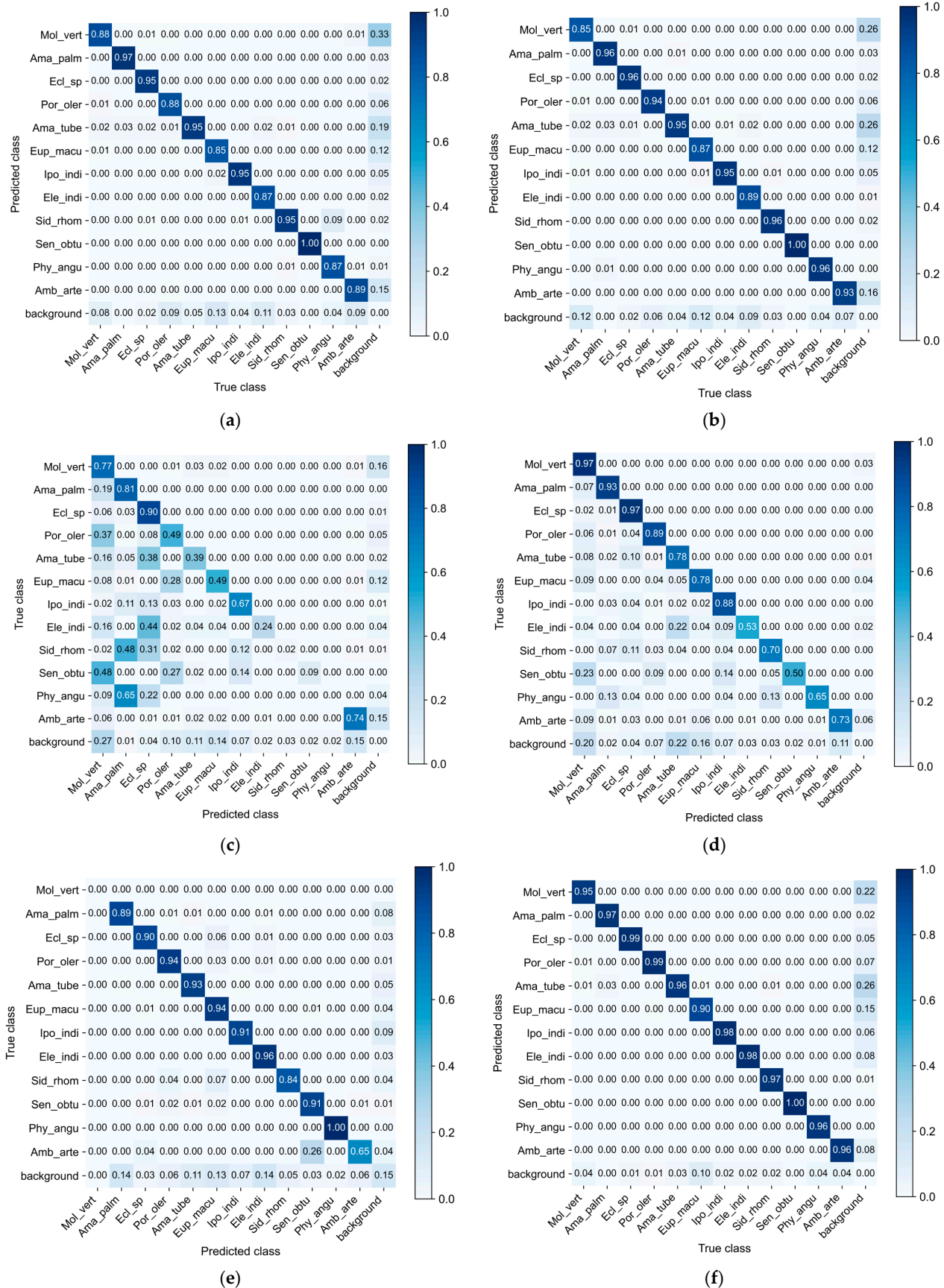


Figure 4. Confusion matrices of the evaluated models on the weed-detection dataset: (a) YOLOv11n; (b) YOLOv11s; (c) SSD-Lite; (d) NanoDet; (e) Faster R-CNN; (f) RT-DETR.

Figure 4c shows results for the SSD-Lite model. However, this architecture produces many false positives and negatives. Specifically, it often confuses targets as objects become smaller or leaves are partially hidden. Therefore, the model does not fully meet modern object-detection needs.

Figure 4d presents the NanoDet confusion matrix. Its strong diagonal indicates reliable weed-class discrimination. However, some off-diagonal entries indicate occasional misclassifications, mainly between similar species and, less often, between the background and other species. Overall, NanoDet achieves strong class separability, though most errors stem from fine-grained distinctions between similar categories.

The Faster R-CNN results are in Figure 4e. The confusion matrix indicates strong class separability for most weed categories, with dominant diagonal values. However, confusion remains for similar species and background-related predictions. Consequently, the two-stage proposal mechanism still misassigns under partial occlusion and clutter, highlighting the trade-off: better recall but higher computational cost and sensitivity to subtle inter-class similarities.

Figure 4f presents the confusion matrix for RT-DETR. The matrix has a strong diagonal, showing robust overall class discrimination and fewer off-diagonal errors than lighter models. However, occasional confusion arises between visually similar weeds and the background. This likely results from dataset-specific appearance overlap and the dependence of transformer-based attention on feature quality in challenging conditions.

3.2. Precision–Recall Curves

Precision–Recall curves reflect the overall robustness of the models in detection tasks and the trade-off between precision and recall. The PR curves of the six lightweight architectures are presented in Figure 5. These curves enable a comparative assessment of each model's sensitivity to false positives and false negatives, as well as its ability to distinguish between challenging classes.

The Precision–Recall (PR) curves in Figure 5 illustrate differences in the precision–recall trade-off across the six evaluated models. For YOLOv11n, the PR curve maintains relatively high precision across a broad range. However, as recall increases, a noticeable decrease in precision is observed for several challenging classes. This indicates difficulty in distinguishing morphologically similar weed species. The YOLOv11s model exhibits a consistently high PR curve across most operating regions. This is reflected in a larger area under the curve, suggesting a more stable balance between precision and recall across different decision thresholds. In contrast, the PR curve for SSD-Lite is substantially lower. Precision drops rapidly at high recall, suggesting greater susceptibility to false detections. The NanoDet model falls between YOLOv11n and YOLOv11s. While recall is generally well preserved, precision becomes less stable for certain difficult classes. This can be attributed to inter-class visual similarity. Faster R-CNN shows comparatively stable precision at higher recall levels. This indicates strong coverage of target objects under challenging conditions. RT-DETR also shows a high overall PR profile, reflecting favourable precision–recall characteristics. However, some degradation may still occur in regions where class appearance overlap is pronounced. Overall, the PR curves in Figure 5 provide a clear comparative view of class separability and detection robustness. SSD-Lite yields the weakest performance, whereas YOLOv11s, Faster R-CNN, and RT-DETR maintain higher performance across a wider range. YOLOv11n and NanoDet offer a moderate and generally stable trade-off.

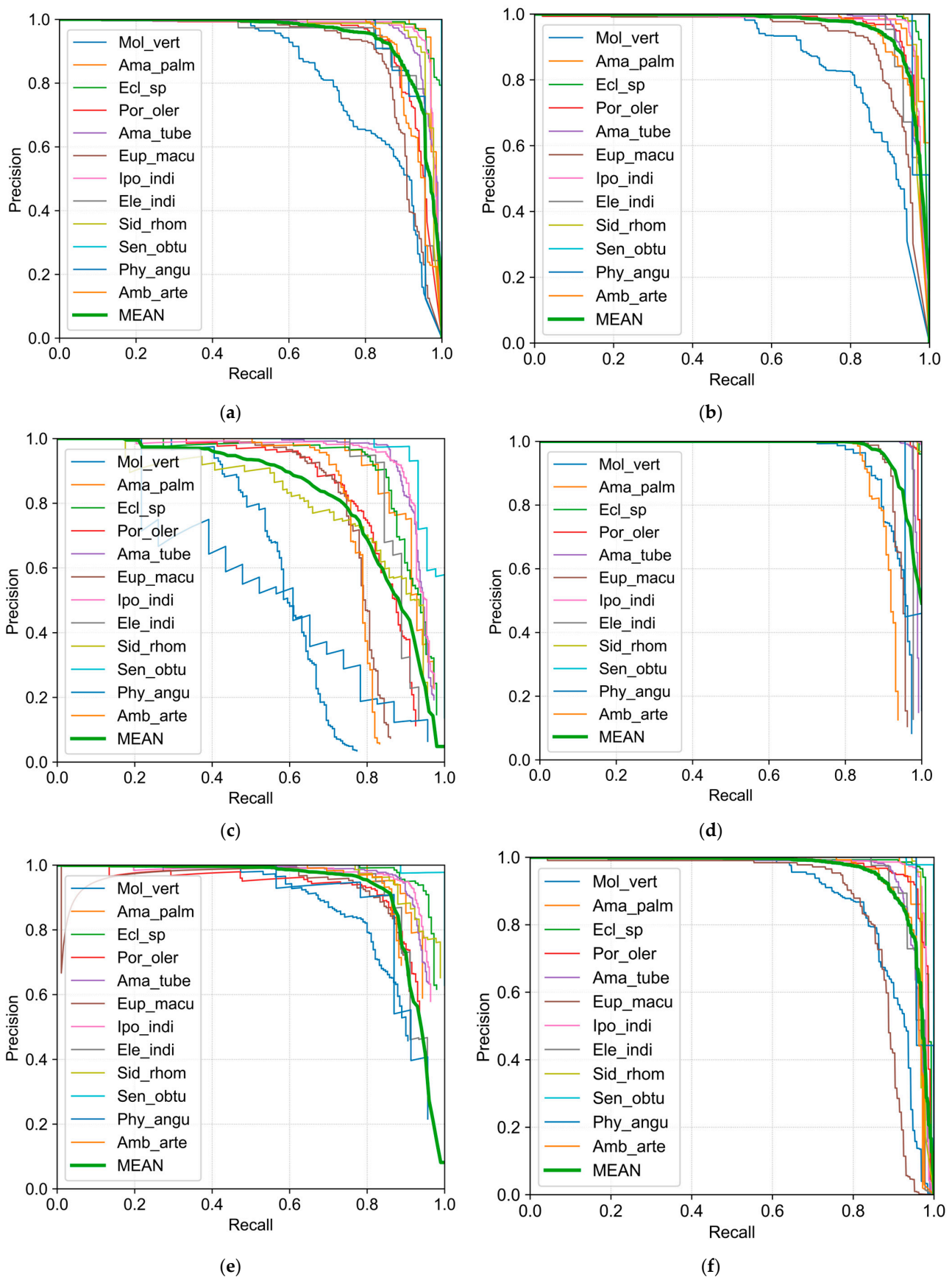


Figure 5. Precision–Recall (PR) curves for the evaluated detectors: (a) YOLOv11n; (b) YOLOv11s; (c) SSD-Lite; (d) NanoDet; (e) Faster R-CNN; (f) RT-DETR.

3.3. F_1 -Score–Confidence Curves

To evaluate the overall stability of the models with respect to confidence-threshold selection, F_1 -score–confidence curves were analysed. Figure 6 illustrates how the F_1 -score changes as the confidence threshold increases for the evaluated models. These curves provide insight into each model's behaviour when selecting an optimal confidence threshold for inference.

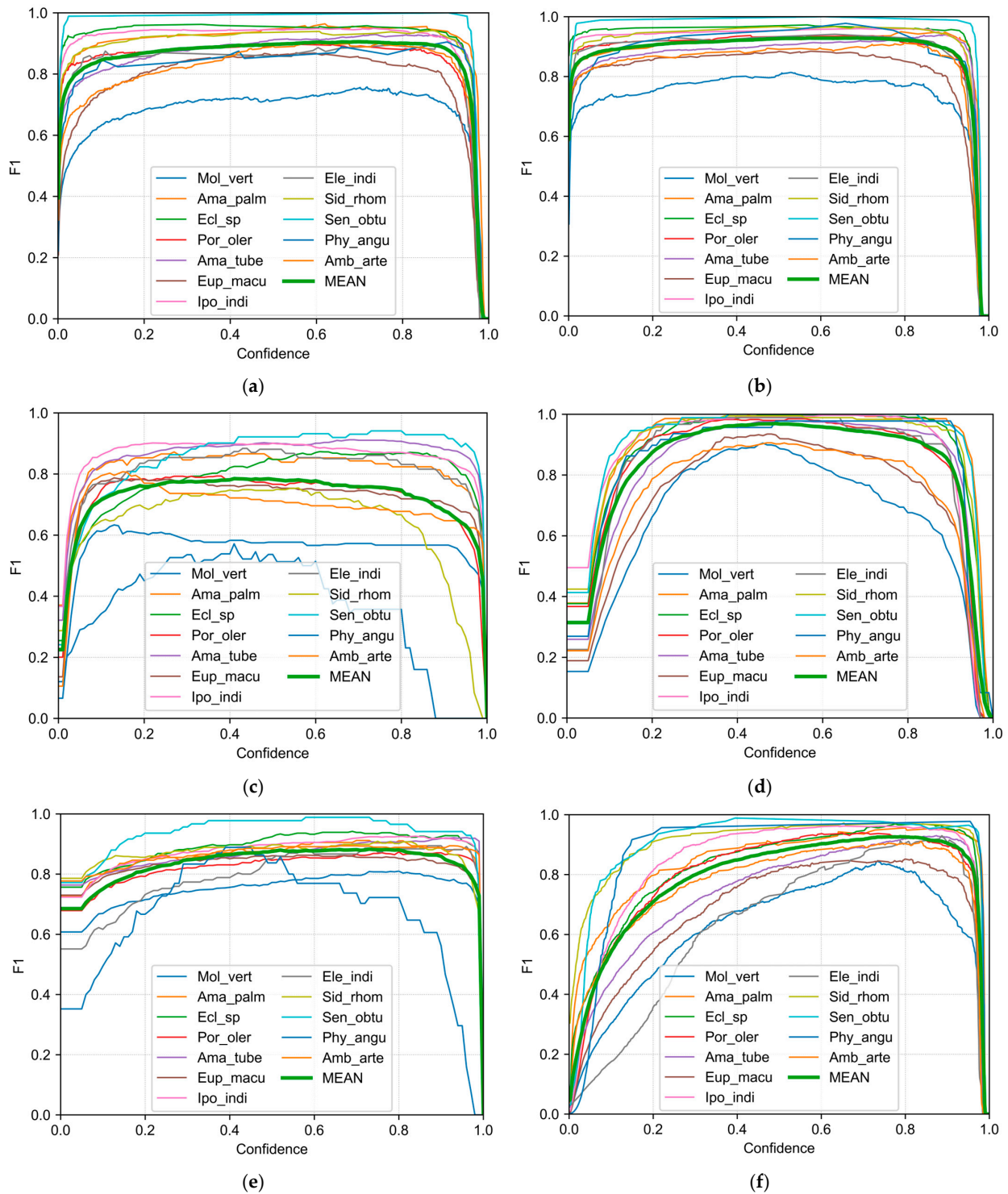


Figure 6. F_1 -score–confidence curves for the evaluated detectors: (a) YOLOv11n; (b) YOLOv11s; (c) SSD-Lite; (d) NanoDet; (e) Faster R-CNN; (f) RT-DETR.

For YOLOv11n, the F_1 -score is unstable at low confidence values, stabilises at intermediate thresholds, then declines at very high thresholds. YOLOv11s achieves the highest and most uniform F_1 -score profile, remaining stable across a wide range of confidence values and thus showing low sensitivity to threshold selection. In contrast, SSD-Lite has a lower overall F_1 -score and greater sensitivity to threshold changes, dropping sharply in the high-confidence regime. NanoDet offers more stability than YOLOv11n but less than YOLOv11s, with F_1 -scores decreasing at high thresholds for several classes. Both Faster R-CNN and RT-DETR maintain relatively high F_1 -scores across broad confidence thresholds, indicating stable performance. All models decline at extremely high thresholds. Overall, YOLOv11s, Faster R-CNN, and RT-DETR are the least sensitive to variations in the confidence threshold, whereas SSD-Lite is the most sensitive. This explicit comparison clarifies the relative performance and sensitivity of the models.

3.4. Model Performance Results

In this section, six models—YOLOv11n, YOLOv11s, SSD-Lite, NanoDet, Faster R-CNN, and RT-DETR—were evaluated for the weed detection task using quantitative and qualitative analyses. Table 1 summarises the quantitative results, listing for each model: detection accuracy (Precision, Recall, and mAP@0.5), computational load (GFLOPs), and inference throughput (FPS) measured on a reference workstation with identical input settings. The table does not include embedded inference performance on Raspberry Pi 5, which is reported separately in Section 3.5.

Table 1. Performance evaluation of weed-detection models on accuracy and computational load.

Model	Precision (%)	Recall (%)	mAP@0.5 (%)	FPS	GFLOPs
YOLOv11n	93.797	87.606	94.600	207.210	6.325
YOLOv11s	94.719	91.043	96.082	184.330	21.326
SSD-Lite	94.765	76.384	82.502	116.824	5.861
NanoDet	98.624	94.247	97.733	152.299	3.495
Faster R-CNN	86.113	91.397	92.027	29.807	267.961
RT-DETR	94.446	90.874	95.496	167.640	103.482

Table 1 highlights key findings: NanoDet achieved the highest detection accuracy (Precision 98.624%, Recall 94.247%, mAP@0.5 97.733%) with the lowest computational load (3.495 GFLOPs). YOLOv11s offered strong detection (mAP@0.5: 96.082%, Recall: 91.043%), but at a much higher load (21.326 GFLOPs). While SSD-Lite had lower accuracy (mAP@0.5 82.502%, Recall 76.384%), its GFLOPs were still low (5.861). Faster R-CNN achieved a high Recall (91.397%), but required the most computation (267.961 GFLOPs) and had the lowest throughput (29.807 FPS). RT-DETR (mAP@0.5 95.496%) combined high accuracy with significant computational load (103.482 GFLOPs) but decent throughput (167.640 FPS). YOLOv11n led workstation throughput (207.210 FPS) with moderate GFLOPs (6.325) and mAP@0.5 94.600%.

3.5. Embedded Platform Performance

To assess the practical suitability of neural models for deployment in an agricultural robot, their performance was experimentally evaluated on a Raspberry Pi 5 single-board computer. During the evaluation, the models' frame rate (FPS) and per-frame processing time (Detect_time) were measured. All models were exported to a unified inference format, and inference was performed using input images of the same resolution (640 × 640), ensuring an objective comparison. The quantitative results are summarised in Table 2.

Table 2. Embedded inference performance on Raspberry Pi 5.

Model	FPS	Inference Time (ms/Frame)
YOLOv11n	3.518	284.291
YOLOv11s	1.360	735.227
SSD-Lite	4.839	206.640
NanoDet	3.015	331.671
RT-DETR	0.321	3112.122
Faster R-CNN	0.154	6500.631

As shown in Table 2, SSD-Lite achieved the highest throughput (4.839 FPS) and the lowest per-frame processing time (206.640 ms). YOLOv11n reached 3.518 FPS (284.291 ms), while NanoDet achieved 3.015 FPS (331.671 ms). YOLOv11s showed lower throughput (1.360 FPS) and higher per-frame time (735.227 ms). RT-DETR and Faster R-CNN had the highest per-frame latency (3112.122 ms and 6500.631 ms) and the lowest throughput (0.321 FPS and 0.154 FPS).

These results indicate that GFLOPs alone do not fully determine real embedded speed: inference performance is also influenced by the internal operation graph, memory-access patterns, operator implementation efficiency, and the level of runtime optimisation. Nevertheless, in this benchmark, compact single-stage detectors (SSD-Lite and YOLOv11n) provided the most practical balance between speed and computational cost on Raspberry Pi 5, whereas RT-DETR and Faster R-CNN were unsuitable for real-time operation.

4. Discussion

The results demonstrate a clear, practically important trade-off between detection quality and deployability on resource-constrained embedded platforms. While several models achieve high accuracy on the test set, their feasibility for onboard inference differs substantially. This difference becomes evident under realistic edge-computing conditions (Raspberry Pi 5, 640×640).

4.1. Accuracy–Efficiency Trade-Off and Model Suitability

Among the evaluated detectors, NanoDet achieved the strongest overall accuracy (Precision 98.624%, Recall 94.247%, mAP@0.5 97.733%) and the lowest computational complexity (3.495 GFLOPs). This demonstrates that NanoDet is highly competitive for weed-detection tasks where accurate separation of similar classes is critical and modest embedded performance is sufficient. In contrast, YOLOv11s showed high performance (mAP@0.5 96.082%) and the most reliable class-wise separability in the confusion-matrix analysis. This suggests that its higher model capacity improves fine-grained discrimination. However, this robustness comes with a significant computational cost (21.326 GFLOPs), which limits its use on edge hardware.

YOLOv11n provides a strong practical compromise. Although its accuracy is slightly lower than that of NanoDet and YOLOv11s (mAP@0.5 94.600%), it delivers the highest inference speed among the tested models in the general benchmark (207.210 FPS) with relatively low complexity (6.325 GFLOPs). Such a balance is especially relevant for autonomous agro-robotic platforms where perception must run continuously alongside navigation, localisation, and control tasks. Therefore, within the evaluated CottonWeedDet12 scenario and Raspberry Pi 5 setup, YOLOv11n emerges as the most operationally reliable option for embedded robotic deployment.

Turning to classical heavy detectors, the results show their limitations for onboard deployment. Faster R-CNN has a strong recall (91.397%) but a very high computational overhead (267.961 GFLOPs). RT-DETR offers high accuracy (mAP@0.5 95.496%) but also

a large computational burden from transformer blocks (103.482 GFLOPs). These architectures may fit server-side processing or high-end GPU platforms, not lightweight edge computers in field robots. Practical, lightweight detectors operate at frame-level latencies of a few hundred milliseconds. For context, prior Raspberry Pi deployments report a pronounced accuracy–latency gap under CPU-constrained inference. For example, in [32], a streamlined YOLOv7 model achieved mAP = 96.89%, but inference time on a Raspberry Pi 4B was 1.18 s per image (<1 FPS), slower than the Raspberry Pi 5 latencies observed here at 640×640 . In contrast, Ref. [33] describes a Raspberry Pi 5 bud-detection system with mAP@0.5 = 86.0% (single-stage) and 85.0% (two-stage), and an end-to-end capture–inference–transmission cycle of under 3 min. This regime reflects periodic, not continuous, frame-by-frame processing.

Recent studies also confirm the importance of deployment-oriented evaluation for weed detection systems. Mwitta et al. (2024) evaluated three real-time-capable deep learning models for weed detection on an embedded platform (Jetson Xavier AGX) [21]. Their findings highlight that practical applicability is determined not only by detection accuracy but also by inference speed under embedded constraints. This observation aligns with the present results: lightweight architectures such as YOLOv11n and NanoDet offer a more favourable balance between detection quality and deployability than computationally heavier models. By contrast, recent 2025 studies have mainly focused either on architecture-specific improvements or on benchmarking within a single model family. For example, Raza et al. (2025) presented a standardised benchmark of 19 YOLO variants for cotton–weed detection, while García-Navarrete et al. (2025) compared YOLO and RT-DETR models for weed detection in maize under controlled greenhouse conditions [25,34]. Similarly, Yang et al. (2025) proposed an enhanced RT-DETR-based architecture for soybean weed detection [35]. Nevertheless, these studies do not provide a unified cross-architecture comparison combined with real embedded validation under common deployment constraints. Therefore, the present study differs from prior work by integrating unified training conditions, heterogeneous detector comparison, and practical embedded evaluation within a single deployment-oriented framework.

These performance differences can be further explained from the perspective of model architecture design. NanoDet employs a lightweight backbone and an efficient detection head, which reduces computational complexity while preserving effective feature representation, enabling high accuracy with low GFLOPs. YOLOv11n achieves a balance between accuracy and efficiency by optimising feature-extraction blocks that enhance representation capacity without significantly increasing computational cost. In contrast, SSD-Lite relies on a simplified architecture with reduced feature extraction, improving inference speed but limiting its ability to capture fine-grained details, resulting in lower recall, particularly for small or visually similar weed classes. These observations highlight the direct relationship between architectural design choices and the observed trade-offs between accuracy and efficiency.

These differences are consistent with the observed confusion-matrix and PR-curve behaviour. Models with stronger feature representation, such as NanoDet and YOLOv11s, are better able to preserve separability for small or visually similar weed classes, whereas simplified architectures such as SSD-Lite are more vulnerable to recall degradation under fine-grained inter-class similarity and partial occlusion. This interpretation is also consistent with recent deployment-oriented weed-detection studies, which report that lightweight detectors can differ substantially in practical accuracy depending on how efficiently they preserve discriminative features under constrained computation [21,25,34].

Within the evaluated CottonWeedDet12 scenario and Raspberry Pi 5 setup, NanoDet is particularly suitable for high-precision weed detection scenarios, whereas YOLOv11n is the

more practical option when embedded real-time deployment under constrained onboard resources is prioritised.

4.2. Robustness to Confidence Threshold Selection

Beyond mean accuracy metrics, deployment stability also depends on a detector's sensitivity to decision thresholds. The F_1 -score–confidence curves indicate that YOLOv11s maintains a stable and high F_1 -score over a broad confidence range, implying reduced sensitivity to threshold tuning. Such behaviour is important in practical field conditions, where illumination changes, partial occlusions, and background clutter may shift confidence distributions and affect prediction consistency.

By contrast, SSD-Lite is more sensitive to threshold changes, with a noticeable performance drop at higher confidence values. This behaviour may increase the risk of missed detections or unstable performance in real-time operation. NanoDet shows better threshold stability than SSD-Lite, although its F_1 -score still declines at high confidence levels for some classes. Faster R-CNN and RT-DETR also maintain relatively stable F_1 -scores over a broad confidence range, indicating lower sensitivity to threshold variation.

Overall, YOLOv11s, and to a lesser extent RT-DETR and Faster R-CNN, appear more robust to confidence-threshold selection, whereas SSD-Lite is the most sensitive among the evaluated models. This aspect is important for practical deployment under variable operating conditions, where model suitability depends not only on mean accuracy but also on stable behaviour across different confidence settings, as also emphasised in recent deployment-oriented weed-detection studies [21,34].

4.3. Embedded Deployment Reality: Latency Is Not Fully Predicted by GFLOPs

The Raspberry Pi 5 evaluation reveals that computational complexity (GFLOPs) does not fully determine real inference latency. Although NanoDet has the lowest GFLOPs, it did not achieve the highest embedded FPS (3.015 FPS, 331.671 ms). In contrast, SSD-Lite reached the best throughput (4.839 FPS, 206.640 ms) despite having higher GFLOPs than NanoDet. This discrepancy suggests that several factors influence performance. These include operator efficiency, memory access patterns, and runtime-level optimisations on ARM CPUs.

On ARM-based embedded CPUs, end-to-end inference latency depends not only on the nominal number of floating-point operations, but also on operator-level implementation efficiency, memory-access behaviour, and runtime support for the specific computation graph. As a result, models with lower GFLOPs may still exhibit higher latency if their operators are less efficiently optimised or if their execution involves greater data-movement overhead between intermediate feature maps.

This indicates that embedded performance is not determined solely by arithmetic complexity. It is also influenced by the interaction between model architecture and platform-specific execution efficiency. In other words, practical latency depends on how well the computational graph is supported by the target runtime, how intermediate feature maps are handled in memory, and how much overhead is introduced outside the core forward pass. Therefore, lower GFLOPs alone cannot be interpreted as sufficient theoretical evidence of lower embedded latency.

At 640×640 resolution, the achieved throughput on Raspberry Pi 5 (≈ 3 – 5 FPS for the best candidates) indicates that fully CPU-based inference is feasible for low-speed robotic motion. However, it may be insufficient for higher-speed platforms or for systems requiring multi-stream perception. Therefore, embedded deployment decisions should consider not only accuracy and GFLOPs. They should also account for runtime compatibility,

operator support, and system-level constraints such as concurrent navigation pipelines and communication overhead.

To further interpret the latency differences, the inference pipeline can be viewed as consisting of data preparation, model computation, and result formatting. On the Raspberry Pi 5, model computation is the main source of latency. SSD-Lite, with a simpler pipeline and efficient operations, has lower latency than NanoDet. NanoDet's extra steps and less-efficient operations increase its latency, despite lower GFLOPs. This demonstrates that both model design and system implementation affect overall latency.

The observed performance differences can be interpreted in terms of architectural design principles. Lightweight models prioritise computational efficiency by reducing feature map complexity and parameter count. This approach directly affects detection accuracy and recall. In contrast, more complex architectures improve feature representation and class separability but at the cost of increased computational overhead. These trade-offs explain the differences observed in accuracy, latency, and efficiency across the evaluated models. Similar discrepancies between nominal computational complexity and practical embedded latency have also been noted in recent deployment-oriented studies, highlighting the importance of platform-specific runtime behaviour [21,32].

4.4. Practical Implications for Autonomous Weed Control

From a practical deployment perspective, detector selection should be guided by the operational objective of the agricultural robot and the available onboard computing resources. If the main priority is to maximise detection quality and reduce missed weeds during selective treatment, models with higher detection performance, such as NanoDet or YOLOv11s, are more suitable. These models provide stronger class discrimination and are therefore preferable in scenarios where detection accuracy is more important than processing speed.

In contrast, for continuous onboard perception on low-power embedded platforms, the model must provide not only acceptable detection accuracy but also stable inference speed. In this case, YOLOv11n offers the most practical balance between accuracy and computational efficiency among the evaluated models. This makes it the most suitable option for embedded agro-robotic deployment under constrained hardware conditions.

Although SSD-Lite achieved the highest throughput on Raspberry Pi 5, its lower recall and mAP@0.5 indicate a higher risk of missed weeds. Therefore, its use may be justified only in scenarios where speed is prioritised over detection completeness.

These results indicate that model selection for autonomous weed control should not rely on accuracy alone. Instead, it should consider the intended deployment scenario, the acceptable level of missed detections, and the computational limitations of the target embedded platform. This practical model-selection logic is also consistent with recent embedded weed-detection studies, which emphasise that deployment suitability depends not only on detection accuracy but also on runtime feasibility under hardware constraints [21,34].

4.5. Limitations and Future Work

This study presents a controlled benchmarking of weed detection models for embedded deployment. However, several limitations should be noted. The experimental validation was conducted under laboratory conditions using artificial weed–crop layouts. This approach ensures reproducibility but does not fully reflect real field variability such as illumination changes, soil diversity, motion blur, and occlusions. The embedded evaluation was limited to a single hardware platform (Raspberry Pi 5), which does not capture the diversity of embedded devices used in agricultural robotics. In addition, power consumption was not directly quantified, although energy efficiency is an important practical factor for

low-power field deployment. Therefore, the reported deployment-oriented conclusions should be interpreted within the scope of the evaluated platform. The use of a unified training configuration ensures fair comparison, but may not capture architecture-specific optimal performance. Furthermore, the study does not include ablation experiments, since the focus is on model-level evaluation. Accordingly, the present conclusions should be interpreted as model-level comparative findings under a unified benchmarking setting, rather than as module-level causal attribution of performance differences. The results are obtained on a single crop-specific dataset and should be interpreted within this context. The study also does not include statistical or sensitivity analyses of input resolution and threshold settings.

Future work will extend the evaluation to real field conditions and multi-crop datasets. It will also include cross-platform benchmarking on heterogeneous embedded devices and incorporate detailed power profiling. Additional efforts will focus on architecture-specific optimisation and robustness enhancement. This may include domain adaptation and deployment-oriented techniques, such as region-of-interest inference and dynamic input resolution.

5. Conclusions

Autonomous weed control requires a detection module that performs accurately in offline evaluations and is deployable on embedded hardware. On such platforms, perception competes for resources with navigation, localisation, and control. The main contribution of this study is a deployment-oriented evaluation framework. It assesses weed detectors from two complementary perspectives: detection quality on a dedicated test set, and end-to-end inference behaviour on a resource-constrained embedded platform (Raspberry Pi 5). This combined assessment is essential. Models that perform well according to conventional accuracy metrics may fail to meet real-time constraints if platform limitations affect latency, throughput, and runtime stability.

The experimental results give a clear quantitative view of the accuracy–efficiency trade-off. NanoDet achieved the highest detection performance: 98.624% precision (correct positive predictions among all predicted positives), 94.247% recall (correct positive predictions among all actual positives), and 97.733% mAP@0.5 (mean Average Precision at 0.5 Intersection over Union), while requiring only 3.495 GFLOPs (billion floating-point operations). However, on Raspberry Pi 5, its embedded performance was 3.015 FPS (frames per second; 331.671 ms per frame), showing moderate real-time capability. In contrast, within the evaluated CottonWeedDet12 scenario, YOLOv11n offered the most practical balance between detection performance and embedded deployment efficiency. It achieved 94.600% mAP@0.5 and 3.518 FPS (284.291 ms per frame) on Raspberry Pi 5, and the highest workstation throughput (207.210 FPS). SSD-Lite had the highest embedded speed at 4.839 FPS (206.640 ms), but its lower detection accuracy—82.502% mAP@0.5 and 76.384% recall—limits its practicality. Heavy models like RT-DETR (0.321 FPS, 3112.122 ms) and Faster R-CNN (0.154 FPS, 6500.631 ms) had prohibitive latency, making them unsuitable for real-time deployment on the evaluated platform.

The embedded measurements show that GFLOPs alone are not a reliable proxy for end-to-end latency. Latency is also influenced by data movement overhead, code execution efficiency, and the costs of pre- and post-processing. Therefore, deployment choices should rely on platform-specific benchmarking rather than solely on offline metrics.

Future work will focus on edge optimisation and broader validation, including tests under real agricultural variability—illumination changes, occlusion, motion blur, and multi-crop environments—to improve generalisation.

Author Contributions: Conceptualization, N.A. and A.M.; methodology, N.A., Y.N. and M.S.; software, Y.N. and M.S.; validation, N.A., Y.N. and M.S.; formal analysis, N.A. and N.N.; investigation, N.A., Y.N., M.S., A.Z. and R.K.; resources, A.M. and A.Z.; data curation, Y.N., M.S. and N.N.; writing—original draft preparation, N.A.; writing—review and editing, A.M., R.B., A.Z. and R.K.; visualisation, Y.N. and M.S.; supervision, A.M.; project administration, N.A. and Y.N.; funding acquisition, Y.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant No. AP23490705 (2024–2026)).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The CottonWeedDet12 dataset used in this study is publicly available at <https://weed-ai.sydney.edu.au/datasets/2c14915b-0827-4b65-9908-d2a6df0d48f3> (accessed on 3 April 2026). Additional data supporting the findings of this study are available from the corresponding authors upon reasonable request.

Acknowledgments: This work has been supported financially by the research project AP23490705 (2024–2026) of the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan and was performed at the Mechanical Engineering Department, Satbayev University, which is gratefully acknowledged by the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

Mol_vert	Mollugo verticillata
Ama_palm	Amaranthus palmeri
Ecl_sp	Eclipta
Por_oler	Portulaca oleracea
Ama_tube	Amaranthus tuberculatus
Eup_macu	Euphorbia maculata
Ipo_indi	Ipomoea indica
Ele_indi	Eleusine indica
Sid_rhom	Sida rhombifolia
Sen_obtu	Senna obtusifolia
Phy_angu	Physalis angulata
Amb_arte	Ambrosia artemisiifolia

References

1. Monteiro, A.; Santos, S. Sustainable approach to weed management: The role of precision weed management. *Agronomy* **2022**, *12*, 118. [\[CrossRef\]](#)
2. Oerke, E.C. Crop losses to pests. *J. Agric. Sci.* **2006**, *144*, 31–43. [\[CrossRef\]](#)
3. Garg, S.; Rumjit, N.P.; Roy, S. Smart agriculture and nanotechnology: Technology, challenges, and new perspective. *Adv. Agrochem* **2024**, *3*, 115–125. [\[CrossRef\]](#)
4. Thorat, D.S.; Sahoo, P.K.; De, D.; Iquebal, M.A. Design and development of ridge profile power weeder. *J. Agric. Eng.* **2014**, *51*, 7. [\[CrossRef\]](#)
5. Olaoye, J.O.; Adekanye, T.A. Development and evaluation of a rotary power weeder. In Proceedings of the Tillage for Agricultural Productivity and Environmental Sustainability Conference, Ilorin, Nigeria, 31 January 2013; pp. 129–141.
6. Xuan, T.D.; Khanh, T.D.; Minh, T.T.N. Implementation of conventional and smart weed management strategies in sustainable agricultural production. *Weed Biol. Manag.* **2025**, *25*, e70000. [\[CrossRef\]](#)
7. Baćmaga, M.; Wyszowska, J.; Kucharski, J. Environmental implication of herbicide use. *Molecules* **2024**, *29*, 5965. [\[CrossRef\]](#)

8. Sustainable Development Goals. United Nations in Kazakhstan. Available online: <https://kazakhstan.un.org/ru/sdgs> (accessed on 3 February 2026).
9. Perks, A. Advanced vision guided robotics provide “future-proof” flexible automation. *Assem. Autom.* **2006**, *26*, 216–220. [CrossRef]
10. Chand, R.; Srivastava, S.K. Changes in the rural labour market and their implications for agriculture. *Econ. Polit. Wkly.* **2014**, *49*, 47–54.
11. Liakos, K.G.; Busato, P.; Moshou, D.; Pearson, S.; Bochtis, D. Machine learning in agriculture: A review. *Sensors* **2018**, *18*, 2674. [CrossRef]
12. Slaughter, D.C.; Giles, D.K.; Downey, D. Autonomous robotic weed control systems: A review. *Comput. Electron. Agric.* **2008**, *61*, 63–78. [CrossRef]
13. Shaikh, F.K.; Karim, S.; Zeadally, S.; Nebhen, J. Recent trends in Internet-of-Things-enabled sensor technologies for smart agriculture. *IEEE Internet Things J.* **2022**, *9*, 23583–23598. [CrossRef]
14. Amit, S.; Ashutosh, S.; Alexander, B.; Tanupriya, C.; Abdu, A.M.; Manuel, S.C. Artificial intelligence and Internet of Things oriented sustainable precision farming: Towards modern agriculture. *Open Life Sci.* **2023**, *18*, 313–326. [CrossRef]
15. Furnitto, N.; Todde, G.; Spagnuolo, M.; Sottosanti, G.; Caria, M.; Schillaci, G.; Failla, S.I.G. Innovations in Robots for Weed and Pest Control: A Systematic Review of Cutting-Edge Research. *Mach. Learn. Knowl. Extr.* **2026**, *8*, 51. [CrossRef]
16. Farming, GT. Available online: <https://www.agricultural-robotics.com/robot/farming-gt> (accessed on 15 January 2026).
17. Ecorobotix. Available online: <https://www.verve.vc/interview-aurelien-demaurex-ecorobotix/> (accessed on 15 January 2026).
18. Stara Imperador Sprayers. Available online: <https://www.stara.com.br/en/products-services/agricultural-machines/sprayers/imperador-3000-and-4000> (accessed on 15 January 2026).
19. Carbon Robotics Introduces G2 Product Line: Will Laser Weeding Eliminate Herbicides in Row Crops? Available online: <https://www.globalagtechinitiative.com/in-field-technologies/robotics-automation/carbon-robotics-introduces-g2-product-line-will-laser-weeding-eliminate-herbicides-in-row-crops/> (accessed on 15 January 2026).
20. Assunção, E.; Gaspar, P.D.; Mesquita, R.; Simões, M.P.; Alibabaei, K.; Veiros, A.; Proença, H. Real-time weed control application using a Jetson Nano edge device and a spray mechanism. *Remote Sens.* **2022**, *14*, 4217. [CrossRef]
21. Mwitwa, C.; Rains, G.C.; Prostko, E. Evaluation of inference performance of deep learning models for real-time weed detection in an embedded computer. *Sensors* **2024**, *24*, 514. [CrossRef] [PubMed]
22. Shadrin, D.; Dmitrii, S.; Illarionova, S.; Kasatov, R.; Akimenkova, M.; Rudensky, G.; Erhan, E. Weed detection on embedded systems using computer vision algorithms. *Comput. Opt.* **2025**, *49*, 103–111. [CrossRef]
23. Khan, Z.; Shen, Y.; Liu, H. Object detection in agriculture: A comprehensive review of methods, applications, challenges, and future directions. *Agriculture* **2025**, *15*, 1351. [CrossRef]
24. Allmendinger, A.; Saltık, A.O.; Peteinatos, G.G.; Stein, A.; Gerhards, R. Assessing the capability of YOLO- and transformer-based object detectors for real-time weed detection. *Precis. Agric.* **2025**, *26*, 52. [CrossRef]
25. Raza, H.; Abu Bakr, M.; Khan, S.D.; Batool, H.; Ullah, H.; Ullah, M. Benchmarking YOLO models for crop growth and weed detection in cotton fields. *AgriEngineering* **2025**, *7*, 375. [CrossRef]
26. Jia, Z.; Zhang, M.; Yuan, C.; Liu, Q.; Liu, H.; Qiu, X.; Zhao, W.; Shi, J. ADL-YOLOv8: A Field Crop Weed Detection Model Based on Improved YOLOv8. *Agronomy* **2024**, *14*, 2355. [CrossRef]
27. Wang, J.; Qi, Z.; Wang, Y.; Liu, Y. A lightweight weed detection model for cotton fields based on an improved YOLOv8n. *Sci. Rep.* **2025**, *15*, 457. [CrossRef] [PubMed]
28. Sulzbach, E.; Scheeren, I.; Veras, M.S.T.; Tosin, M.C.; Kroth, W.A.E.; Merotto, A., Jr.; Markus, C. Deep learning model optimization methods and performance evaluation of YOLOv8 for enhanced weed detection in soybeans. *Comput. Electron. Agric.* **2025**, *232*, 110117. [CrossRef]
29. Jin, X.; Zhang, J.; Wang, F.; Zhao, M.; Wang, Y.; Yang, J.; Wu, J.; Zhou, B. PHRF-RTDETR: A lightweight weed detection method for upland rice based on RT-DETR. *Front. Plant Sci.* **2025**, *16*, 1556275. [CrossRef]
30. Bilous, N.; Malko, V.; Frohme, M.; Nechyporenko, A. Comparison of CNN-Based Architectures for Detection of Different Object Classes. *AI* **2024**, *5*, 2300–2320. [CrossRef]
31. Lakshmanan, V.; Robinson, S.; Munn, M. *Machine Learning Design Patterns: Solutions to Common Challenges in Data Preparation, Model Building, and MLOps*; O'Reilly Media: Sebastopol, CA, USA, 2020.
32. Gu, S.; Meng, W.; Sun, G. Streamlining YOLOv7 for rapid and accurate detection of rapeseed varieties on embedded device. *Sensors* **2024**, *24*, 5585. [CrossRef] [PubMed]
33. Chiang, C.-E.; Liang, W.-Z.; Chen, J.; Qiao, X.; Tsoleva, V.; Yang, Z.; Oboamah, J. BudCAM: An edge computing camera system for bud detection in muscadine grapevines. *Agriculture* **2025**, *15*, 2220. [CrossRef]

34. García-Navarrete, O.L.; Camacho-Tamayo, J.H.; Bregon, A.B.; Martín-García, J.; Navas-Gracia, L.M. Performance Analysis of Real-Time Detection Transformer and You Only Look Once Models for Weed Detection in Maize Cultivation. *Agronomy* **2025**, *15*, 796. [[CrossRef](#)]
35. Lu, Z.; Zhu, C.; Liu, L.; Yang, Y.; Wang, J.; Xia, W.; Xu, K.; Tie, J. Star-YOLO: A lightweight and efficient model for weed detection in cotton fields using advanced YOLOv8 improvements. *Comput. Electron. Agric.* **2025**, *235*, 110306. [[CrossRef](#)]

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.