applied sciences

MDPI

Article

Embedding a Real-Time Strawberry Detection Model into a Pesticide-Spraying Mobile Robot for Greenhouse Operation

Khalid El Amraoui [1], Mohamed El Ansari [2], Mouataz Lghoul [1], Mustapha El Alaoui [1], Abdelkrim Abanay [3], Bouazza Jabri [1], Lhoussaine Masmoudi [1] and José Valente de Oliveira [4,5,*]

1 LCS Laboratory, Physics Department, Faculty of Sciences, Mohammed 5 University in Rabat, Ibn Battouta Street, Rabat 10000, Morocco; khalid.elamraoui@um5r.ac.ma (K.E.A.); mouataz_lghoul@um5.ac.ma (M.L.); m.elalaoui@um5r.ac.ma (M.E.A.); b.jabri@um5r.ac.ma (B.J.); l.masmoudi@um5r.ac.ma (L.M.)
2 Informatics and Applications Laboratory, Faculty of Science, Moulay Ismail University in Meknes, Zitoune Street, Meknes 11201, Morocco; melansari@gmail.com
3 IRSM, High Institute of Management, Administration, and Computer Engineering (ISMAGI), Rabat 10120, Morocco
4 Faculty of Science and Technology, Campus de Gambelas, Universidade do Algarve, 8005-391 Faro, Portugal
5 NOVA-LINCS, and Center of Intelligent Systems, IDMEC/LAETA, University of Lisbon, 1049-001 Lisboa, Portugal
* Correspondence: jvo@ualg.pt

Abstract: The real-time detection of fruits and plants is a crucial aspect of digital agriculture, enhancing farming efficiency and productivity. This study addresses the challenge of embedding a real-time strawberry detection system in a small mobile robot operating within a greenhouse environment. The embedded system is based on the YOLO architecture running in a single GPU card, with the Open Neural Network Exchange (ONNX) representation being employed to accelerate the detection process. The experiments conducted in this study demonstrate that the proposed model achieves a mean average precision (mAP) of over 97%, processing eight frames per second for 512 × 512 pixel images. These results affirm the utility of the proposed approach in detecting strawberry plants in order to optimize the spraying process and avoid inflicting any harm on the plants. The goal of this research is to highlight the potential of integrating advanced detection algorithms into small-scale robotics, providing a viable solution for enhancing precision agriculture practices.

Keywords: agricultural robotics; real-time detection; YOLO; embedded system; greenhouse

Citation: El Amraoui, K.; El Ansari, M.; Lghoul, M.; El Alaoui, M.; Abanay, A.; Jabri, B.; Masmoudi, L.; Valente de Oliveira, J. Embedding a Real-Time Strawberry Detection Model into a Pesticide-Spraying Mobile Robot for Greenhouse Operation. *Appl. Sci.* **2024**, *14*, 7195. https://doi.org/10.3390/app14167195

Academic Editor: Alessandro Gasparetto

Received: 11 July 2024
Revised: 3 August 2024
Accepted: 5 August 2024
Published: 15 August 2024

1. Introduction

Agriculture continues to play a crucial role in economies relying on agricultural activities [1], particularly in countries like Morocco, where it constitutes approximately 10.33% of the Gross Domestic Product (GDP) as of 2022 [2]. This sector is a significant source of employment in Morocco, contributing to nearly 33.3% of jobs and accounting for over 23% of exports [3]. Fresh strawberries are securing their position among the top 10 most exported fruits and vegetables from this country: the revenue earned through strawberry production consistently ranges from USD 40 to USD 70 million almost every year and is still increasing by an average of 3% annually [4]. However, the production of such soft fruits faces a great challenge because of pest infestation and diseases: the costs of managing these can contribute to nearly half of the production costs [5].

Recently, innovative techniques have been developed and used to improve agricultural practices, giving birth to the term "precision agriculture" [6]. The introduction of new tools and technologies, such as sophisticated irrigation systems or modern agricultural machinery, has significantly increased productivity and the ability to feed a growing population. However, plant detection in large areas remains a challenging task for farmers, particularly

when dealing with small plants such as strawberry plants. This issue has prompted numerous studies aimed at finding solutions, including research on plant segmentation [7], fruit counting [8], and other related approaches [9].

In red fruit industries, such as the tomato industry [10,11], the integration of robotics with machine learning algorithms enables a more precise, efficient, and sustainable approach, assisting farmers in making informed decisions to enhance yields, optimize the use of agricultural inputs, and reduce their environmental impact [12]. These human-like, perceptive, high-capability machines have shown great potential, especially when dealing with detection tasks. Malik et al. [10] introduced an advanced yet easy-to-embed algorithm for detecting ripe tomatoes, utilizing enhancements in the HSV (Hue, Saturation, Value) color space and watershed segmentation. This approach was able to reach an accuracy detection of up to 81.6% and was designed to be used for all red fruits. However, these methods, despite exhibiting acceptable performance for certain data acquisition conditions, do not generalize effectively. Hence, recent studies have concentrated on exploring universal feature extraction methods to address the constraints faced by traditional image detection algorithms.

To the best of our knowledge, Sa et al. [13] pioneered the application of deep learning networks for fruit detection. These techniques are known for their capability to learn complex features autonomously and can be used in the development of an end-to-end nonlinear model with high precision for detection and other applications. An example of such a model was designed by Lamb et al. [14], who used a convolutional neural network for strawberry fruit detection. Their network was based on three sets of convolutional layers (classification, detection, and prediction) capable of being embedded in Raspberry Pi 3B. The detection speed achieved by this model was 1.63 frames per second, with an average precision (AP) of 84.2%. Another example was introduced by Yu et al. [15], who employed the widely recognized Mask R-CNN framework [16] with a ResNet 50 backbone [17] combined with a feature pyramid network (FPN) [18] for feature extraction to attain an AP of 95.78% for strawberry fruits in a non-structured environment. This method covers overlapping and hidden fruits under varying illumination intensities. Moreover, the processing time for images of 640 $\times$ 480 pixels was just 0.13 s when using a high-performance computer. However, these multi-stage detection methods demand substantial resources for region proposal selection, thereby posing limitations on the detection speed. Consequently, they are not suitable for real-time field detection applications.

Delving further into the realm of real-time detection, Redmon et al. introduced the YOLO (You Only Look Once) architecture [19]. Unlike traditional methods that predict a proposal and subsequently refine it, YOLO directly predicts the final detection by dividing the image into grids and performing a prediction for each grid cell. This makes the YOLO network the most representative network with real-time capabilities [20]. Currently, there are 10 versions that have improved upon the performance of the original YOLO. YOLOv3 was the first YOLO model to be widely adopted in real-time applications. It demonstrates remarkable results in detecting the smallest objects, thanks to its deep architecture, which incorporates three size prediction layers. Prico et al. [21] proposed a real-time weed detection system for green onion crops using YOLOv3. The system was able to reach an AP of 93.81% and an F1 score of 0.94 on a five-minute UAV video with a resolution of 864 $\times$ 688 using a high-performance computer. In another study, Liu et al. [22] used a circular bounding box with the YOLOv3 model. This resulted in a mean AP of 0.96 when using an image of 3648 $\times$ 2056 pixels at a speed detection rate of 0.054 s.

Regarding strawberry-related applications, He et al. [23] applied the YOLOv4 network for strawberry maturity detection and localization for robotic harvesting in RGB images. Their approach employed a two-stage deep learning methodology utilizing YOLOv4 and YOLOv4-tiny models to identify mature strawberries and estimate their centers from both RGB and depth images. They reached an AP of 91.73% with a processing speed of 55.19 ms per image for maturity detection and a mean AP of 86.45% with a 4.16 ms processing time per image for strawberry localization. Xie et al. [24] proposed a method for the rapid and

accurate identification of strawberry fruit in greenhouses using an improved YOLOv5s approach. Their method, called YOLO-SR (YOLO–Strawberry Ripeness), replaced the C3 module with a lightweight multi-path aggregation network based on Channel Shuffle, enhancing the feature extraction capability. This method achieved a mAP of 94.3% and an F1 score of 93.7%. However, running these algorithms in a practical setting using small robots is very difficult because the YOLO network needs a strong GPU (Graphics Processing Unit) with over 4 Gigabytes of memory, which most electronic boards cannot handle [20].

As a solution, researchers have developed a lightweight version of the YOLO network, introducing the term YOLO-tiny. In their study, Zhang et al. [25] designed and implemented a lightweight deep neural network (RTSD-Net) for real-time strawberry detection on embedded devices, such as Jetson Nano. By reducing the number of convolutional layers and the cross-stage partial network of a YOLOv4-tiny architecture, and by using the TensorRT method to speed up the model, they achieved a mean AP of 82.44% on an aerial strawberry dataset, and the processing speed was 25.2 frames per second (FPS). These results were achieved using a dataset comprising only three types of objects, with a uniform image background, which may not be representative of all types of scenarios. Additionally, the network architecture of the YOLO-tiny model has fewer layers and parameters, allowing for quicker performance but increasing the risk of information loss in complex environments [26]. To the authors' best knowledge, there have been no previous studies on the use of YOLO models for strawberry plant detection in complex environments for pesticide spraying application. Consequently, this paper explores the possibility of using the YOLO model for detecting strawberry plants in such types of applications.

The pesticide-spraying process is a complex task that requires instant decision-making by the robot during a survey [27], in contrast to applications such as yield prediction [28] or disease detection [29], which can be performed separately using a high-performance computer. For that, limited choices on the suitable detection model are available. This study highlighted the strengths and weaknesses of YOLO versions that are already being implemented in robotic platforms for instant decision-making, offering a data-driven basis for selecting the most suitable model for pesticide spraying robots.

For embedded systems with limited computational resources, YOLOv3 is the most suitable YOLO version in terms of memory and computational requirements compared with the later versions. Some studies, such as [11,20,22,30], used embedded YOLO and affirmed a good balance between the speed and accuracy of YOLOv3 for real-time applications, even if newer versions existed. Based on that, this study aims to develop a real-time plant detection algorithm that can be implemented in an agricultural pesticide-spraying robot using an NVIDIA Jetson TX2 GPU. The developed method, using YOLOv3, was specifically applied to strawberry plants within a greenhouse environment. To assess the effectiveness of our implementation, we used the Open Neural Network Exchange representation (ONNX). In order to show the performance of our proposal, a set of real-world experiments was carried out in a didactical strawberry greenhouse, which revealed that the proposed model can reach a mAP of over 97% with a processing time of 15 ms per frame.

This paper is organized as follows: Section 2 presents a brief discussion of the robot and YOLOv3 model that is used in our application. The main results are detailed in Section 3, and a section on the main conclusions ends this paper.

2. Materials and Methods

2.1. AgriEco Robot

The mobile robot "AgriEco Robot" was developed in our laboratory to be used in a didactical strawberry greenhouse [31]. Its primary objective is to control the pesticide-spraying process, encompassing aspects such as quantity regulation and ensuring precise localized spraying. It consists of a robotic chassis 550 × 650 mm in size made of steel, equipped with four-wheel drive. The spraying system is composed of a 10 L bank of polythene, a submerged pump, and two motorized arms. A navigation system, using

Hokuyo URG-04LX-UG01 2D Lidar, from HOKUYO AUTOMATIC CO LTD in Osaka, Japan [32], is installed in the robot's front section. The robot has a vision system using 2 types of cameras, a ZED camera stereo and omnidirectional camera. See Figure 1.

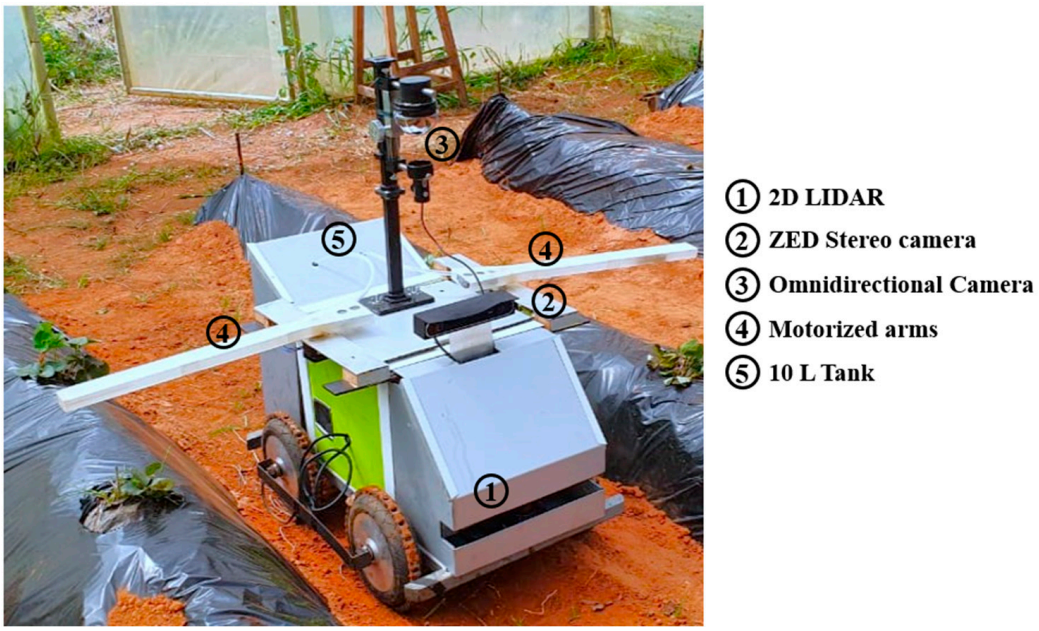

Figure 1. AgriEco Robot platform.

The robot utilizes the open-source framework Robot Operating System (ROS) [33], which combines CPU and GPU electronic components to execute its diverse range of tasks. For simple tasks like controlling the arms, the pumps, and the wheels, the robot uses Raspberry Pi 4. For deep learning processing tasks, it uses a built-in NVIDIA Jetson TX2 cart from NVIDIA company in the USA [34], equipped with a Quad-core 2.0 Ghz 64-bit ARMv8 A57, a dual-core 2.0 Ghz ARMv8 Denver, a 256 CUDA core 1.3 MHz Nvidia Pascal, and 8 GB memory. To power the robot, a 36 V/30 Ah lithium-ion battery is used. These configurations make it well-suited as an embedded system for this study application.

2.2. Deep Learning Model

While YOLO models themselves are well-established, this work innovatively integrates these models into an already existing pesticide-spraying robot designed for the complex greenhouse environment. This integration is non-trivial and requires addressing unique challenges related to real-time object detection, decision-making, and precise actuation in dynamic greenhouse settings. These challenges include the robot's autonomous navigation system, the plants' orchard environment, and pesticide actuation control. Additionally, several other challenges must be considered because of their direct impact on the proposed application, such as the lack of a sufficient training dataset, the choice of the loss function to improve performance, and the implementation using a single GPU NVIDIA Jetson TX2 board. These constraints limited our choices to versions that are stable, easily embedded, and exhibit excellent performance in instant decision-making.

Nowadays, there are 10 versions of YOLO, and the newest versions show a fast yet accurate performance, making them highly suitable for real-time applications [35]. However, earlier versions of YOLO still offer distinct advantages, such as simplicity and resource efficiency, that make them better candidates in applications with limited computational resources such as edge devices [36]. To select the most suitable YOLO version for this study, a comparative analysis was performed by comparing the most-used version in the embedded system (3, 4-tiny, 5-small, 5-medium) using NVIDIA Jetson TX2. This comparison took into

consideration the model's performance ins term of the number of parameters, processing speed, and mAP on the COCO dataset [37]. Table 1 summarizes the overall performance.

Table 1. Comparison between most used YOLO version in embedded system with NVIDIA Jetson TX2.

YOLO Version	Parameters	Processing Speed (FPS)	mAP (COCO)
YOLOv3 [38]	61.5 million	~10–15 FPS	57.9%
YOLOv4-tiny [39]	6 million	~25–30 FPS	36.9%
YOLOv5-Small [40]	7.2 million	~20–25 FPS	39.4%
YOLOv5-medium [40]	21.2 million	~15–20 FPS	45.4%

The values in Table 1 suggest that YOLOv5-medium provides the best balance of performance between the processing speed and mAP. In contrast, YOLOv3 demonstrates the best mAP compared with the other versions, but it has the lowest frames per second (FPS), which is expected given its high number of parameters. Based on this analysis, YOLOv5-medium can be considered optimal for applications where moderate accuracy and moderate speed are acceptable, while YOLOv3 is more suitable for applications that require high accuracy with an accepted slower processing speed, which is the case in this study.

Pesticide spraying is a stationary and critical process, where the robot navigates slowly between the strawberry rows to maintain stability and ensure a straight navigation line. Also, localized spraying is crucial to avoid harming the plants. This requires the robot to come to a full stop before adjusting the spraying nozzle to avoid the maximum surface of the plant. Based on that, YOLOv3 was chosen in this study since the spraying process is critical and demands high precision, which requires the highest mAP, unlike the processing speed, which is less critical because of the robot's slow movement.

The YOLOv3 [38] architecture, an automatic, robust, real-time, deep learning model, was chosen to perform real-time detection of the strawberry crop in a greenhouse. YOLO stands for (You Only Look Once), and it is a family of models used for real-time object detection. As suggested by its name, YOLO refers to a single network that localizes and identifies objects in a single pass, making it rather efficient. It has been applied in many agricultural applications, e.g., [41–43]. The third version YOLOv3, which is recommended for small objects [38], combines 2 key contributions from 2 different architectures as follows: the skip connections from the ResNet architecture [17] and 3 prediction stages from Feature Pyramid Networks (FPNs) [18], resulting in the architecture shown in Figure 2.

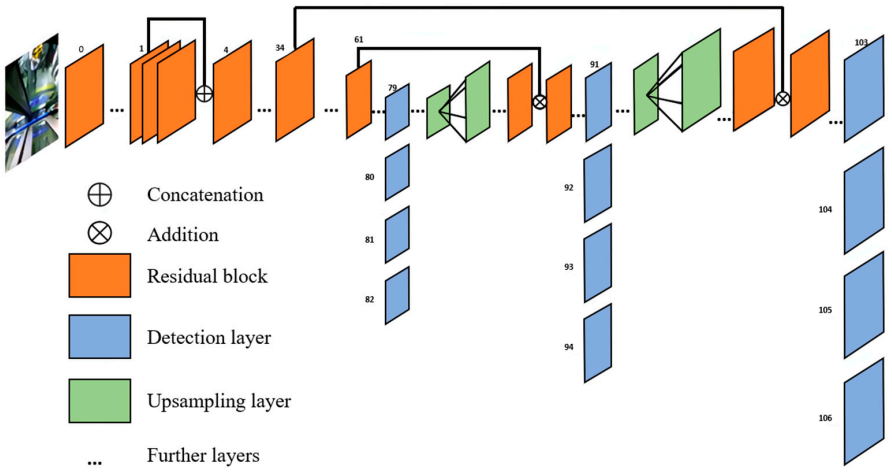

Figure 2. YOLOv3 network architecture [44].

Compared with vanilla feedforward neural networks, the ResNet architecture uses skip connections [45] between the layers. Consequently, in the backpropagation, the weight updates are based on eliminating $F(x)$ in order to obtain $y = x$. This makes the ResNet

architecture state-of-the-art in many deep learning models for segmentation tasks [46], image denoising [47], and object detection.

The YOLOv3 algorithm uses a multi-scale feature extraction network named DarkNet53, which originally had 53 layers pretrained on the ImageNet dataset. For the detection task, an additional 53 layers are stacked on top of it, giving a fully convolutional underlying architecture of 106 layers for YOLOv3 (Figure 3). Also, it consists of three prediction stages, each followed by an up-sampling layer to increase the size of the feature maps. This enables the model to detect objects of various sizes effectively [48]. Initially, predictions are made on smaller feature maps to detect large-scale objects. Subsequently, predictions are made on medium-sized feature maps to identify medium-sized objects, and finally, predictions on the largest feature maps focus on detecting small objects. By combining these predictions into a single output, the model generates an image with bounding boxes delineating different objects.

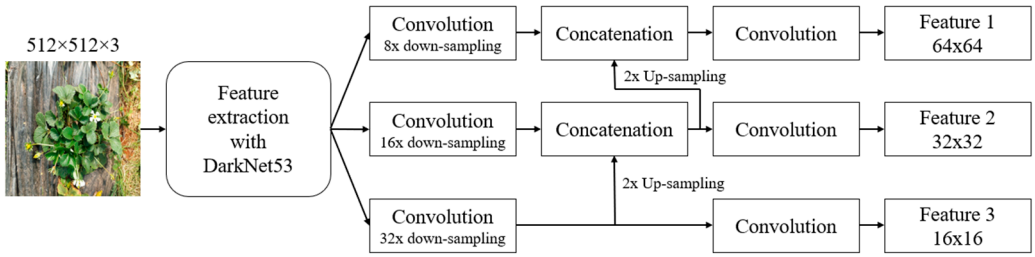

Figure 3. DarkNet53 Multi-scale feature extraction pipeline.

One other characteristic of the YOLOv3 model is the employment of intersection over union (IOU) [49]. This metric is frequently used in object detection benchmarks. YOLOv3 uses this metric by determining the error between the predicted bounding box and the ground truth. These bounding boxes are determined by five predictions. The center of the bounding box coordinates is labeled x and y, the dimensions of the bounding box are labeled h and w, and finally, the class probability is determined. The probability value is equal to 1 if a grid cell has a portion of the ground truth bounding box, and 0 if not. Figure 4 illustrates the detection process of the YOLO models. The first step involves taking the image grids (Figure 4a) and predicting the class probabilities map, distinguishing the object from the background (Figure 4b). Subsequently, the model predicts the bounding boxes (Figure 4c), which collectively form the predicted bounding box surrounding the object. In this instance, the object refers to the plants (Figure 4d).

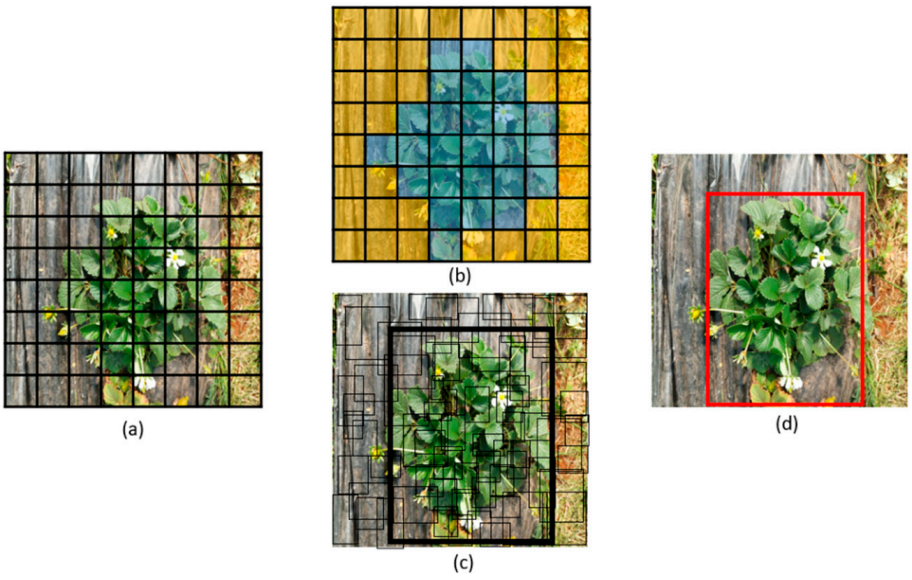

Figure 4. Detection steps of the YOLO model. (**a**) Input image with an $N \times N$ grid, (**b**) class probability prediction map, (**c**) bonding box prediction for each grid, and (**d**) the detection result.

3. Proposed Method

The main idea is to train a model based on the YOLOv3 architecture for strawberry real-time detection. The model is then implemented in a robotic platform via the NVIDIA Jetson TX2 card using ONNX representation to accelerate detection.

3.1. Data Acquisition

The dataset used to train the proposed model was locally captured from a strawberry greenhouse using a mobile phone camera with a resolution of 3472 × 3472 pixels, capturing various plant sizes in different angles and lighting conditions. The greenhouse is considered small with an area of 36 m^2 containing four rows with five plants each Figure 5. The images were captured from an overhead perspective to simulate the robot's visual system. Furthermore, the following distinct times of day were considered: morning, noon, and afternoon.

Figure 5. The experimental greenhouse used in this work.

Because of the limited availability of the environment, a total of 150 images were captured, focusing on the intricate orchard environment of strawberry plants within the greenhouse, as shown in Figure 6. Each image presents a complex backdrop, featuring various objects that pose challenges to training a model effectively. The diverse assortment of objects within the background adds layers of complexity to the task. Moreover, the small size of the dataset used makes the training process for object detection more complex and harder to achieve. To address this issue, a data augmentation step was employed to improve the training data by applying a series of transformations such as rotation, translation, and color enhancement [50].

Figure 6. Strawberry plant images under different capture conditions (illumination, angles) in the greenhouse environment.

3.2. Data Augmentation

To train a YOLO model for object detection, a large dataset is required as it improves the model's precision [51]. In this study case, only 150 images were taken because of the small environment in which the experiment was performed. Furthermore, to achieve accurate detection, this study accounted for various types of interference that could arise during plant detection, as shown in Figure 6; however, this complicated background can cause an overfitting to the model [50]. To address this issue, the data augmentation method was used to expand the training dataset. Methods such as adjusting color brightness, rotating images, and vertical–horizontal translations have the potential to enhance the performance of detection [52,53]. In this study, by using a Python script, transformations such as image rotation, color adjustment, and spatial translations were considered to expand the data.

The dataset was augmented to reach a total of 556 images, where 400 images were considered as the training dataset and 156 images as the validation dataset. The annotations used to train and test the network were generated through manual labeling using [54], which involved marking the plants with rectangular bounding boxes in all the images of the training and testing datasets.

3.3. Data Pre-Processing

The image size was modified from the original size to 512 × 512 pixels, which represents the maximum feasible image size to train the model using the computer hardware utilized in this study. To create annotations, we use MakeSense [54], a free-to-use online tool, to label the areas of each plant by drawing a bounding box around it and then exporting it as an xml file in PASCAL-VOC format. A total of 556 labels were exported, including information about the image shape along with the object type followed by the plant bounding box coordinates.

3.4. Running Environment

The main hardware platform used to train this model was a desktop computer equipped with an Intel i7 8th Gen processor, 16 GB of DDR4 2666 MHz memory, and Nvidia GeForce GTX 1070 with 8 GB VRAM. We used Python 3.8.8, TensorFlow 2.3, the CUDA 10.1 parallel computing framework with the CUDNN 7.6 deep neural network acceleration library, and OpenCV computer vision 4.0.0., as detailed in Table 2.

Table 2. Machine used for training.

CPU	Intel® i7 8th Gen
GPU	NVidia GTX 1070
Frequency	3.2 GHz
RAM	16 GB
Storage	500 Gb SSD
Operating system	Windows 10
Programming platform	Python 3.8

3.5. Model Training

As discussed in the previous section, our model is based on the YOLOv3 architecture. A total of over 556 images were randomly distributed to the training dataset, which received 400 images, and 156 images were distributed to the validation dataset. A batch size of four and fixed momentum value of 0.9, as well as a learning rate of 0.0001, were considered. The training process was repeated 10 times independently, where the trainable weights were initialized. To track the best model for each execution, an early stopping method was applied when the loss on the validation set was not improved for five consecutive epochs. The validation loss results over the 10-time training process is summarized in Table 3. These results indicate that, on average, it took 10 to 11 epochs for the model to reach stability. Each process took over 12 h to complete 800 iterations across the 10 epochs, totaling 8000 iterations (Table 4).

Table 3. The validation loss results over 10 independent executions.

Training process	1	2	3	4	5	6	7	8	9	10
Best validation loss	8.75	8.43	8.69	8.40	8.33	8.00	7.75	8.25	8.10	7.94
Epoch	13	17	10	12	8	13	7	15	11	9

Table 4. Model training hyperparameters.

Iterations	8000
Batch size	4
Learning rate	0.0001
Epochs	10
Confidence threshold	0.5

To overcome the limitation of the training dataset and obtain more accurate outcomes, a transfer learning approach was employed to train the YOLOv3 model. It offers the benefit of enabling a network trained on a limited ground-truth dataset to achieve significant accuracy in detection [20]. Trained weights from ImageNet for YOLOv3 darknet [33] were transferred to the model.

During training, the model produced an output in every iteration and validation loss was minimized. The evaluation metrics used in this study are mean average precision (*mAP*), recall, and F1-score, defined by Equations (1), (2), and (3), respectively. The model with the highest precision among the 10 executions was chosen as the optimal model.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i = AP \tag{1}$$

where N is the number of classes and AP_i is the average precision for class i. In this study case, $N = 1$, which implies $mAP = AP$.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

$$\mathrm{F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{3}$$

The loss function used in this study is given in (4), where $loss_1$, $loss_2$, and $loss_3$ are the bounding box regression loss function, Smooth L1 loss, and classification loss, respectively.

$$loss = loss_1 + loss_2 + loss_3 \tag{4}$$

For $loss_1$, we utilize intersection over union (IoU) based on the provided annotations. These annotations include $\left(x_1^g, y_1^g\right)$ and $\left(x_2^g, y_2^g\right)$ for representing the top-left and bottom-right coordinates of the ground truth bounding box, respectively. In the prediction, we have x_1^p, y_1^p and $\left(x_2^p, y_2^p\right)$, and Equation (5) is used to calculate the IOU as follows:

$$loss_1 = IoU = \frac{Area_{intersection}}{Area_{union}} \tag{5}$$

where

$$Area_{intersection} = max\left(0, min\left(x_2^p, x_2^g\right) - max\left(x_1^p, x_1^g\right)\right) \times max\left(0, min\left(y_2^p, y_2^g\right) - max\left(y_1^p, y_1^g\right)\right),$$

and

$$Area_{union} = Area_{predicted} + Area_{ground-truth} - Area_{intersection}$$

Smooth L1 loss is given in (6), where x represents the difference between the predicted and the target bounding box coordinates. The reason behind the employment of this loss function pertains to [55], where it is affirmed that Smooth L1 loss penalizes large errors more gently compared with traditional MSE loss, which makes it suitable for object detection tasks.

$$loss_2 = \mathrm{SmoothL1}(x) = \begin{cases} 0.5 * x^2 & if\ |x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \tag{6}$$

This study represents a binary classification problem, so the binary cross-entropy loss function shown in (7) is used.

$$loss_3 = L_{binary\_crossentropy} = -\left(y_{true} \log\left(y_{pred}\right) + (1 - y_{true}) \log\left(1 - y_{pred}\right)\right) \tag{7}$$

where

y_{true} is the true class label (0 or 1).

y_{pred} is the predicted probability of the positive class (class 1).

4. Results and Discussion

The performance of the best model extracted from the 10 different executions on the validation set are listed in Table 5, indicating a precision of 0.73 and a recall of 0.95, leading to an F1-score of 0.83 for the YOLOv3 model. YOLOv3-tiny showed minor results, with a precision of 0.60 and a recall of 0.71, leading to an F1-score of 0.65. These results are expected because of the small training dataset along with the reduced number of layers in the tiny version. An F1-score of 0.83 indicates that YOLOv3 has a good balance between the precision and recall for this dataset, showing its efficiency in identifying true positives while minimizing false positives.

Table 5. YOLOv3 and YOLOv3-tiny validation metric summary on the strawberry dataset.

Model	Precision	Recall	F1 Score
YOLOv3	0.73	0.95	0.83
YOLOv3-tiny	0.60	0.71	0.65

Based on that, real-world experiments were conducted using the strawberry plants in the greenhouse to verify the model's generalization performance. First, we performed an in-lab test using the tools shown in Figure 7, including the NVIDIA Jetson TX2 card, a web camera with a resolution of 640 × 480, a monitor for visualization, a keyboard, and a mouse. To assess the effectiveness of

our implementation, the ONNX representation was used. To assess the accuracy, suitability, and consistency of the proposed model, three alternative contemporary scenarios were trained and tested on the same datasets. These scenarios included traditional YOLOv3, YOLOv3-tiny, and YOLOv3-tiny with ONNX. The experiment was performed using two types of data, i.e., video and live video. We encountered certain challenges with the robot, as it is still in the development stage. So, we conducted a simulation utilizing the tools illustrated in Figure 7, positioning the camera at an optimal altitude to cover two rows of the plantation. Table 6 summarizes numerical evidence on the performance of the model compared to the other scenarios. The model achieves an accuracy of more than 97% with a processing time of over 15 milliseconds per frame.

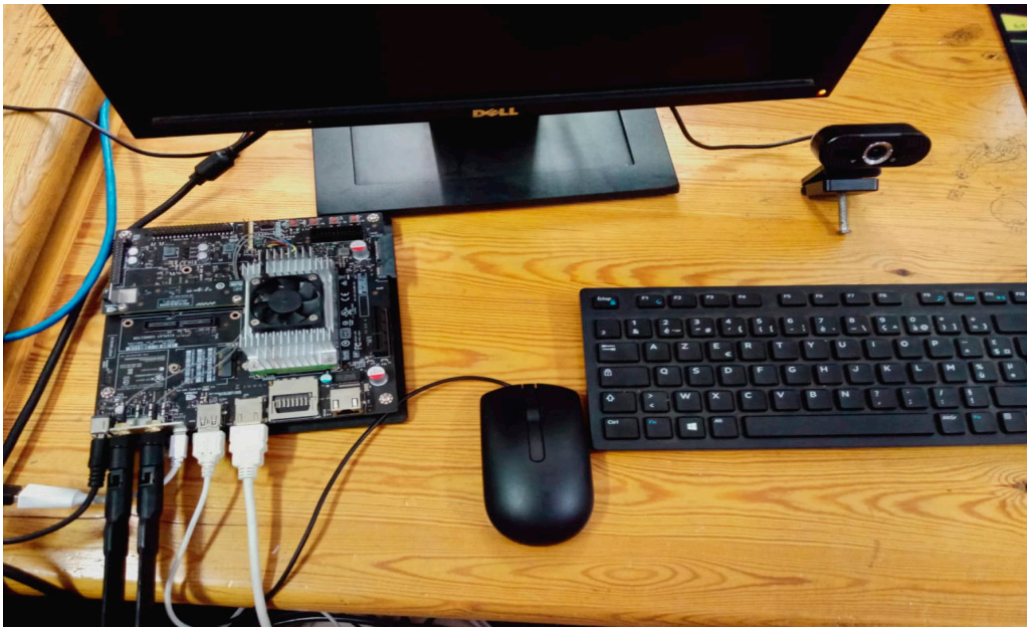

Figure 7. In-lab tools used to perform real-time strawberry plant detection.

Table 6. Model speed and accuracy with and without ONNX representation.

		Model							
		YOLOv3		**YOLOv3 with ONNX**		**YOLOv3-Tiny**		**YOLOv3-Tiny with ONNX**	
Data	**Resolution**	**mAP**	**FPS**	**mAP**	**FPS**	**mAP**	**FPS**	**mAP**	**FPS**
Video	1920 × 1080	93.5%	3	96.3%	6	89.5%	5	90.7%	8
Live video	640 × 480	90.4%	4	94%	8	86.1%	7	88.3%	10

The results shown in Table 6 demonstrate real-time plant detection with mAP above 96%, which reveals the effectiveness and utility of the proposed model. Moreover, it showcases superior performance compared with the other three scenarios for both types of data. However, when considering processing speed, YOLOv3-tiny exhibits potential for achieving faster speeds because of its optimal balance between detection speed and accuracy, particularly when utilizing the ONNX representation. Nevertheless, in applications such as robotic pesticide spraying, detection accuracy holds greater significance, as the robot must ensure precise spraying while avoiding substantial areas of the plants. Additionally, YOLOv3-tiny versions are characterized by their compact networks and fewer prediction stages, which may risk information loss on objects of various sizes, given the varying sizes of plants. This issue is particularly significant in our case, where we have a limited dataset to train the model. Furthermore, when assessing robot navigation speed, a difference of 2 FPS may not be as critical as an approximate 6% increase in accuracy. It is important to acknowledge that these results were obtained by running the models using the NVIDIA Jetson TX2 board and not a computer. Figure 8 shows some online real-time detection in the greenhouse environment covering different day illumination conditions in the morning and afternoon.

(**a**) (**b**)

Figure 8. Strawberry plant real-time detection using the trained model. (**a**) Afternoon test and (**b**) morning test.

It is also noticeable that the mAP is affected by the type of data used, possibly because of variations in acquisition resolution. As illustrated in Table 6, testing the model on higher-resolution videos yields superior results across all scenarios. This improvement could also be attributed to the stability of the acquisition process; for instance, the tested video exhibits less vibration compared with live video feeds. Furthermore, the quality of the training data may contribute to these outcomes. However, greater flexibility and diversity in factors such as angle, lighting conditions, focal length, sensitivity, and exposure duration during image capture result in a larger volume of collected data. Moreover, employing a variety of data augmentation techniques enhances the effectiveness of model training, leading to higher detection accuracy. In summary, the results of this study indicate that YOLOv3 with ONNX representation notably enhances detection accuracy. Additionally, it significantly boosts the image processing speed, which makes it suitable for implementation in electronic boards, such as the one used in this study.

5. Conclusions

This paper presented a deep learning model for strawberry plant real-time detection embedded in a pesticide-spraying robotic platform for greenhouse applications. Comparative studies on YOLO versions for embedded systems were conducted, leading to the selection of YOLOv3 as the suitable model for this application. To train the model, a dataset was locally captured covering different illumination conditions in the greenhouse environment. To perform the detection and ensure a better and more effective implementation, the ONNX representation was needed. The model shows high accuracies, above 97%, which affirm its utility in plant detection tasks. However, it is important to acknowledge that the model has certain limitations, as the accuracies may not consistently exceed 97%. We attribute this to potential issues related to the lack of training data and the quality of annotations. As an extension of this study, a tracking system based on a machine learning model will be added so the robot can track the plants accurately, thereby facilitating tasks such as plant counting and other applications.

Author Contributions: K.E.A.: writing—original draft, data curation, conceptualization, methodology, software, formal analysis, and writing—review and editing. M.E.A. (Mohamed El Ansari): validation, formal analysis, and project administration. M.L.: data curation and resources. M.E.A. (Mustapha El Alaoui): validation and data curation. A.A.: data curation and resources. B.J.: validation and resources. L.M.: project administration, supervision, resources, and funding acquisition. J.V.d.O.: writing—review and editing, validation, resources, and funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Ministry of Higher Education, Scientific Research and Innovation of Morocco (MESRSI), the National Centre of Scientific and Technical Research of Morocco (CNRST), and the Digital Development Agency of Morocco (ADD) through the AL-KHAWARIZMI program of Morocco, project Alkhawarizmi/2020/37. This work was also supported by NOVA LINCS ref. UIDB/04516/2020 (https://doi.org/10.54499/UIDB/04516/2020) with the financial support of FCT.IP.

Data Availability Statement: Dataset available on request from the authors.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Food and Agriculture Organization of the United Nations (Ed.) *The Future of Food and Agriculture: Trends and Challenges*; Food and Agriculture Organization of the United Nations: Rome, Italy, 2017; ISBN 978-92-5-109551-5.
2. Morocco—GDP Distribution Across Economic Sectors 2012–2022. Statista. Available online: https://www.statista.com/statistics/502771/morocco-gdp-distribution-across-economic-sectors/ (accessed on 6 February 2024).
3. Morocco: Food Share in Merchandise Exports. Statista. Available online: https://www.statista.com/statistics/1218971/food-share-in-merchandise-exports-in-morocco/ (accessed on 6 February 2024).
4. Oluwole, V. Morocco's Fresh Strawberry Exports Generate up to $70 Million in Annual Revenue. Available online: https://africa.businessinsider.com/local/markets/moroccos-fresh-strawberry-exports-generate-up-to-dollar70-million-in-annual-revenue/yrkgkzv (accessed on 6 February 2024).
5. Li, Y.; Wang, H.; Dang, L.M.; Sadeghi-Niaraki, A.; Moon, H. Crop pest recognition in natural scenes using convolutional neural networks. *Comput. Electron. Agric.* **2020**, *169*, 105174. [CrossRef]
6. Pierce, F.J.; Nowak, P. Aspects of Precision Agriculture. In *Advances in Agronomy*; Sparks, D.L., Ed.; Academic Press: Cambridge, MA, USA, 1999; Volume 67, pp. 1–85. [CrossRef]
7. Hamuda, E.; Glavin, M.; Jones, E. A survey of image processing techniques for plant extraction and segmentation in the field. *Comput. Electron. Agric.* **2016**, *125*, 184–199. [CrossRef]
8. Liu, X.; Chen, S.W.; Liu, C.; Shivakumar, S.S.; Das, J.; Taylor, C.J.; Underwood, J.; Kumar, V. Monocular Camera Based Fruit Counting and Mapping with Semantic Data Association. *IEEE Robot. Autom. Lett.* **2019**, *4*, 2296–2303. [CrossRef]
9. Singh, V.; Misra, A.K. Detection of plant leaf diseases using image segmentation and soft computing techniques. *Inf. Process. Agric.* **2017**, *4*, 41–49. [CrossRef]
10. Malik, M.H.; Zhang, T.; Li, H.; Zhang, M.; Shabbir, S.; Saeed, A. Mature Tomato Fruit Detection Algorithm Based on improved HSV and Watershed Algorithm. *IFAC-PapersOnLine* **2018**, *51*, 431–436. [CrossRef]
11. Wang, X.; Vladislav, Z.; Viktor, O.; Wu, Z.; Zhao, M. Online recognition and yield estimation of tomato in plant factory based on YOLOv3. *Sci. Rep.* **2022**, *12*, 8686. [CrossRef]
12. Treboux, J.; Genoud, D. Improved Machine Learning Methodology for High Precision Agriculture. In Proceedings of the 2018 Global Internet of Things Summit (GIoTS), Bilbao, Spain, 4–7 June 2018; pp. 1–6. [CrossRef]
13. Sa, I.; Ge, Z.; Dayoub, F.; Upcroft, B.; Perez, T.; McCool, C. DeepFruits: A Fruit Detection System Using Deep Neural Networks. *Sensors* **2016**, *16*, 1222. [CrossRef]
14. Lamb, N.; Chuah, M.C. A Strawberry Detection System Using Convolutional Neural Networks. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 2515–2520. [CrossRef]
15. Yu, Y.; Zhang, K.; Yang, L.; Zhang, D. Fruit detection for strawberry harvesting robot in non-structural environment based on Mask-RCNN. *Comput. Electron. Agric.* **2019**, *163*, 104846. [CrossRef]
16. He, K.; Gkioxari, G.; Dollar, P.; Girshick, R. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
17. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; IEEE: New York, NY, USA, 2016; pp. 770–778. [CrossRef]
18. Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 936–944. [CrossRef]
19. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; IEEE: New York, NY, USA, 2016; pp. 779–788. [CrossRef]
20. Fu, L.; Feng, Y.; Wu, J.; Liu, Z.; Gao, F.; Majeed, Y.; Al-Mallahi, A.; Zhang, Q.; Li, R.; Cui, Y. Fast and accurate detection of kiwifruit in orchard using improved YOLOv3-tiny model. *Precis. Agric.* **2021**, *22*, 754–776. [CrossRef]
21. Parico, A.I.B.; Ahamed, T. An Aerial Weed Detection System for Green Onion Crops Using the You Only Look Once (YOLOv3) Deep Learning Algorithm. *Eng. Agric. Environ. Food* **2020**, *13*, 42–48. [CrossRef]

22. 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. [CrossRef] [PubMed]
23. He, Z.; Karkee, M.; Zhang, Q. Detecting and Localizing Strawberry Centers for Robotic Harvesting in Field Environment. *IFAC-PapersOnLine* **2022**, *55*, 30–35. [CrossRef]
24. Xie, Z.; Chen, R.; Lin, C.; Zeng, J. A Lightweight Real-Time Method for Strawberry Ripeness Detection Based on Improved Yolo. Available online: https://ssrn.com/abstract=4570965 (accessed on 6 February 2024).
25. Zhang, Y.; Yu, J.; Chen, Y.; Yang, W.; Zhang, W.; He, Y. Real-time strawberry detection using deep neural networks on embedded system (rtsd-net): An edge AI application. *Comput. Electron. Agric.* **2022**, *192*, 106586. [CrossRef]
26. Yang, S.; Zhang, J.; Bo, C.; Wang, M.; Chen, L. Fast vehicle logo detection in complex scenes. *Opt. Laser Technol.* **2019**, *110*, 196–201. [CrossRef]
27. Ghafar, A.S.A.; Hajjaj, S.S.H.; Gsangaya, K.R.; Sultan, M.T.H.; Mail, M.F.; Hua, L.S. Design and development of a robot for spraying fertilizers and pesticides for agriculture. *Mater. Today Proc.* **2023**, *81*, 242–248. [CrossRef]
28. Ma, B.; Hua, Z.; Wen, Y.; Deng, H.; Zhao, Y.; Pu, L.; Song, H. Using an improved lightweight YOLOv8 model for real-time detection of multi-stage apple fruit in complex orchard environments. *Artif. Intell. Agric.* **2024**, *11*, 70–82. [CrossRef]
29. Solimani, F.; Cardellicchio, A.; Dimauro, G.; Petrozza, A.; Summerer, S.; Cellini, F.; Renò, V. Optimizing tomato plant phenotyping detection: Boosting YOLOv8 architecture to tackle data complexity. *Comput. Electron. Agric.* **2024**, *218*, 108728. [CrossRef]
30. Madasamy, K.; Shanmuganathan, V.; Kandasamy, V.; Lee, M.Y.; Thangadurai, M. OSDDY: Embedded system-based object surveillance detection system with small drone using deep YOLO. *J. Image Video Proc.* **2021**, *2021*, 19. [CrossRef]
31. Abanay, A.; Masmoudi, L.; Ansari, M.E.; Gonzalez-Jimenez, J.; Moreno, F.-A.; Abanay, A.; Masmoudi, L.; Ansari, M.E.; Gonzalez-Jimenez, J.; Moreno, F.-A. LIDAR-based autonomous navigation method for an agricultural mobile robot in strawberry greenhouse: AgriEco Robot. *AIMS Electron. Electr. Eng.* **2022**, *6*, 317–328. [CrossRef]
32. Abanay, A.; Masmoudi, L.; El Ansari, M. A calibration method of 2D LIDAR-Visual sensors embedded on an agricultural robot. *Optik* **2022**, *249*, 168254. [CrossRef]
33. Min, S.K.; Delgado, R.; Byoung, W.C. Comparative Study of ROS on Embedded System for a Mobile Robot. *J. Autom. Mob. Robot. Intell. Syst.* **2018**, *12*, 61–67.
34. Amert, T.; Otterness, N.; Yang, M.; Anderson, J.H.; Smith, F.D. GPU Scheduling on the NVIDIA TX2: Hidden Details Revealed. In Proceedings of the 2017 IEEE Real-Time Systems Symposium (RTSS), Paris, France, 5–8 December 2017; pp. 104–115. [CrossRef]
35. Jiang, P.; Qi, A.; Zhong, J.; Luo, Y.; Hu, W.; Shi, Y.; Liu, T. Field cabbage detection and positioning system based on improved YOLOv8n. *Plant Methods* **2024**, *20*, 96. [CrossRef]
36. Zhou, J.; Hu, W.; Zou, A.; Zhai, S.; Liu, T.; Yang, W.; Jiang, P. Lightweight Detection Algorithm of Kiwifruit Based on Improved YOLOX-S. *Agriculture* **2022**, *12*, 7. [CrossRef]
37. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In Proceedings of the Computer Vision—ECCV 2014, Zurich, Switzerland, 6–12 September 2014; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 740–755. [CrossRef]
38. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *Comput. Vis. Pattern Recognit.* **2018**, *1804*, 1–6.
39. Beyaz, A.; Gül, V. YOLOv4 and Tiny YOLOv4 Based Forage Crop Detection with an Artificial Intelligence Board. *Braz. Arch. Biol. Technol.* **2023**, *66*, e23220803. [CrossRef]
40. Li, C.; Zhang, B.; Li, L.; Li, L.; Geng, Y.; Cheng, M.; Xiaoming, X.; Chu, X.; Wei, X. Yolov6: A single-stage object detection framework for industrial applications. In Proceedings of the Twelfth International Conference on Learning Representations (ICLR2024), Vienna, Austria, 7–11 May 2024; Available online: https://openreview.net/forum?id=7c3ZOKGQ6s (accessed on 10 July 2024).
41. Ponnusamy, V.; Coumaran, A.; Shunmugam, A.S.; Rajaram, K.; Senthilvelavan, S. Smart Glass: Real-Time Leaf Disease Detection using YOLO Transfer Learning. In Proceedings of the 2020 International Conference on Communication and Signal Processing (ICCSP), Melmaruvathur, India, 28–30 July 2020; pp. 1150–1154. [CrossRef]
42. Qin, Z.; Wang, W.; Dammer, K.-H.; Guo, L.; Cao, Z. Ag-YOLO: A Real-Time Low-Cost Detector for Precise Spraying with Case Study of Palms. *Front. Plant Sci.* **2021**, *12*, 753603. [CrossRef]
43. Buzzy, M.; Thesma, V.; Davoodi, M.; Mohammadpour Velni, J. Real-Time Plant Leaf Counting Using Deep Object Detection Networks. *Sensors* **2020**, *20*, 6896. [CrossRef]
44. Dai, Y.; Liu, W.; Li, H.; Liu, L. Efficient Foreign Object Detection Between PSDs and Metro Doors via Deep Neural Networks. *IEEE Access* **2020**, *8*, 46723–46734. [CrossRef]
45. Wu, D.; Wang, Y.; Xia, S.-T.; Bailey, J.; Ma, X. Skip Connections Matter: On the Transferability of Adversarial Examples Generated with ResNets. In Proceedings of the 2020 International Conference on Learning Representations, Addis Ababa, Ethiopia, 30 April 2020; Available online: https://openreview.net/forum?id=BJlRs34Fvr (accessed on 17 July 2023).
46. Mubashar, M.; Ali, H.; Grönlund, C.; Azmat, S. R2U++: A multiscale recurrent residual U-Net with dense skip connections for medical image segmentation. *Neural Comput. Appl.* **2022**, *34*, 17723–17739. [CrossRef]
47. Peng, Y.; Zhang, L.; Liu, S.; Wu, X.; Zhang, Y.; Wang, X. Dilated Residual Networks with Symmetric Skip Connection for image denoising. *Neurocomputing* **2019**, *345*, 67–76. [CrossRef]
48. Wang, H.; Hu, Z.; Guo, Y.; Yang, Z.; Zhou, F.; Xu, P. A Real-Time Safety Helmet Wearing Detection Approach Based on CSYOLOv3. *Appl. Sci.* **2020**, *10*, 6732. [CrossRef]

49. Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; Savarese, S. Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 658–666. [CrossRef]
50. Bargoti, S.; Underwood, J.P. Image Segmentation for Fruit Detection and Yield Estimation in Apple Orchards. *J. Field Robot.* **2017**, *34*, 1039–1060. [CrossRef]
51. Sun, C.; Shrivastava, A.; Singh, S.; Gupta, A. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; IEEE: New York, NY, USA, 2017; pp. 843–852. [CrossRef]
52. Tian, Y.; Yang, G.; Wang, Z.; Wang, H.; Li, E.; Liang, Z. Apple detection during different growth stages in orchards using the improved YOLO-V3 model. *Comput. Electron. Agric.* **2019**, *157*, 417–426. [CrossRef]
53. Shorten, C.; Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. *J. Big Data* **2019**, *6*, 60. [CrossRef]
54. Skalski, P. Makesense.Ai. Available online: https://github.com/SkalskiP/make-sense (accessed on 18 July 2023).
55. Sutanto, A.R.; Kang, D.-K. A Novel Diminish Smooth L1 Loss Model with Generative Adversarial Network. In Proceedings of the Intelligent Human Computer Interaction, Daegu, Republic of Korea, 24–26 November 2020; Singh, M., Kang, D.-K., Lee, J.-H., Tiwary, U.S., Singh, D., Chung, W.-Y., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 361–368. [CrossRef]