remote sensing

Article

Detecting Cassava Plants under Different Field Conditions Using UAV-Based RGB Images and Deep Learning Models

Emmanuel C. Nnadozie [1,2,3], Ogechukwu N. Iloanusi [1], Ozoemena A. Ani [2] and Kang Yu [3,*]

[1] Department of Electronic Engineering, University of Nigeria, Nsukka 410002, Nigeria; emmanuel.nnadozie@unn.edu.ng (E.C.N.); ogechukwu.iloanusi@unn.edu.ng (O.N.I.)
[2] Department of Mechatronic Engineering, University of Nigeria, Nsukka 410002, Nigeria; ozoemena.ani@unn.edu.ng
[3] Precision Agriculture Lab, School of Life Sciences, Technical University of Munich, 85354 Freising, Germany
* Correspondence: kang.yu@tum.de

Abstract: A significant number of object detection models have been researched for use in plant detection. However, deployment and evaluation of the models for real-time detection as well as for crop counting under varying real field conditions is lacking. In this work, two versions of a state-of-the-art object detection model—YOLOv5n and YOLOv5s—were deployed and evaluated for cassava detection. We compared the performance of the models when trained with different input image resolutions, images of different growth stages, weed interference, and illumination conditions. The models were deployed on an NVIDIA Jetson AGX Orin embedded GPU in order to observe the real-time performance of the models. Results of a use case in a farm field showed that YOLOv5s yielded the best accuracy whereas YOLOv5n had the best inference speed in detecting cassava plants. YOLOv5s allowed for more precise crop counting, compared to the YOLOv5n which mis-detected cassava plants. YOLOv5s performed better under weed interference at the cost of a low speed. The findings of this work may serve to as a reference for making a choice of which model fits an intended real-life plant detection application, taking into consideration the need for a trade-off between of detection speed, detection accuracy, and memory usage.

Keywords: object detection; plant detection; YOLOv5; deep learning; computer vision; crop counting; precision agriculture

Citation: Nnadozie, E.C.; Iloanusi, O.N.; Ani, O.A.; Yu, K. Detecting Cassava Plants under Different Field Conditions Using UAV-Based RGB Images and Deep Learning Models. *Remote Sens.* **2023**, *15*, 2322. https://doi.org/10.3390/rs15092322

Academic Editors: Enrico Corrado Borgogno Mondino, Filippo Sarvia, Samuele De Petris and Tommaso Orusa

Received: 8 March 2023
Revised: 25 April 2023
Accepted: 26 April 2023
Published: 28 April 2023

1. Introduction

There is currently a global threat to food security, which is attributed mainly to world population growth, conflicts, and climate change [1,2]. Consequently, there is an increasing call for research directed at developing sustainable solutions to the food challenge; such that agrochemical inputs are minimized while increasing yield. These challenges present opportunities for innovations in farming technologies; thus encouraging the adaptation of existing cutting-edge technologies and the development of new methods [3].

Plant detection is an important step in farm management. Such farm activities that require plant detection include weed control [4], crop yield estimation [5], pest and disease detection, and spot spraying [6]. Not only is plant detection of interest in precision agriculture, but recently, ecologists have begun to highlight the importance of plant detection for biodiversity monitoring. For instance, in [7,8] plant detection is highlighted as an essential step in determining the ecological distribution of weeds, understanding their impact on the environment, as well as establishing an information system for their management. Another farm activity where object detection finds use is crop counting. To better estimate crop yield, it is vital to keep a count of the planted crops. Crop count at early growth stage can help determine the germination rate [9,10], which may provide better information on the plant vigor and could serve as a basis for better yield estimation [11,12]. Furthermore, plant counting helps to assess germination rate and make replanting decisions for low crop density areas of the farm [13–15]. Insights from plant count, such as inconsistent spacing, small gaps, and wide empty areas, can help to determine potential problems in the farm such as disease and pest infestation, waterlogging, or excess agrochemical application, which in turn form the basis for necessary control interventions such as site-specific managements [15,16].

Manual counting of the plants can be a huge burden as well as time consuming. Computer vision techniques provide an accurate and time-saving approach to crop counting [11]. Therefore, the development and deployment of plant detection models on autonomous and semi-autonomous farm vehicles including terrain vehicles and, more recently, unmanned aerial vehicles (UAVs) offer great potentials and alternative approaches for site- and plant-specific management. Such vehicles are fitted with sensors that feed data to the detection model which identifies the plants using computer vision techniques [17]. Traditional image processing techniques such as color thresholding, filtering, and differencing are methods first studied by researchers for plant detection [18]. More recently, learning-based methods such as support vector machines, artificial neural networks, random forests, among others [19–21] have been shown to have superior performance over traditional image processing methods for plant identification. While traditional machine learning methods showed promising results in controlled environments, their accuracy is undermined in real-life field conditions. Their lack of robustness to field conditions such as natural lighting, varying plant density, leaf occlusions, and different plant growth stages necessitated the development of deep learning methods for plant detection [22]. In addition to the robustness of deep learning methods to field variations, they are capable of end-to-end learning of hierarchical features of plant images, thus eliminating the need for the laborious task of feature engineering required in machine learning.

Convolutional neural networks including AlexNet, VGG16, Inception-ResNet, etc., are among the deep learning methods employed for plant detection [23–25]. Gao et al. [26] developed a CNN model called CMPNet based on ResNet and SeNet for identification of 30 wheat varieties in multiple growth stages including the tillering stage. Their method recorded over 90% accuracy for all growth stages. Region proposal-based models such as Faster-RCNN and Mask-RCNN were used by [22,27,28] for plant detection. While such models deliver promising results in terms of accuracy, because they use a two-stage detection pipeline, they have slower inference speeds than their regression-based counterparts such as the Single Shot Multibox Detector (SSD) and YOLO networks [28,29].

The YOLO object detection model was introduced by Redmon et al. [30]. Unlike previous object detection models such as Faster R-CNN [31] which employ a two-stage process of region proposals followed by bounding box classification, YOLO is a single-stage detector performing all the object detection processes in a single network. The regression-based YOLO model is favored for real-time applications as it overcomes the bottlenecks posed by the complex architecture of the region-based detectors, greatly reducing the inference time, while enhancing generalizability of the model. YOLO networks have been deployed in weed detection for high inference speed which makes them attractive for real-time applications. YOLOv3 was deployed on a ground robot for real-time weed detection in a carrot farm; and the setup achieved up to 89% precision [32]. Gao et al. developed a detection model based on Tiny YOLO for detecting *Convolvulus sepium* in sugar beet fields and achieved an average precision (AP) up to 0.897 [33]. Xu et al. [34] in their paper demonstrated the use of YOLOv5 for maize leaf counting. Their results showed a good accuracy for different growth stages of the maize plant. Mota-Delfin et al. [35] compared different versions of the YOLO detection model for whole maize plant counting at different growth stages. Their results showed that YOLOv5 performed better than YOLOv4, YOLOv4-tiny, YOLOv4-tiny-3l. Nevertheless, it is unclear whether deploying a lightweight YOLO model onto UAV allows for continuous monitoring across growth stages and under varying illumination and field interference scenarios effectively.

In spite of the plethora of works on plant detection using deep learning methods, research on the deployment of plant detection models on embedded devices and evaluation of such systems for real-time performance is lacking. Such research becomes necessary given that various farm management tasks such as weed detection with selective herbicide spraying requires real-time plant identification. Furthermore, where there exists the desire to incorporate real-time plant detection in a given farm activity, the farmer or agricultural stakeholder should have the details of the real-time performance of the model in terms of detection speed, accuracy, and memory requirements. These details are key to making a choice of which detection model to select. Model selection will often be influenced by the application requirements (such as minimum acceptable accuracy and/or detection speed) and available resources (such as computing power of the embedded system on which the model would run). Thus, given the real-time performance data of the model, the user is able to make an informed tradeoff between accuracy, speed, and memory requirement in determining the suitability of the model for the intended application. Importantly, a dataset reflecting a wide range of real field conditions such as variations in natural illumination, presence of shadows, and leaf occlusion is necessary for real-time applications of plant detection. Moreover, little research in plant detection has focused on crops peculiar to sub-Saharan Africa such as cassava (*Manihot esculenta*) which is a major food crop in sub-Saharan Africa [36].

The objectives of this study were to: (i) build a representative dataset for cassava plant detection; (ii) train and evaluate the performance of state-of-the-art object detectors on the cassava dataset; and (iii) deploy the selected model on an actual embedded device and evaluate its real-time performance on use cases including UAV-based selective spraying and counting plants under varying field conditions.

2. Materials and Methods

2.1. Dataset

2.1.1. Experimental Setup

For this research, an experimental cassava farm was set up. The farm setup process included land selection, clearing, ploughing, ridging, planting, and manuring. The cassava cuttings were planted at distances of 1 m apart according to the standard recommendations by IITA for cassava cultivation in the derived savanna region of Nigeria [37].

2.1.2. Image Acquisition

A 550 mm quadcopter was constructed for the purpose of capturing aerial images of the farm. A GoPro Hero 7 camera (GoPro Inc., California, CA, USA), with a CMOS of 1/2.3-inch producing a pixel resolution of 12 megapixels, was mounted on the quadcopter to capture RGB images of the farm. The farm images were captured at altitudes ranging between 2 m to 3 m between the months of September and October, 2021, specifically, 25, 29, and 60 days after planting date. The data collection dates fall into two major growth stages of cassava plant: the first two dates fall into the early growth stage, while the last date falls in the vegetative growth stage. The images were collected under different natural lighting conditions, cassava growth stages, as well as varying weed densities and leaf occlusions. Each of the captured images were of pixel size 4000 × 3000 pixels. Image capture times ranged from 11 a.m. to 4 p.m.

2.1.3. Image Annotation

The annotation of the images involved drawing bounding boxes around the objects of interest and assigning object classes to the bounding boxes. The YOLOv5 requires that the bounding box dimensions be normalized to the range [0, 1] and represented in the "c x y w h" format; where c is the object class index (starting from 0), x and y represent the x-y coordinates of the center of the bounding box, and w and h represent the width and height of the bounding boxes, respectively. For each image, the annotations must be saved in a .txt file, where each bounding box information is entered on a new line. The python-based LabelImg [38] graphical annotation tool was chosen for image annotation in this work because of its ease of use and ability to save the annotations in the YOLOv5 format.

2.1.4. Data Augmentation

To avoid overfitting to the data, this work defaulted to the use of the inbuilt train-time data augmentation of YOLOv5. This includes image-space and color-space transformations and the novel mosaic data augmentation proposed by the author of YOLOv5. The mosaic augmentation performs random image augmentations of the original image and assembles it into a mosaic image consisting of the original image and a number of randomly augmented images. The data augmentation can be modified in the hyperparameters which includes options for specifying or even disabling augmentation types and values. By adjusting the augmentation hyperparameters, our model was exposed to broader semantic representations of the training data.

2.2. Object Detection Model

2.2.1. Model Training and Validation

The YOLOv5 release comprised of YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x variants with accordingly increasing weight file size, training time, number of parameters, and model accuracy. Thus, YOLOv5n has the smallest weight file, least training time and number of parameters, making it suitable for deployment on memory constrained devices. Conversely, YOLOv5x the largest and most accurate of the variants is best suited for applications where very high accuracy is a priority and memory is not a problem. Since our target was for real-time applications, in this work, we chose the two smallest models, YOLOv5n and YOLOv5s for their lightness, high inference speeds, and acceptable accuracy. For simplicity, YOLOv5n and YOLOv5s will be referred to as Y5n and Y5s, respectively.

The YOLOv5 model training and inference was performed on NVIDIA GeForce 3060 on our local workstation using Jupyter Notebooks.

The model was trained and validated on one hundred and twenty-five images containing over 1700 cassava objects. The dataset was split into a train:validation ratio of 80:20. Additionally, a different set of images were used to evaluate the real-time detection speed of the models with the Jetson Orin embedded GPU.

The model was trained for different combinations of input image and batch sizes. These combinations were also trained on multiple models of the YOLOv5 as mentioned previously. The range of image input sizes include 960 $\times$ 720, 640 $\times$ 480, 512 $\times$ 384, and 256 $\times$ 192; while the batch size ranged from 8 to 64. The maximum batch sizes for training on 960 $\times$ 720 and 640 $\times$ 480 image sizes were limited to not more than 20 and 32, respectively, because the GPU memory did not permit loading more than these limits for the respective image sizes. The results were evaluated to identify which model and parameter combinations gave the best performance for real-time applications. For all the training instances, a fixed epoch of 400 was used.

Given that for computer vision tasks, using pretrained weights can lead to better accuracy than using random weights [39], we used the pretrained weights corresponding to the respective models which were trained on the COCO dataset [40] by the YOLOv5 author. We set the initial learning rate at 0.01, the momentum at 0.937, and the weight decay at 0.0005.

2.2.2. Performance Metrics

The precision and recall, and mean average precision of the model were used for evaluation of the model performance. The precision is the degree to which the model correctly predicts the objects. The precision, P is represented by the relation

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

where TP is true positives, FP is false positives. As observed from Equation (1), the higher the false predictions, the lower the precision.

Recall on the other hand is the degree to which the model captures all ground truth. Recall, R is given by Equation (2)

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

where FN is false negative.

The mean average precision, mAP, has become the acceptable metric in the research community for evaluating object detection models. The mAP indicates the weighted mean of precisions obtained for different intersection over union (IOU) thresholds. In this work, the precision, recall, mAP@0.5, and mAP@0.5:0.95 of all models were extrapolated.

2.3. Model Deployment on NVIDIA Jetson AGX Orin

To obtain the real-time performance data of the models, it was necessary to deploy the models on actual devices that are used in the field, not on the workstation GPUs used in the lab. The NVIDIA® Jetson AGX Orin™ (NVIDIA Corporation, Santa Clara, CA, USA), or Jetson Orin for short, was our choice of embedded GPU for real-time model performance evaluation. The Jetson Orin is the latest entry in the NVIDIA Jetson embedded GPUs, featuring up to 1792 NVIDIA CUDA cores and 56 tensor cores, and capable of hosting large and complex deep learning-based object detection models. The Jetson Orin can be mounted on both aerial and ground farm vehicles for real-time field applications.

2.3.1. Jetson Orin Setup

The NVIDIA® Jetson AGX Orin™ development kit consisted of the Jetson Orin module and the reference carrier board with attendant accessories. The Ubuntu 20.04 is the operating system installed on the Jetson Orin. The Jetpack 5.0 provided the software that powers the Jetson Orin module, and includes L4T with Linux kernel 5.10 and reference file system based on the Ubuntu 20.04.

To be able to run YOLOv5 models on the Jetson Orin, Pytorch, Torchvision, and all dependent packages were installed on the Jetson Orin. The YOLOv5 model was cloned from the GitHub repository of the authors and all dependent packages were installed.

2.3.2. Inference on Jetson Orin

Selected weights, representative of the trained models, were used to detect cassava plants on the inference images. For both Y5s and Y5n, and for each image size, the weights of the models trained on the highest batch size were selected for the inference. The detection process consists of the preprocessing, inference, and non-maximum suppression (NMS) stages. The detection algorithm outputs the latency (in milliseconds) of the three stages above. Then the detection speed (in frames per second, fps) is obtained as the inverse of the sum of the three outputted latencies as shown in Equation (3).

$$Speed\ (fps) = \frac{1000}{preprocess(ms) + inference(ms) + Non - Maximum\ Suppression(ms)} \tag{3}$$

The detection speed of the of the models were compared to highlight real-time model performance for the two YOLO versions given different image sizes.

A real-life use case scenario was explored to highlight the practical implication of the findings of the work. Equation (4) was used to calculate the UAV flight mission duration for the use case (see Section 3.3.3).

$$fastest\ mission\ duration\ (s) = {}^{Farm\ size\ (m^2)}\Big/_{(detection\ speed\ (fps)\ x\ GSD\ (m^2))} \tag{4}$$

2.4. Validation for Cassava Counting

Eighteen inference images were selected to evaluate the model performance in cassava counting. A regression plot showed the models' cassava count versus the actual cassava count. The performance of the models under varying field conditions including weed density, leaf occlusion, illumination, and growth stages were examined.

3. Results

3.1. Captured UAV Images

Example images captured in this study showed that the imaging conditions varied in their illuminations (Figure 1a,b,d), soil coverage (Figure 1c,d), and weed infestation conditions (Figure 1a,d).

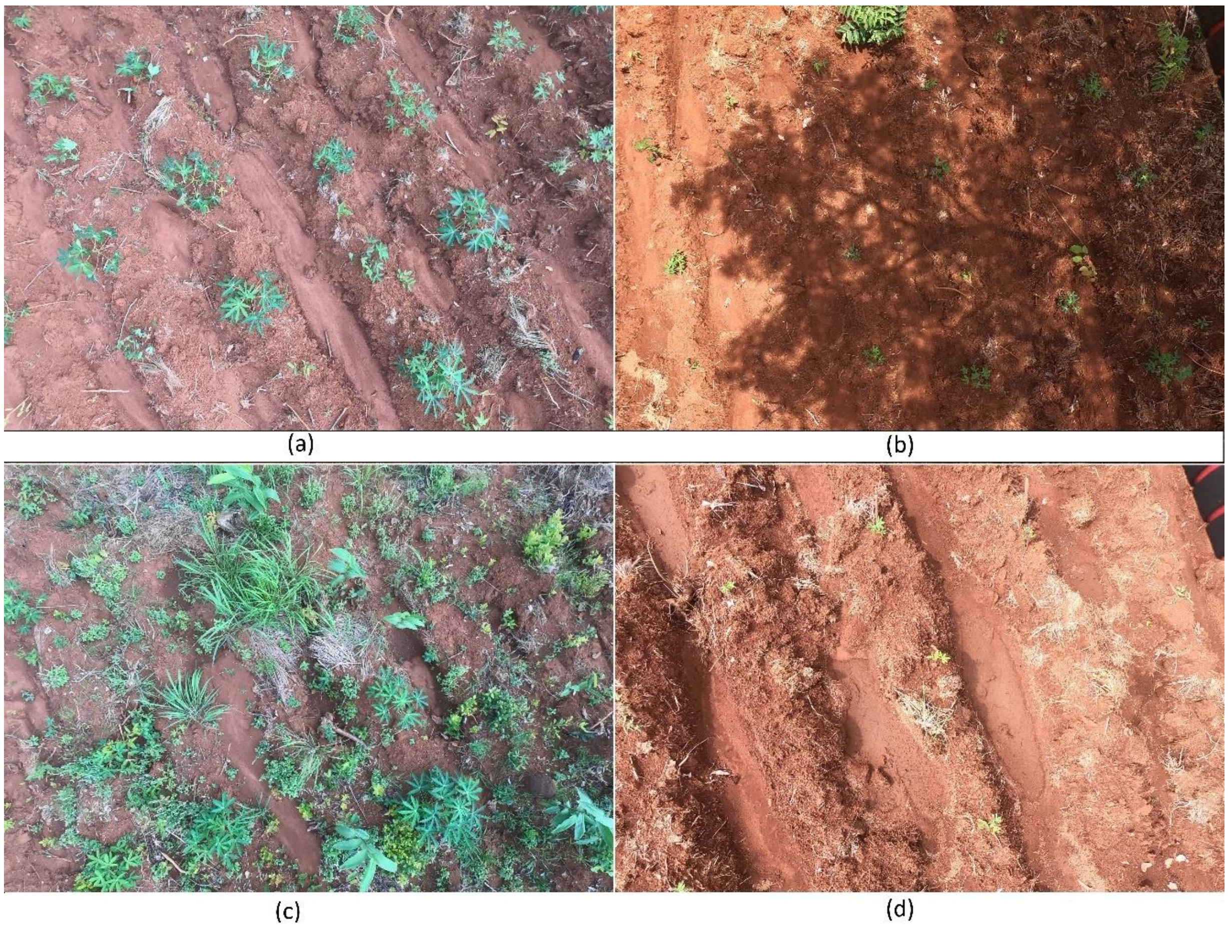

Figure 1. UAV field images showing varying field conditions; (**a**) low lighting, low weed density, vegetative cassava growth stage, (**b**) high illumination with shadow, (**c**) high weed density with high leaf occlusion, (**d**) high illumination with early cassava growth stage.

To improve the robustness of the models, images were also captured in different row orientations. Images showing different row orientations are shown in Figure 2.

Sample annotated images are shown in Figure 3. Rectangular bounding boxes are drawn over each cassava object. The object class labels are also shown.

3.2. Model Training Performance

Table 1 shows the summary of the training results. The training duration, precision, recall, mAP@0.5, and mAP@0.5:0.95 are shown for different input image sizes, batch sizes, as well as for the two YOLOv5 models. While we present in the table all the aforementioned performance metrics for the sake of readers who may be interested in the details, our discussion will consistently reference the mAP@0.5:0.95 which is the more widely accepted performance metric in the object detection research community.

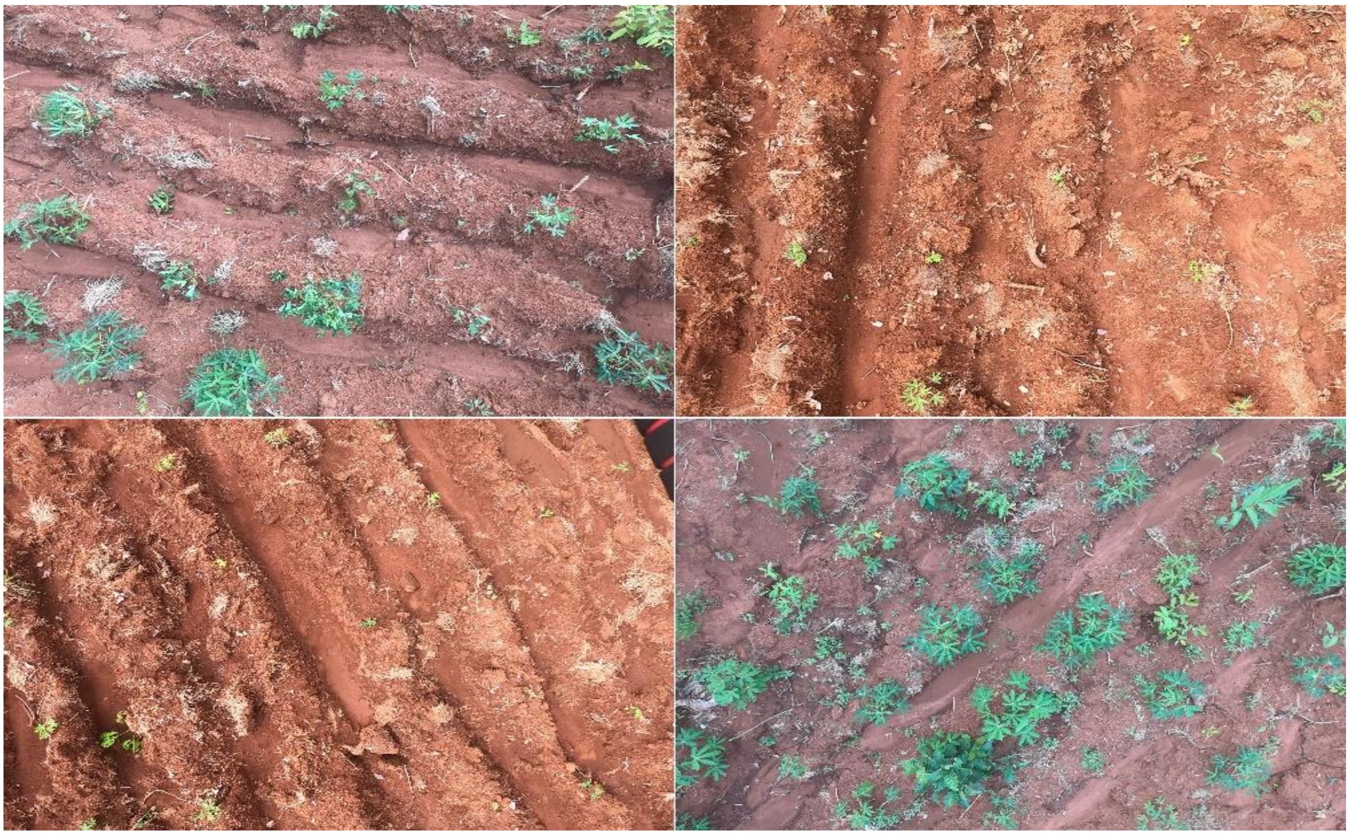
Figure 2. UAV field images showing different row orientations.

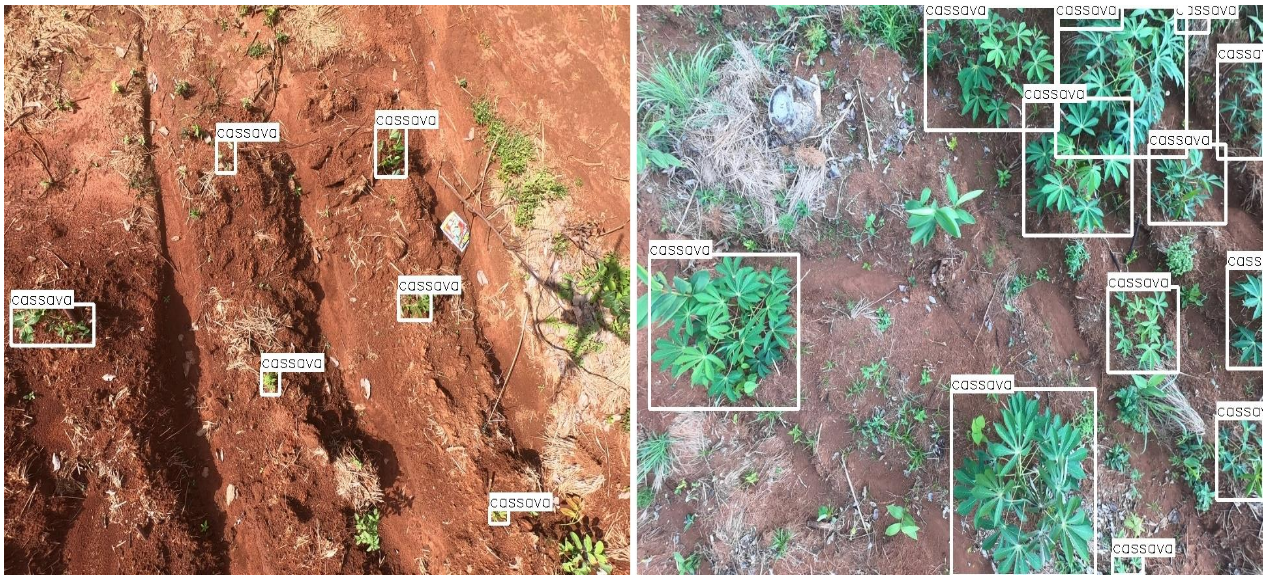

Figure 3. Annotated field images with object class labels.

3.2.1. Performance of the Models for Varying Image Sizes

The plot in Figure 4 shows the curves of the best mAP@0.5:0.95 for each image size for each YOLOv5 model. The curves have been smoothened for more visibility.

Table 1. Summary of model training results.

		Y5n					*Y5s*				
Image Size	**Batch Size**	**Train Time (h)**	**Precision**	**Recall**	**mAP@0.5**	**mAP@0.5:0.95**	**Train Time (h)**	**Precision**	**Recall**	**mAP@0.5**	**mAP@0.5:0.95**
960 × 720	20	0.344	0.951	0.897	0.939	0.724	0.402	0.967	0.918	0.960	0.791
	16	0.354	0.959	0.886	0.947	0.736	0.422	0.954	0.922	0.965	0.793
	8	0.433	0.954	0.907	0.946	0.740	0.482	0.960	0.918	0.955	0.798
640 × 480	32	0.234	0.950	0.838	0.903	0.628	0.276	0.978	0.834	0.918	0.674
	16	0.294	0.911	0.864	0.911	0.637	0.316	0.938	0.867	0.917	0.679
	8	0.403	0.927	0.853	0.904	0.635	0.431	0.962	0.859	0.924	0.695
512 × 384	64	0.185	0.881	0.781	0.836	0.512	0.224	0.897	0.810	0.867	0.566
	32	0.227	0.947	0.760	0.854	0.551	0.253	0.930	0.808	0.880	0.600
	16	0.279	0.932	0.783	0.868	0.560	0.318	0.906	0.829	0.886	0.605
	8	0.389	0.916	0.785	0.861	0.559	0.422	0.914	0.848	0.888	0.610
256 × 192	64	0.169	0.786	0.602	0.658	0.326	0.191	0.877	0.648	0.721	0.386
	32	0.214	0.841	0.659	0.722	0.369	0.232	0.837	0.695	0.738	0.412
	16	0.274	0.864	0.659	0.726	0.385	0.289	0.886	0.709	0.759	0.414
	8	0.352	0.855	0.644	0.723	0.384	0.371	0.892	0.682	0.749	0.433

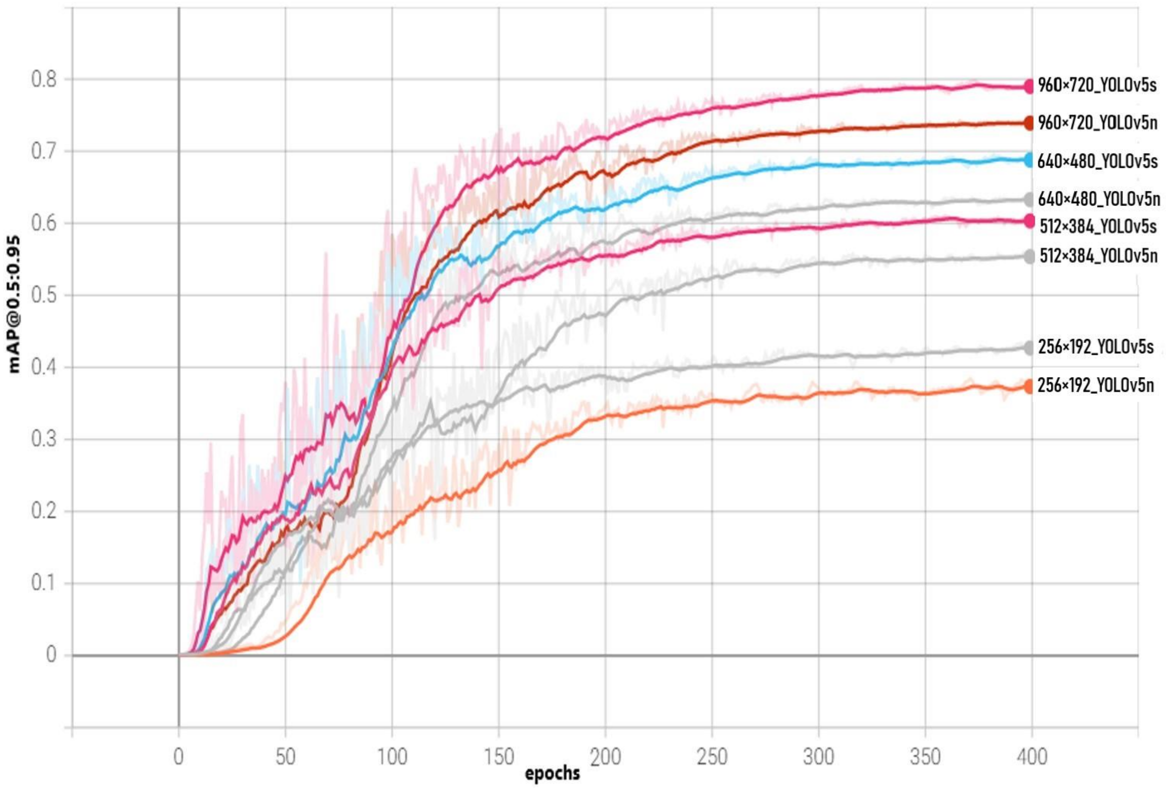

Figure 4. mAP@0.5:0.95 for various image sizes.

The chart in Figure 5 shows a plot of the training time and mAP@0.5:0.95 for all image sizes and across all batch sizes.

3.2.2. Effect of Varying Batch Sizes on the Model Performance

Figure 6 is a plot of mAP@0.5:0.95 for different batch sizes.

3.3. Model Inference Performance

3.3.1. Detection Speed

We present in Table 2 a summary of the inference results with additional columns showing accuracy and weights file size.

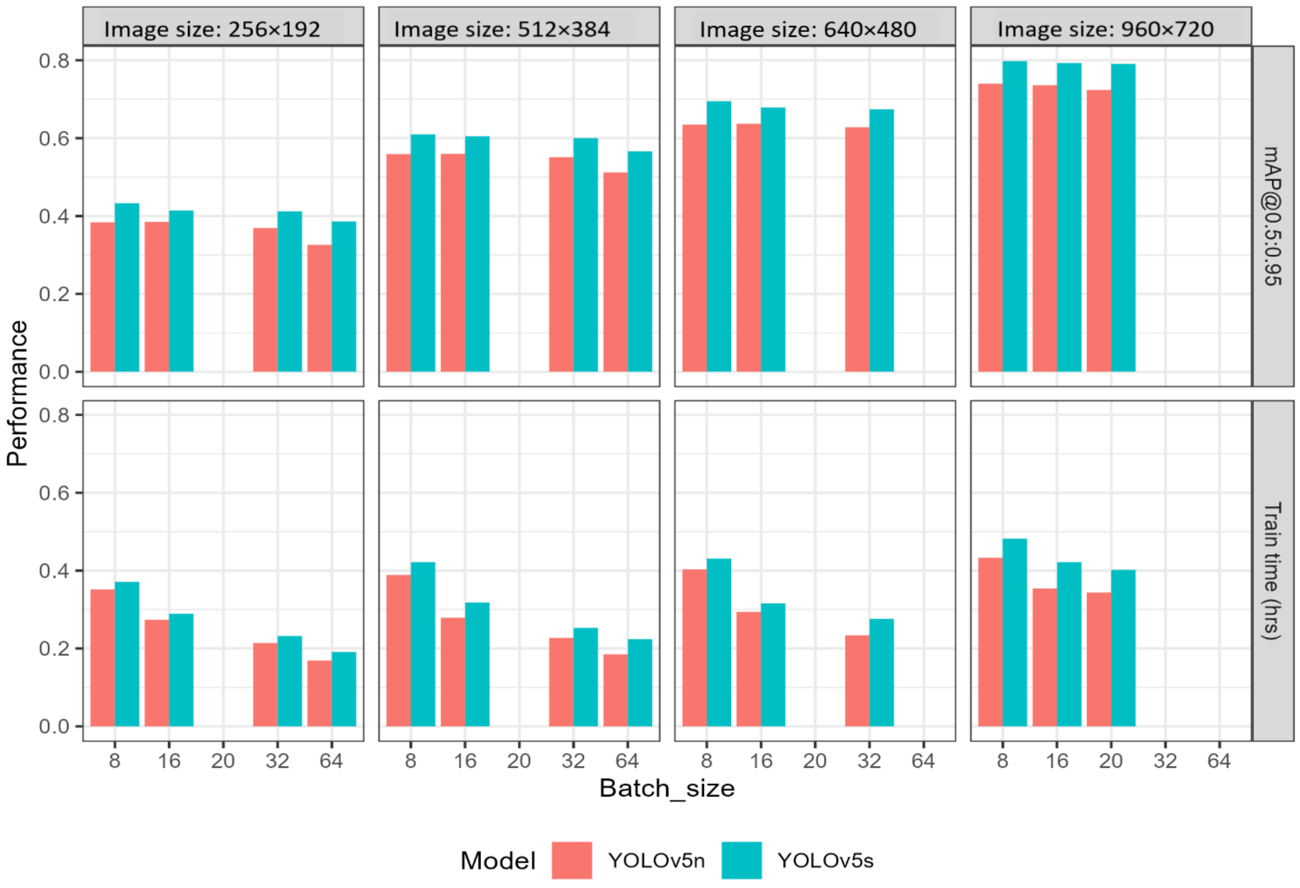

Figure 5. Training time and mAP@0.5:0.95 for all image sizes and batch sizes.

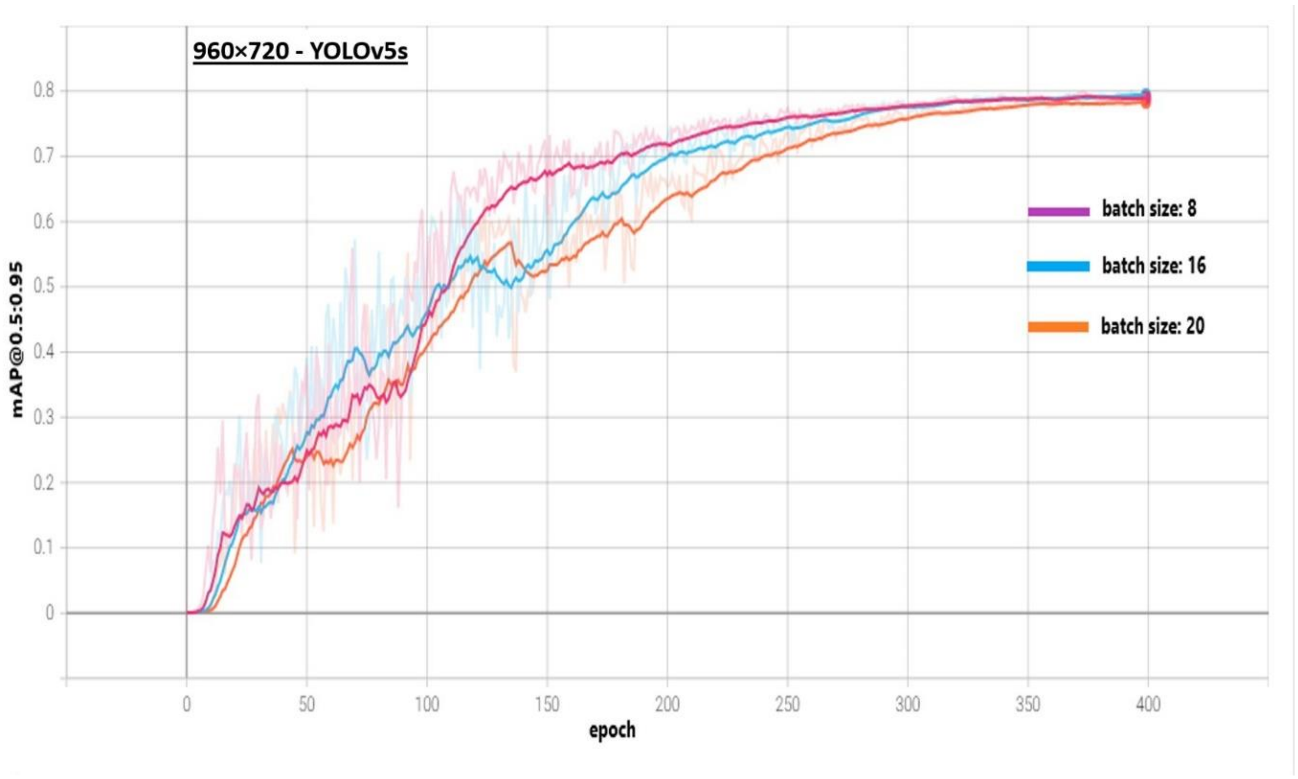

Figure 6. mAP@0.5:0.95 for different batch sizes.

Table 2. Inference on Jetson Orin showing detection speed.

Model	Image Size	Speed (FPS)	mAP@0.5:0.95	Weights File Size (mb)
Y5n	960 × 720	38.26	0.724	4.0
	640 × 480	52.67	0.628	3.8
	512 × 384	54.14	0.512	3.7
	256 × 192	58.32	0.326	3.6
Y5s	960 × 720	34.79	0.791	14.3
	640 × 480	47.60	0.674	14.1
	512 × 384	50.18	0.566	14.0
	256 × 192	57.08	0.386	13.9

Figure 7 visualizes the detection speeds of the models.

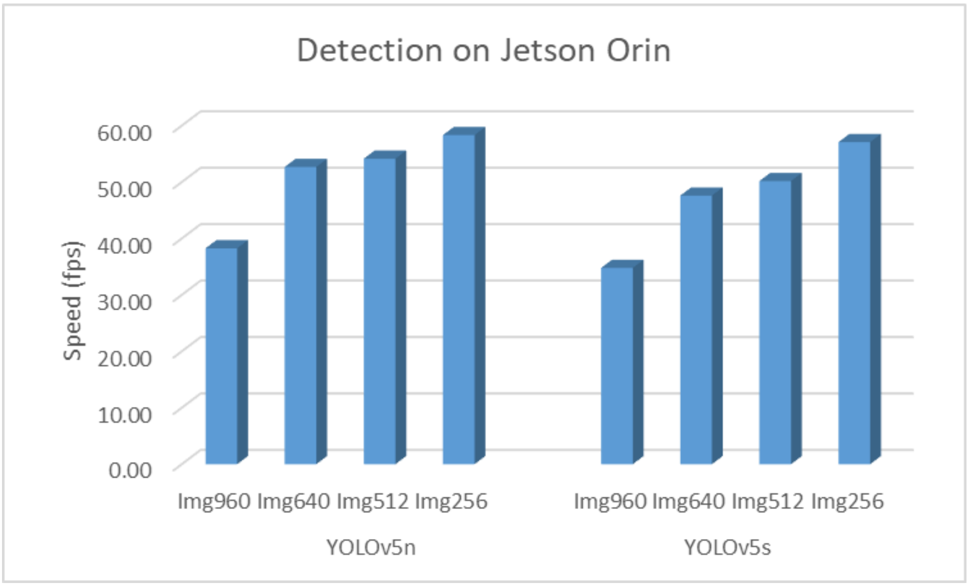

Figure 7. Inference speed of the models on test images.

3.3.2. Speed vs. Accuracy

In Figure 8, a plot of speed vs. accuracy is presented to highlight the relationship between speed and accuracy across the two models over different image pixel resolutions.

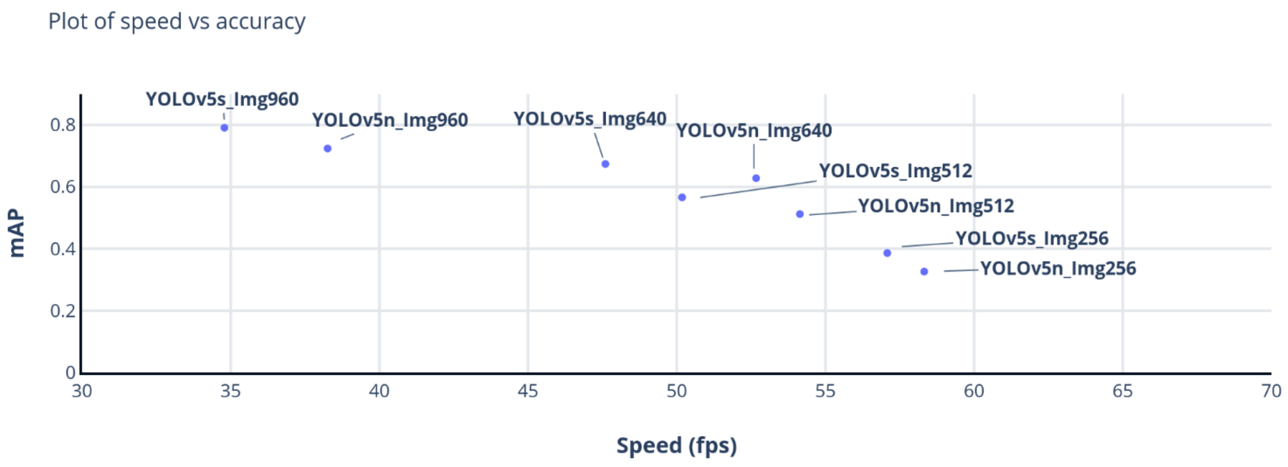

Figure 8. Plot of detection speed vs. mAP@0.5:0.95.

3.3.3. Use Case on a Farm Field

The chosen application scenario is the use of a UAV for real-time selective spraying in a cassava farm of size 60,000 m^2. The drone captures images of the farm at an average altitude of 2.5 m. The images are fed into an onboard Jetson Orin embedded GPU for cassava detection, which triggers the sprayer. The spray amount is varied based on the area of the image frame covered by the detected cassava.

The images are captured by a GoPro Hero 7 camera at 4000 × 3000 pixels and resized on the GPU to one of the four input resolutions used in this work. At an altitude of 2.5 m, the ground sampling distance (GSD) was calculated to be 5.14 m by 3.86 m (19.84 m^2). The ground measurement of a singular pixel in an image is referred to as the ground sampling distance. Based on the model's detection speed, we can obtain the fastest possible time an agricultural drone loaded with each of the models can complete a single mission. This projection is based on the idea that the UAV can fly only as fast as the model can detect cassava. As an example, for the Y5n at 960 × 720 image pixel resolution, with detection speed of 38.26 fps, the UAV cannot fly faster than 38.26 times the GSD per second. The fastest possible mission durations were calculated using Equation (4). The results are shown in Table 3.

Table 3. Fastest possible UAV mission completion time of each model.

Model	Image Size	Speed (fps)	Best Possible Mission Duration (s)
Y5n	960 × 720	38.26	78.31
	640 × 480	52.67	57.42
	512 × 384	54.14	55.85
	256 × 192	58.32	51.86
Y5s	960 × 720	34.79	86.93
	640 × 480	47.60	63.53
	512 × 384	50.18	60.27
	256 × 192	57.08	52.98

3.4. Model Performance for Cassava Counting

The Root Mean Square Error (RMSE) and R^2 for cassava counting using Y5s and Y5n are presented in Table 4.

Table 4. RMSE for cassava counting with Y5s and Y5n.

	Y5s	Y5n
RMSE	0.82	1.31
R^2	0.9982	0.9949

Figure 9 shows a plot of the predicted count versus the actual count.

3.4.1. Light Conditions

Figure 10 shows Y5s and Y5n models inferenced on the same image with shadow. The detected plants are bounded by green boxes. The circled objects in white were missed by the models.

3.4.2. Growth Stages

Figure 11 shows the model inference at vegetative stage. The detected plants are bounded by green boxes. The circled portion in white shows a missed detection due to high leaf occlusion.

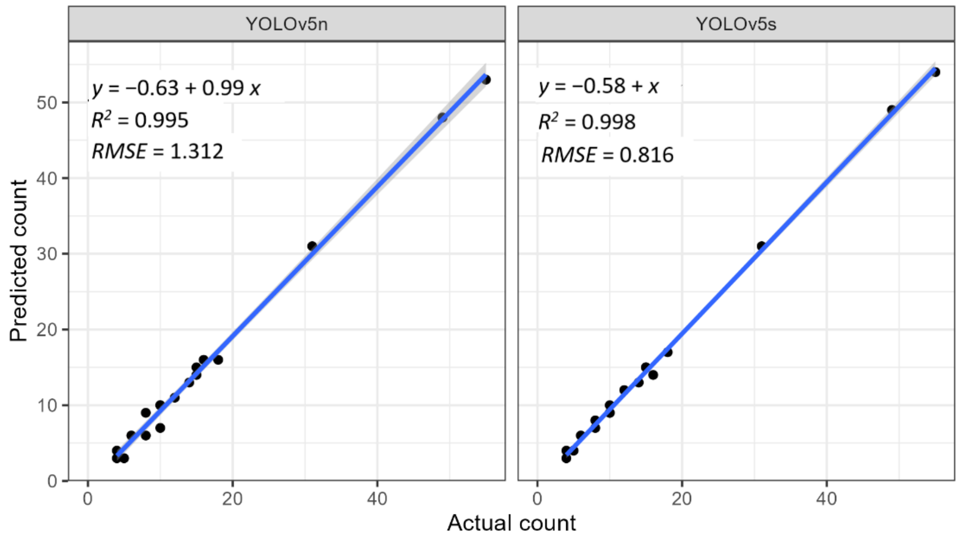

Figure 9. A plot of actual cassava count versus predicted count of Y5s and Y5n.

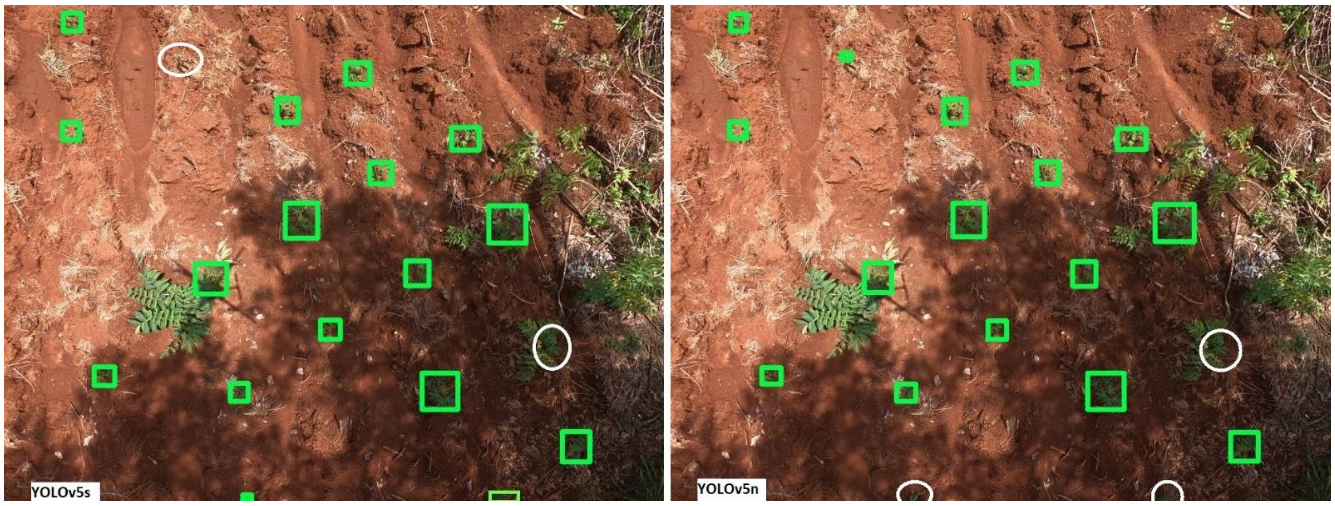

Figure 10. Inference on images with shadow, missed cassava plants are circled.

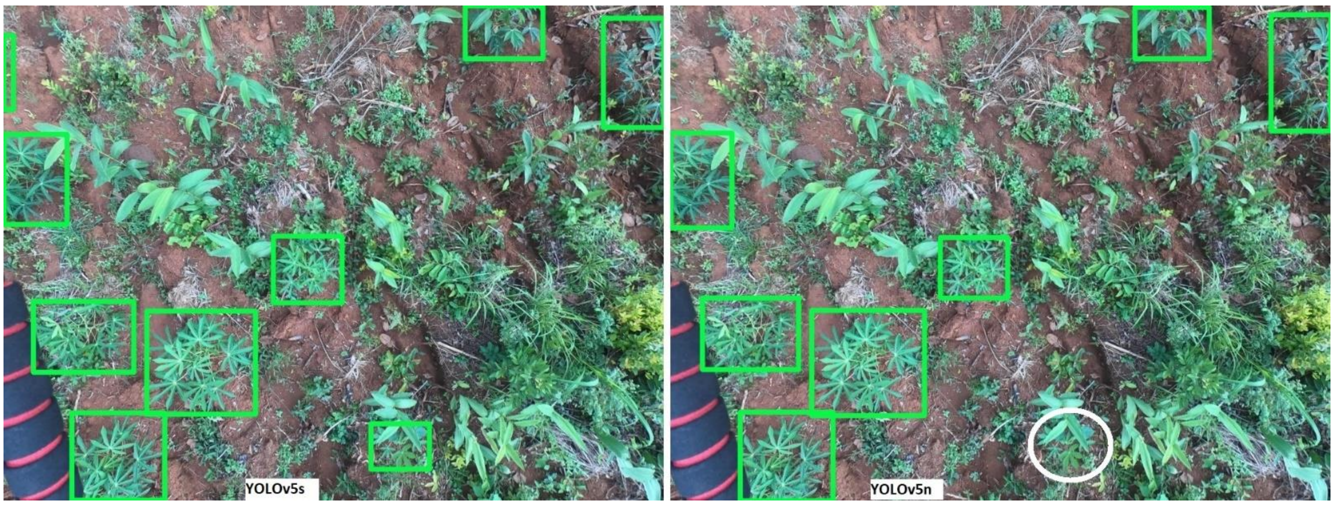

Figure 11. Inference on test image showing high leaf occlusion.

3.4.3. Weed Density

Figure 12 shows prediction under high weed density. The detected plants are bounded by green boxes. Incorrect predictions are circled in white.

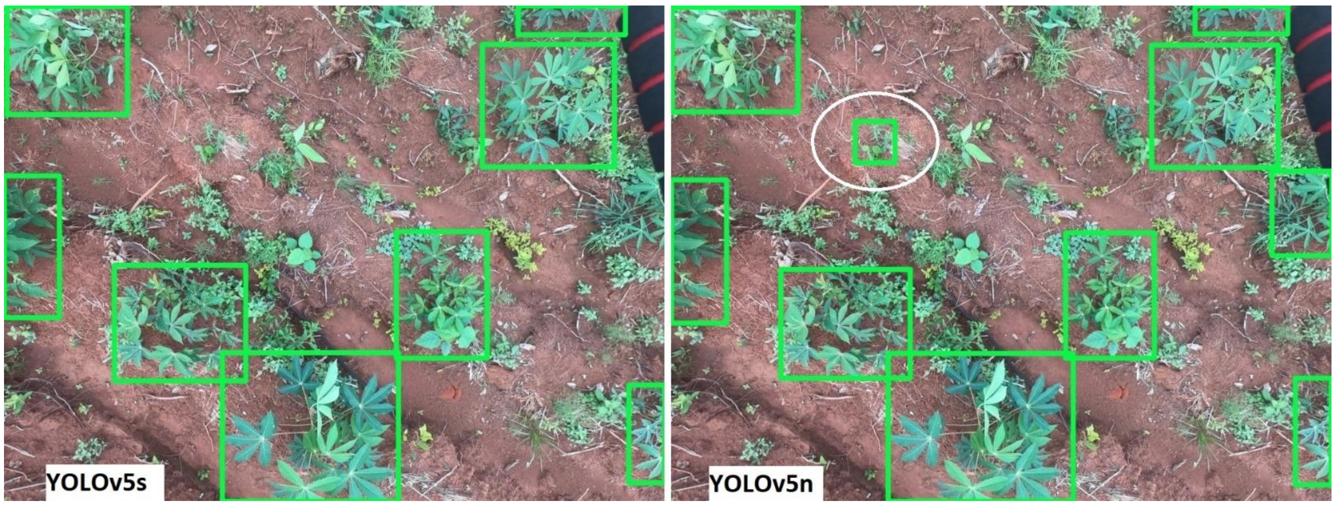

Figure 12. Inference image showing high weed density; Y5s with generally higher prediction confidences, andY5n with generally lower prediction confidences.

4. Discussion

4.1. Captured UAV Images

As shown in Figures 1 and 2, the dataset reflects real field conditions including variations in natural illumination, different cassava growth stages, varying weed densities, leaf occlusions, and shadow presence. Furthermore, in Figure 3, the bounding boxes are shown to be drawn as tightly as possible about the cassava plants to enhance learning of the cassava features by the models [41,42]. Consequently, as would be discussed later, the models showed robustness and suitability for real field applications.

4.2. Model Training Performance

4.2.1. Performance of the Models for Varying Image Sizes

The mAP@0.5:0.95 decreases as the image pixel resolution decreases as shown in Figure 4. This reduction in performance is because it is more difficult for the models to resolve small objects as the image size decreases [43]. However, the increase in performance with increase in image size comes at the cost of increased training time for the models, as shown in Figure 5. This is due to the fact that the larger the image size, the more memory and computational resources are required to process it; as well as more features to learn. Therefore, for applications where training time is critical or memory constraint is high, lower image sizes may be used, but at the expense of model accuracy. It is important to note that the YOLOv5 cannot guarantee detection accuracy for objects with bounding boxes less than 3 pixels, and will throw a warning at training time [44]. In this work, such warnings were noted for images sizes of 256 $\times$ 192.

We also note that for each image size, the Y5s model performs better than the Y5n model. The better performance of Y5s is because it is able to learn better than Y5n, having more feature extraction modules, convolution kernels, leading to more learnable parameters [45]. However, this is at the expense of having a heavier weights file than the Y5n. For instance, the Y5s learns with over seven million parameters and has a weight file of 14.6 Mb, whereas the Y5n has less than two million parameters with a weight file size about three times less at 4.1 Mb. The effect of weights size is especially critical in real-time applications where memory constrained non-GPU devices are employed [46]. The detection speed at inference time increases as the number of processed parameters reduces.

Interestingly, as observed in Figure 5, an interplay of batch size, image size, and model selection could lead to better performance of the Y5n model. For instance, we extrapolate from our results as shown in Figure 5 that training Y5n on 960 × 720 images yields higher mAP@0.5:0.95 than training Y5s on any of the lower image sizes. The only cost is a slightly higher training time. This may be desirable for applications such as real-time weed detection from UAV images where the resolution needs to be as high as possible and the detection model as small as possible to fit the memory constraints for UAV-mounted processors.

4.2.2. Effect of Varying Batch Sizes on the Model Performance

For all image sizes, the training time increases with decrease in batch size as shown in Figure 5. This implies that taking more images per batch is more memory efficient. However, from the mAP@0.5:0.95 curve for different batch sizes of 960 × 720 input image size shown in Figure 6, which is fairly representative of other image sizes, the difference in mAP@0.5:0.95 is not very significant. Thus, we conclude that the major advantage of higher batch sizes is the decrease in training time.

4.3. Model Inference Performance

4.3.1. Detection Speed

Over 50% of the models have detection speeds above 50 fps (Table 2). The lowest recorded speed of 34.79 fps surpasses the commonly acceptable speed for most real-time applications [47]. This implies that these models can be deployed in applications requiring very fast detection [17]. A use case scenario to demonstrate this is discussed in Section 4.3.3.

4.3.2. Speed vs. Accuracy

In Figure 8, it is observed that, generally, detection speed decreases with an increase in image sizes across both model versions. For each image size, the Y5n was faster than the corresponding Y5s. This is expected given that the Y5s models require more computing time because of their deeper architecture and higher number of parameters as evidenced in the weights size [29]. Therefore, to obtain higher detection speeds, one might have to settle for lower image sizes that lead to reduced accuracy [29].

4.3.3. Use Case on a Farm Field

The best possible mission speeds for the use case of plant detection and selective spraying, as presented in Table 3 are theoretical. For instance, Y5n on 960 × 720 resolution, to attain the best mission duration of 78.31 s, the UAV will have to travel at a speed of 38.26 times the GSD (m/s) which is 147.68 m/s. Most UAVs will not travel at this speed for agricultural applications. Even if they have to, there has to be a zero latency in other components of the system such as the spraying trigger response time, if the UAV must execute the spraying mission with the required accuracy. Overall, our work highlights the fact that faster detection offers the possibility of increasing flight speed, thus reducing the flight time; hence, more missions can be completed. This is an important consideration for users that offer agricultural drone services. The faster the model, the less time they spend on a mission, and the more customers they can cater to.

4.4. Model Performance for Cassava Counting

Since the inference speed of all the models were acceptable for real-time application, for comparison, for cassava counting demo, we selected the best performing model instance in terms of mAP@0.5:0.95 each for Y5n and Y5s. The first is the Y5n with 960 × 720 input image resolution and training batch size of eight; the second is the Y5s with 960 × 720 input image resolution and training batch size of eight. This selection of just two model instances was also performed with the understanding that the results of the comparison across the models will be similar as pixel resolution decreases, but with the afore established attendant performance degradation that follows reduction in input image size. Y5s had better RMSE and R^2 values than Y5n as shown in Figure 9. However, both models showed good results.

4.4.1. Light Conditions

Examination of the test images with high lighting as well as with low lighting revealed that both model instances showed similar performance in terms of detecting the cassava objects. However, in high illumination images with large shadows from nearby trees, the Y5n model missed more cassava plants under the shadow (Figure 10). This could be because the percentage of training images with large shadows was very little compared to images without shadows, therefore the model had insufficient images with shadows to learn from. This could be solved by collecting and re-training the model with more images with shadows. Moreover, this is not a major performance setback for real-life scenario given that large shadows (usually from big tress) are a rare occurrence in farms where UAVs are deployed for cassava counting.

4.4.2. Growth Stages

For images with early growth stage of the cassava, both model instances had no difficulty detecting almost all cassava objects. This high performance could be partly due to the fact that the early growth stage of the cassava corresponds to the stage of little or no weed on the farm. Therefore, the challenge of detecting cassava plants in the presence of leaf occlusions was absent. Both models also performed well on images of the later vegetative growth stage of the cassava. However, some cassava plants were missed by the Y5n model. These were mainly the cassava plants that had delayed growth and still appeared little or were extremely occluded by the weeds. Y5n model failed to detect a highly occluded cassava plant (circled in Figure 11), which was successfully detected by the Y5s model.

4.4.3. Weed Density

As the weed density increased, the degree of plant leaf occlusion increased. Though the models performed very well in detecting cassava plants on images of the early growth stage of cassava which corresponded to the period of almost zero weed density; however, for images with high weed density, we observed that though the detection was good for both models, the Y5n model predicted with less confidence than the Y5s model for objects with high leaf occlusion. We also observed some incorrect predictions with the Y5n model. Figure 12 shows both models inferenced on the same test image with Y5s yielding higher prediction confidence levels. The incorrect prediction is circled in Figure 12.

4.5. Limitations

Whereas the dataset richly reflected varying field conditions, some equally important field conditions such as cassava plants with stress symptoms such as wilting or diseases such as cassava mosaic which could alter the leaf morphology were not present. Furthermore, weeds with very high visual resemblance to cassava such as hemp were not present in the dataset. The absence of images having the stress symptoms earlier mentioned was due to the authors' aim of truly representing the prevalent real field conditions. The cassava growers in the study region commonly observe preventive measures such as the use of disease resistant cassava species among other measures. Producing the listed stress conditions would mean artificially inducing the stress conditions. The aforementioned aim of representing real field conditions was also responsible for the absence of weeds such as hemp which are not natural weeds found in the study region.

4.6. Future Work

For future work, we intend to develop techniques for optimizing and reducing the size and complexity of the models so as to successfully deploy them on cheaper non-GPU mobile devices with very little computing power, while still maintaining high model accuracy.

5. Conclusions

In this work, we evaluated two YOLOv5 object detection models for cassava plant detection for real-time applications. We varied the image sizes for the training of the models as well as using different batch sizes during training. For both Y5n and Y5s, training time increases as the image size increases, but at the expense of decreasing accuracy. Generally, Y5s showed higher accuracy than Y5n for the same image size, but with greater training time and higher weight size. However, it was determined that training higher resolution images on Y5n (which has smaller weights file) could yield better mAP@0.5:0.95 than training lower image sizes on Y5s, which is desirable for real-time applications where memory constraints are high. The models were deployed to an NVIDIA® Jetson AGX Orin™ embedded GPU to obtain the real-time inference performance of the models. The models also showed good results for cassava counting, and Y5s showed better detection accuracy in different real-life scenarios than Y5n.

Author Contributions: Conceptualization, E.C.N., O.A.A., O.N.I. and K.Y.; methodology, E.C.N., O.A.A., O.N.I. and K.Y.; data curation, E.C.N.; writing—original draft preparation, E.C.N.; writing—review and editing, E.C.N., O.A.A., O.N.I. and K.Y.; supervision, O.A.A., O.N.I. and K.Y.; funding acquisition, E.C.N., O.A.A., O.N.I. and K.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by TETFUND NRF 2020 with grant number TETF/ES/DR&D-CE/NRF2020/SETI/88/VOL.1 and DAAD In-Country/In-Region Scholarship with personal reference number, 91718448. The APC was funded by Technical University Munich.

Data Availability Statement: The cassava dataset is available from the authors upon request. The data is not publicly available because the dataset is still being expanded and is part of an ongoing funded project and will be published at a later date.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Food and Agriculture Organisation of the United Nations. *The Future of Food and Agriculture–Trends and Challenges*; FAO: Rome, Italy, 2017.
2. Department of Economics and Social Affairs, Population Division. *World Population Prospects 2019*; United Nations: New York, NY, USA, 2019.
3. Duckett, T.; Pearson, S.; Blackmore, S.; Grieve, B.; Chen, W.-H.; Cielniak, G.; Cleaversmith, J.; Dai, J.; Davis, S.; Fox, C.; et al. Agricultural Robotics: The Future of Robotic Agriculture. *arXiv* **2018**, arXiv:1806.06762.
4. Rahman, A.; Lu, Y.; Wang, H. Performance Evaluation of Deep Learning Object Detectors for Weed Detection for Cotton. *Smart Agric. Technol.* **2023**, *3*, 100126. [CrossRef]
5. MacEachern, C.B.; Esau, T.J.; Schumann, A.W.; Hennessy, P.J.; Zaman, Q.U. Detection of Fruit Maturity Stage and Yield Estimation in Wild Blueberry Using Deep Learning Convolutional Neural Networks. *Smart Agric. Technol.* **2023**, *3*, 100099. [CrossRef]
6. Jackulin, C.; Murugavalli, S. A Comprehensive Review on Detection of Plant Disease Using Machine Learning and Deep Learning Approaches. *Meas. Sens.* **2022**, *24*, 100441. [CrossRef]
7. Adair, R.J.; Richard, H.G. *Impact of Environmental Weeds on Biodiversity: A Review and Development of a Methodology*; Biodiversity Group, Environment Australia: Canberra, Australia, 1998.
8. Balasubramanian, D.; Grard, P.; Le Bourgeois, T.; Ramesh, B.R. A Biodiversity Platform for Weed Identification and Knowledge System in the Western Indian Ocean. In Proceedings of the Biodiversity Information Standards (TDWG), Jönköping, Sweden, 27–31 October 2014; pp. 1–3.
9. Podlaski, S.; Chomontowski, C. Various Methods of Assessing Sugar Beet Seed Vigour and Its Impact on the Germination Process, Field Emergence and Sugar Yield. *Sugar Tech* **2020**, *22*, 130–136. [CrossRef]
10. Li, B.; Xu, X.; Han, J.; Zhang, L.; Bian, C.; Jin, L.; Liu, J. The Estimation of Crop Emergence in Potatoes by UAV RGB Imagery. *Plant Methods* **2019**, *15*, 15. [CrossRef]
11. Valente, J.; Sari, B.; Kooistra, L.; Kramer, H.; Mücher, S. Automated Crop Plant Counting from Very High-Resolution Aerial Imagery. *Precis. Agric.* **2020**, *21*, 1366–1384. [CrossRef]

12. Jin, X.; Liu, S.; Baret, F.; Hemerlé, M.; Comar, A. Estimates of Plant Density of Wheat Crops at Emergence from Very Low Altitude UAV Imagery. *Remote Sens. Environ.* **2017**, *198*, 105–114. [CrossRef]
13. Liu, M.; Su, W.-H.; Wang, X.-Q. Quantitative Evaluation of Maize Emergence Using UAV Imagery and Deep Learning. *Remote Sens.* **2023**, *15*, 1979. [CrossRef]
14. Bai, Y.; Nie, C.; Wang, H.; Cheng, M.; Liu, S.; Yu, X.; Shao, M.; Wang, Z.; Wang, S.; Tuohuti, N.; et al. A Fast and Robust Method for Plant Count in Sunflower and Maize at Different Seedling Stages Using High-Resolution UAV RGB Imagery. *Precis. Agric.* **2022**, *23*, 1720–1742. [CrossRef]
15. Vong, C.N.; Conway, L.S.; Zhou, J.; Kitchen, N.R.; Sudduth, K.A. Early Corn Stand Count of Different Cropping Systems Using UAV-Imagery and Deep Learning. *Comput. Electron. Agric.* **2021**, *186*, 106214. [CrossRef]
16. Lu, H.; Cao, Z. TasselNetV2+: A Fast Implementation for High-Throughput Plant Counting from High-Resolution RGB Imagery. *Front. Plant Sci.* **2020**, *11*, 541960. [CrossRef] [PubMed]
17. Ukaegbu, U.F.; Tartibu, L.K.; Okwu, M.O.; Olayode, I.O. Development of a Light-Weight Unmanned Aerial Vehicle for Precision Agriculture. *Sensors* **2021**, *21*, 4417. [CrossRef] [PubMed]
18. Mustafa, M.M.; Hussain, A.; Ghazali, K.H.; Riyadi, S. Implementation of Image Processing Technique in Real Time Vision System for Automatic Weeding Strategy. In Proceedings of the ISSPIT 2007—2007 IEEE International Symposium on Signal Processing and Information Technology, Giza, Egypt, 15–18 December 2007; pp. 632–635.
19. Saha, D.; Hanson, A.; Shin, S.Y. Development of Enhanced Weed Detection System with Adaptive Thresholding and Support Vector Machine. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems—RACS '16*; ACM Press: New York, NY, USA, 2016; pp. 85–88.
20. Barrero, O.; Rojas, D.; Gonzalez, C.; Perdomo, S. Weed Detection in Rice Fields Using Aerial Images and Neural Networks. In Proceedings of the 2016 XXI Symposium on Signal Processing, Images and Artificial Vision (STSIVA), IEEE, Bucaramanga, Colombia, 31 August–2 September 2016; pp. 1–4.
21. Wu, Z.; Chen, Y.; Zhao, B.; Kang, X.; Ding, Y. Review of Weed Detection Methods Based on Computer Vision. *Sensors* **2021**, *21*, 3647. [CrossRef]
22. Thanh Le, V.N.; Truong, G.; Alameh, K. Detecting Weeds from Crops under Complex Field Environments Based on Faster RCNN. In Proceedings of the 2020 IEEE Eighth International Conference on Communications and Electronics (ICCE), Phu Quoc Island, Vietnam, 13–15 January 2021; Bahk, S., Tran-Gia, P., Van der Spiegel, J., Quynh, N.X., Eds.; pp. 350–355.
23. Tang, J.; Wang, D.; Zhang, Z.; He, L.; Xin, J.; Xu, Y. Weed Identification Based on K-Means Feature Learning Combined with Convolutional Neural Network. *Comput. Electron. Agric.* **2017**, *135*, 63–70. [CrossRef]
24. Bah, M.D.; Dericquebourg, E.; Hafiane, A.; Canals, R. Deep Learning Based Classification System for Identifying Weeds Using High-Resolution UAV Imagery. In *Advances in Intelligent Systems and Computing*; Springer: Berlin/Heidelberg, Germany, 2019; Volume 857, pp. 176–187, ISBN 9783030011765.
25. Espejo-Garcia, B.; Mylonas, N.; Athanasakos, L.; Fountas, S.; Vasilakoglou, I. Towards Weeds Identification Assistance through Transfer Learning. *Comput. Electron. Agric.* **2020**, *171*, 105306. [CrossRef]
26. Gao, J.; Liu, C.; Han, J.; Lu, Q.; Wang, H.; Zhang, J.; Bai, X.; Luo, J. Identification Method of Wheat Cultivars by Using a Convolutional Neural Network Combined with Images of Multiple Growth Periods of Wheat. *Symmetry* **2021**, *13*, 2012. [CrossRef]
27. Khan, S.; Tufail, M.; Khan, M.T.; Khan, Z.A.; Anwar, S. Deep Learning-Based Identification System of Weeds and Crops in Strawberry and Pea Fields for a Precision Agriculture Sprayer. *Precis. Agric.* **2021**, *22*, 1711–1727. [CrossRef]
28. Osorio, K.; Puerto, A.; Pedraza, C.; Jamaica, D.; Rodríguez, L. A Deep Learning Approach for Weed Detection in Lettuce Crops Using Multispectral Images. *AgriEngineering* **2020**, *2*, 471–488. [CrossRef]
29. Huang, J.; Rathod, V.; Sun, C.; Zhu, M.; Korattikara, A.; Fathi, A.; Fischer, I.; Wojna, Z.; Song, Y.; Guadarrama, S.; et al. Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. In Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; pp. 3296–3305. [CrossRef]
30. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788. [CrossRef]
31. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef]
32. Czymmek, V.; Harders, L.O.; Knoll, F.J.; Hussmann, S. Vision-Based Deep Learning Approach for Real-Time Detection of Weeds in Organic Farming. In Proceedings of the 2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Auckland, New Zealand, 20–23 May 2019; Volume 2019, pp. 1–5.
33. Gao, J.; French, A.P.; Pound, M.P.; He, Y.; Pridmore, T.P.; Pieters, J.G. Deep Convolutional Neural Networks for Image-Based Convolvulus Sepium Detection in Sugar Beet Fields. *Plant Methods* **2020**, *16*, 29. [CrossRef] [PubMed]
34. Xu, X.; Wang, L.; Shu, M.; Liang, X.; Ghafoor, A.Z.; Liu, Y.; Ma, Y.; Zhu, J. Detection and Counting of Maize Leaves Based on Two-Stage Deep Learning with UAV-Based RGB Image. *Remote Sens.* **2022**, *14*, 5388. [CrossRef]
35. Mota-Delfin, C.; López-Canteñs, G.d.J.; López-Cruz, I.L.; Romantchik-Kriuchkova, E.; Olguín-Rojas, J.C. Detection and Counting of Corn Plants in the Presence of Weeds with Convolutional Neural Networks. *Remote Sens.* **2022**, *14*, 4892. [CrossRef]
36. Food and Agriculture Organisation of the United Nations. *Cassava Diseases in Africa a Major Threat to Food Security*; Strategic programme framework 2010–2015; Food and Agriculture Organisation of the United Nations: Rome, Italy, 2010.

37. Hauser, S.; Wairegi, L.; Asadu, C.L.A.; Asawalam, D.O.; Jokthan, G.; Ugbe, U. *Cassava System Cropping Guide*; Africa Soil Health Consortium: Nairobi, Kenya, 2014.
38. Tzutalin. 2015. *LabelImg (version 1.8.6).* Windows. Git Code.
39. Hertel, L.; Barth, E.; Kaster, T.; Martinetz, T. Deep Convolutional Neural Networks as Generic Feature Extractors. In Proceedings of the International Joint Conference on Neural Networks, Killarney, Ireland, 12–17 July 2015. [CrossRef]
40. Lin, T.-Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C.L.; Dollár, P. Microsoft COCO: Common Objects in Context. In Proceedings of the IEEE Conference on Computer Visual and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3686–3693.
41. Geiß, M.; Wagner, R.; Baresch, M.; Steiner, J.; Zwick, M. Automatic Bounding Box Annotation with Small Training Datasets for Industrial Manufacturing. *Micromachines* **2023**, *14*, 442. [CrossRef] [PubMed]
42. Wang, J.; Xia, B. Weakly Supervised Image Segmentation beyond Tight Bounding Box Annotations. *arXiv* **2023**, arXiv:2301.12053.
43. Deng, C.; Wang, M.; Liu, L.; Liu, Y.; Jiang, Y. Extended Feature Pyramid Network for Small Object Detection. *IEEE Trans. Multimed.* **2022**, *24*, 1968–1979. [CrossRef]
44. Glenn, J. Image Augmentation Functions. Available online: https://github.com/ultralytics/yolov5/blob/6ea81bb3a9bb1701bc0aa9ccca546368ce1fa400/utils/augmentations.py#L279-L284 (accessed on 25 April 2023).
45. Marko, H.; Ljudevit, J.; Gordan, G. A Comparative Study of YOLOv5 Models Performance for Image Localization and Classification. In Proceedings of the Central European Conference on Information and Intelligent Systems, Dubrovnik, Croatia, 20–22 September 2022; pp. 349–356.
46. Ullah, M.B. CPU Based YOLO: A Real Time Object Detection Algorithm. In Proceedings of the 2020 IEEE Region 10 Symposium (TENSYMP), Dhaka, Bangladesh, 5–7 June 2020; pp. 552–555.
47. Lee, J.; Hwang, K. YOLO with Adaptive Frame Control for Real-Time Object Detection Applications. *Multimed. Tools Appl.* **2022**, *81*, 36375–36396. [CrossRef]