agronomy

MDPI

Article

Deep Learning for Detecting and Classifying the Growth Stages of *Consolida regalis* Weeds on Fields

Abeer M. Almalky * and Khaled R. Ahmed *

School of Computing, Southern Illinois University, Carbondale, IL 62901, USA
* Correspondence: abeer.almalky@siu.edu (A.M.A.); khaled.ahmed@siu.edu (K.R.A.)

Abstract: Due to the massive surge in the world population, the agriculture cycle expansion is necessary to accommodate the anticipated demand. However, this expansion is challenged by weed invasion, a detrimental factor for agricultural production and quality. Therefore, an accurate, automatic, low-cost, environment-friendly, and real-time weed detection technique is required to control weeds on fields. Furthermore, automating the weed classification process according to growth stages is crucial for using appropriate weed controlling techniques, which represents a gap of research. The main focus of the undertaken research described in this paper is on providing a feasibility study for the agriculture community using recent deep-learning models to address this gap of research on classification of weed growth stages. For this paper we used a drone to collect a dataset of four weed (*Consolida regalis*) growth stages. In addition, we developed and trained one-stage and two-stage models YOLOv5, RetinaNet (with Resnet-101-FPN, Resnet-50-FPN backbones) and Faster R-CNN (with Resnet-101-DC5, Resnet-101-FPN, Resnet-50-FPN backbones), respectively. The results show that the generated Yolov5-small model succeeds in detecting weeds and classifying weed growth stages in real time with the highest recall of 0.794. RetinaNet with ResNet-101-FPN backbone shows accurate results in the testing phase (average precision of 87.457). Although Yolov5-large showed the highest precision in classifying almost all weed growth stages, Yolov5-large could not detect all objects in tested images. Overall, RetinaNet with ResNet-101-FPN backbones shows accurate and high precision, whereas Yolov5-small shows the shortest inference time in real time for detecting a weed and classifying its growth stages.

Keywords: deep learning; weed detection; weed growth stage detection; YOLOv5; faster R-CNN; RetinaNet; precision farming; machine vision

check for updates

Citation: Almalky, A.M.; Ahmed, K.R. Deep Learning for Detecting and Classifying the Growth Stages of *Consolida regalis* Weeds on Fields. *Agronomy* **2023**, *13*, 934. https://doi.org/10.3390/agronomy13030934

Academic Editors: Salvatore Filippo Di Gennaro and Roberto Marani

Received: 23 January 2023
Revised: 8 March 2023
Accepted: 9 March 2023
Published: 21 March 2023

1. Introduction

By 2050, the world population is projected to increase to 9 billion. As a result of this expansion, the agriculture production is necessarily expected to increase to up to 70% to cope with the world population demand [1]. However, the increase in the agriculture production encounters a considerable number of challenges, such as climate change, pests, and weeds [2]. Swanton et al. define weeds as undesirable plants that compete with crops for water, nutrition, sunlight, and space [3]. Additionally, weeds are severely affecting the quality and value of crops [4]. According to many studies, weeds cause significant economic losses that reach around 33 billion US dollars annually in the United States of America's economy [5] and 11 billion US dollars in India [6]. Since weeds affect the quality of crops, farmers take measures to control the spread of weeds in fields by applying different management techniques including using chemicals [7]. Controlling weeds by spraying herbicides in all fields is an effective solution to reduce the damages caused by weeds. However, herbicides have a harmful influence on the health of people, plants, soil, animals, and pollute the environment in general [8,9]. One study [10] recommended the use of sensitive variable-spraying methods which use specific types of herbicides over the weeds in specific growth stages to minimize the health and pollution problems caused by spraying chemicals all over the fields. Even though selective spraying methods seem to be an efficient weed controlling solution, these methods require a great deal of time and hard manual labor. Automating the process of controlling weeds and selective spraying of chemicals can reduce the required time, labor cost, and the amount of used herbicides [11].

Several studies have been conducted in automating weed detection and classification. These conducted studies used two approaches: image-processing-based techniques [12–18] and a model-based approach. For the model-based approach, two types of techniques are included: machine-learning (ML) techniques and deep-learning (DL) techniques.

Table 1 below summarizes studies [12–18] conducted in detecting weeds using image processing techniques, along with the pre-processing techniques and method/programming language used in each study.

Table 1. Weed-detection-related work using image processing techniques.

Ref.	Pre-Processing	Programming Language/Method
[12]	Color segmentation, edge detection and filtering.	MATLAB
[13]	Selecting YCrCb color space and Cg component, calculating the line in the middle of crop, calculating minimum error ratio of Bayesian decision.	Bayesian decision
[14]	Color segmentation, dividing images into horizontal strips, creating component vector for each strip.	C++
[15]	Edge detection, background subtraction.	MATLAB
[16]	Analyzing differences in spectral reflectance between vegetation and soil, image segmentation using morphological dilation.	Bayes Rule and the k-Nearest Neighbor (KNN)
[17]	Subtracting the green parts in the image, filtering using medium and morphological filters.	MATLAB
[18]	Background removal, RGB images acquisition.	Two suggested equations

Even though the studies utilized image processing techniques for detecting weeds that showed sufficient accuracy, these studies suffered from multiple limitations: (1) frequent adjustment in several parameters, especially a threshold in the image segmentation pre-processing step, (2) high similarity between crop and weed features, (3) manual weed features extraction by experts, (4) computational complexity, and (5) very low accuracy in real-time weed detection.

Because of the improvements in image processing techniques and presentation of low-cost camera devices, many ML models for weed detection and classification have been developed. Table 2 below shows different studies [19–26] which used ML techniques to detect and classify weeds, and presents the extracted features/segmentation method and best result for each study.

Table 2. Weed-detection-related work using ML techniques.

Ref.	Extracted Features/Segmentation Method	Models	Best Result
[19]	Local binary pattern (LBP) segmentation.	Support Vector Machine (SVM)	Not mentioned
[20]	Vegetation indices for color features extraction, gray level co-occurrence matrices (GLCM), Gabor filters.	SVM, Random Forest (RF) Mahalanobis Classifier (MC), KNN	F1_score = 88% by SVM model
[21]	Three spectral bands of Red, Green, Near Infrared (NIR), texture layer.	Supervised Kohonen Network (SKN), Counter-propagation Artificial Neural Network (CP-ANN), XY-Fusion network (XY-F)	Accuracy = 98.87% by CP-ANN model

Table 2. *Cont.*

Ref.	Extracted Features/Segmentation Method	Models	Best Result
[22]	Spectral reflectance.	Mixture of Gaussians (MOG), one-class self-organizing map (SOM) classifiers, one-class SVM	Performance = 31–98% by MOG model
[23]	Linear binary patterns, BRISK, Fourier analysis, watershed.	SVM, linear discriminants KNN, meta-classifier combinations	SVM model showed best result
[24]	Color-based segmentation, watershed segmentation, edge-based segmentation, Hu moments for analyzing leaf shape.	RF	Accuracy = 95% by RF.
[25]	Converting images to orthomosaic images, extracting RGB-band reflectance.	RF, SVM, KNN	Accuracy = 96% by RF.
[26]	185 spectral features (including reflectance and vegetation index features).	RF	Precision = 94% in detecting crops, and 95.9% in detecting weeds.

Despite the noteworthy accuracy showed by these generated models using ML techniques, these studies reported a number of limitations. Similar to the image processing techniques for detecting weeds, ML models of weed detection suffered from the similarity of extracted features between crop and weeds, as well as the manual feature choosing and extraction. In addition, ML models for detecting weeds require large training data and result in false weed detection in the testing phase.

Convolutional Neural Network (CNN) allows the deep learning (DL) techniques to overcome the limitation of manual feature extraction. As a result, DL techniques are able to implement feature extraction and classification in parallel automatically. Generally, DL object detectors can be classified into two categories: one-stage detectors and two-stage detectors [27]. Several conducted studies have used one-stage detectors including You Only Look Once (YOLO) [28] and Single Shot Multibox Detector (SSD) [29] in generating models that detect and classify weeds. These studies showed high accuracy and fast inferencing speed. Additionally, large number of studies have developed trained two-stage detectors such as Faster Region-based Convolutional Neural Network (Faster R-CNN) [30] for detecting and classifying weeds. The reported accuracies achieved by two-stage detectors were high; however, the two-stage detectors suffered from slow detecting speed in weed detection and classification. Table 3 below illustrates studies [31–45] that used DL models for detecting weeds in addition to the best result achieved by each study.

Table 3. Weed-detection-related work using DL techniques.

Ref.	Models	Best Result
[31]	Google-Net, Visual Geometry Group Net (VGG-Net), Detect-Net	F1_score > 0.99 by Detect-Net
[32]	Faster R-CNN, SSD	Faster R-CNN model showed best result
[33]	Google-Net	Accuracy = 86.6%
[34]	YOLOV3	F1_score = 0.95
[35]	YOLOV5	Accuracy = 77%
[36]	YOLOV5-medium, YOLOV3, YOLOV4, Faster R-CNN	Mean Average Precision (mAp) = 87.5 by YOLOV5-medium
[37]	Alex-net, Google-Net, Inception-V3, Xception	Accuracy = 97.7% by Inception-V3
[38]	Faster R-CNN with different anchor box scales and aspect ratios	mAP = 96.02 with the addition of 64 $\times$ 64 scale size, and aspect ratio of 1:3 and 3:1

Table 3. *Cont.*

Ref.	Models	Best Result
[39]	Faster R-CNN with different backbones (Alexnet, VGG16, VGG19, Squeezeenet, Googlenet, Inceptionv3, Mobilenetv2, Resnet18, Resnet50, and Resnet101)	Precision = 0.98, recall = 0.96 by Faster R-CNN with VGG19
[40]	EfficientDet (coefficient 3), YOLOv5	AP for detecting monocot/dicot = 30.70 /51.50% by YOLOv5
[41]	Tested 27 DL models	F1 = 99.1% by ResNet101
[42]	VGG16, ResNet-50, Inception-V3, Inception-ResNet-v2 MobileNetV2	Average precision = 98.29, recall = 98.30 by Resnet-50
[43]	GoogLeNet, MobileNet-v3, ShuffleNet-v2, VGG-Net	Accuracy $\geq$ 0.999, F1 $\geq$ 0.998 by ShuffleNet-v2 and VGG-Net
[44]	Inception-v3	Accuracy = 70%
[45]	Inception-V4, Efficient-NetB7	Accuracy = 70% by Efficient-NetB7

Most of the generated models using DL techniques for detecting weeds suffer from the trade-off between accuracy and speed in real-time detection and detecting small weeds. In fact, there is a lack of large datasets for crops and weeds at various growth stages of crops and weeds [46]. Therefore, the above-mentioned DL algorithms perform poor under different growth stages and actual field backgrounds. Moreover, a very limited number of studies [44,45] have discussed the classification of weed growth stages, which has a big contribution in selective herbicides spraying. Even though the two mentioned studies on detecting weed growth stages [44,45] used Inception-v3, Inception-V4, and Efficient-NetB7, respectively, and achieved 70% accuracy, these studies relied on natural signals (number of leaves and nutrition absorption), which are difficult to detect using DL models.

As an extension of our previous work in [47], this paper includes building and training two more DL models, enabling us to compare and suggest the best DL model for the purpose of weeds detection and their growth stages classification. The main objective of this study, therefore, is to fulfil the gap in the literature by generating a model that is efficient in detecting weeds and classifying weed growth stages in fields and balances the trade-off between accuracy and real-time performance using a large dataset. The main contributions of this paper are summarized as follows: (1) providing a detailed review about the previous works that used computer vision techniques in detecting weeds and classifying weed growth stages; (2) collecting a large weed dataset for *Consolida regalis* weeds; (3) labeling the dataset according to the weed growth stages; (4) evaluating and comparing the performance of three DL models that detect weeds and classify their growth stages.

The rest of the paper is structured as follows: The materials and models are discussed in Section 2. Section 3 summarizes the environment setup and performance evaluation metrics. The results are thoroughly explained in Section 4. Section 5 presents a discussion about the results. Section 6 concludes and explains some further future work.

2. Materials and Methods

The following sections briefly summarize the details of the dataset in this study. In addition, a brief description of YOLOv5 [48], Faster R-CNN [30], and RetinaNet [49], which are used in training our model for automating weed detection and growth stages classification, will be shown.

2.1. Dataset

Even though a number of benchmark weed datasets are available online, these datasets suffer from a limited number of images and unavailability of weed labeling. Moreover, no weed datasets classified according to weed growth stages are publicly available. Therefore, in this work, we collected a weed dataset using a UAV equipped with a digital camera with the features shown in Table 4.

Table 4. UAV camera features and collected dataset properties.

Sensor	1″ CMOS, Effective Pixels: 20 million
Lens	FOV: about 77°, 35 mm Format Equivalent: 28 mm Aperture: f/2.8–f/11, Shooting Range: 1 m to ∞
ISO Range	Video: 100–6400, Photo: 100–3200 (auto), 100–12,800 (manual)
Still Image Size	5472 × 3648
Video Resolution	4K: 3840 × 2160 24/25/30p, 2.7K: 2688 × 1512 24/25/30/48/50/60p FHD: 1920 × 1080 24/25/30/48/50/60/120p
Place	Turkey
Time for Collecting Dataset	One year and six months
Number of Recorded Videos	47 videos
Videos Duration	~5 min/each video
Number of Instances	• Weed 1-1: 63 Instances. • Weed 1-2: 1749 Instances. • Weed 1-3: 3754 Instances. • Weed 1-4: 30,175 Instances.

As stated in Table 4 previously, the UAV was flown in multiple wheat fields in Turkey in different daytimes for one year and six months to capture videos of the weeds in different growth stages. In total, 47 videos with 5 min duration per each video were captured. In addition, some weed seeds were grown in a laboratory to take videos of the weed in its very early growth stages (the first and second growth stages). The resulting videos were divided into frames. Then, we cleaned the frames that did not contain any weed species. The images in the dataset include one type of weed (*Consolida regalis*) classified into four different growth stages and wheat. To label the collected dataset (1) we cooperated with an expert from the weed science department; (2) we used LabelImg [50], an open-source annotation tool, to label our images in .txt format for YOLOv5, and later .txt files converted to COCO.json format for training Faster R-CNN and RetinaNet. The total number of images in the dataset is 3731 with the following number of instances: 63, 1749, 3754, and 30,175 for the first, second, third, and fourth growth stages, respectively. Figure 1a–d show samples of the *Consolida regalis* weed growth stages: the first, second, third, and fourth correspondingly labeled in the ground-truth box in red color.

For training our models, we divided the *Consolida regalis* weed dataset into 2612 images for training, 738 images for validation, and 381 images were used for testing. Additionally, to avoid the confusion between the weed and wheat/soil in the trained models, 1853 images were added as background images.

Since the limited number of images in dataset triggers overfitting [51], adequate amount of data is required for having accurate results in DL models. To avoid overfitting and gain the advantage of regularization, we applied augmentation techniques. To implement augmentation, multiple parameters are popularly used in deep-learning models, such as scaling, color adjustments, rotation, and mosaic augmentation, etc. In YOLOv5, the used augmentation technique is the mosaic augmentation technique which was first released in YOLOv4 and continues to be a primary part in YOLOv5. Mosaic augmentation technique operates by cropping multiple images randomly and combining them into a grid form shown in Figure 2. In addition to increasing the number of images in training dataset, the mosaic augmentation technique has further advantages: (1) improving the model in terms of including the objects (weeds) hidden behind other objects by preserving the relative scale of weed compared to the image; (2) combining different classes (weed growth stages) that might not have been detected altogether in the same image previously; (3) creating variance in the number of weed growth stages in the images, reaching up to 44 weed growth stage instances in one single image. The used augmentation parameters in YOLOv5 are as follows: scale factor of 0.511, shear of 0.0, flip up-down of 0.0, flip left-right of 0.5, mosaic of 0.77768, and translation of 0.07335. Moreover, image resizing was performed in YOLOv5 training by scaling one of the larger images dimensions to 1024, and another dimension was rescaled maintaining the aspect ratio. In Figure 2, the samples of weed growth stages are shown in ground-truth boxes with yellow color.

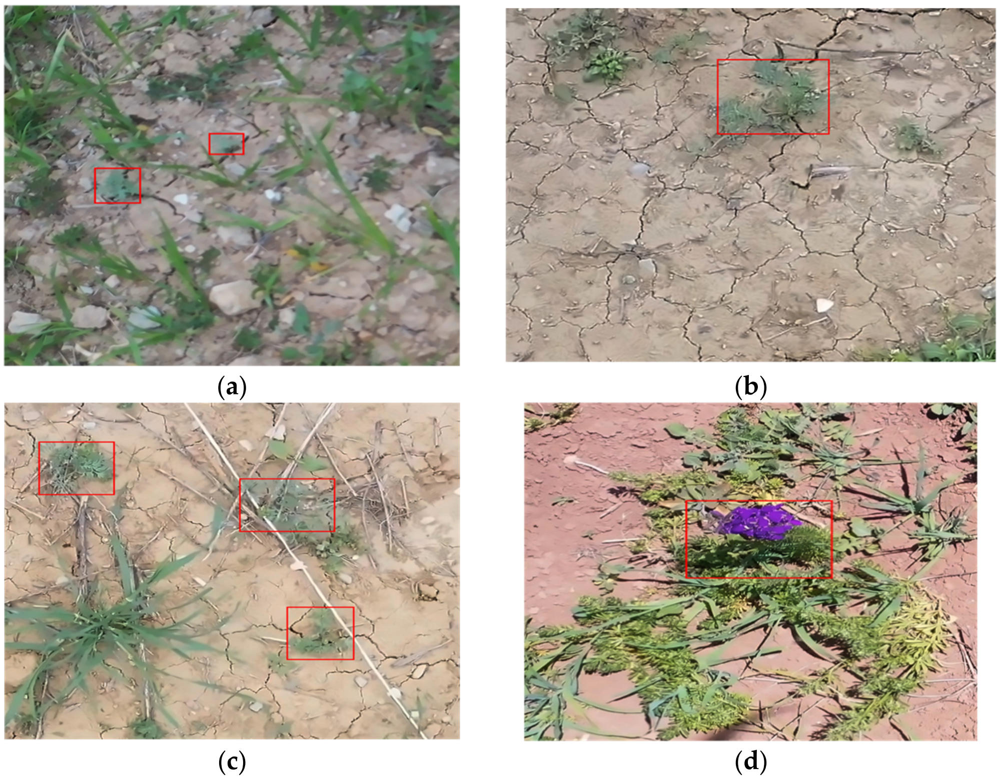


Figure 1. *Consolida regalis* weed growth stages: (**a**) first stage, (**b**) second stage, (**c**) third stage, and (**d**) fourth stage.

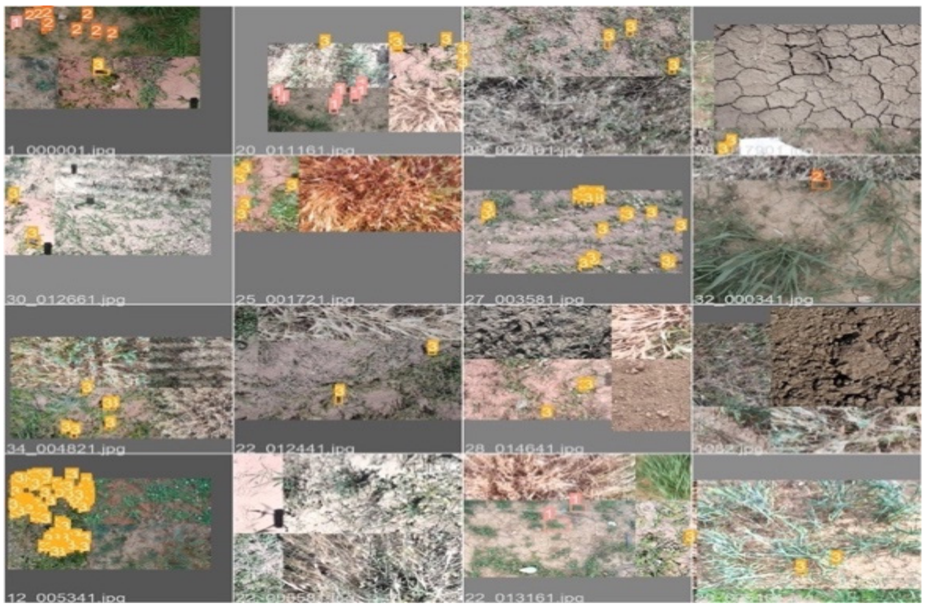


Figure 2. Mosaic YOLOv5.

2.2. *Weed Object Detectors*

Deep-learning object detectors were classified into two categories: one-stage detectors and two-stage detectors [27]. In this paper, we developed both detectors for detecting and classifying the growth stages of *Consolida regalis* weeds.

2.2.1. One-Stage Weed Detectors

In this subsection, a detail explanation about the trained one-stage object detectors (YOLOv5 and RetinaNet) for detecting *Consolida regalis* weed and classification will be presented.

YOLOv5

Redmon et al. [28] proposed a one-stage object detection method named "YOLO". YOLO uses the image pixels to detect the bounding boxes and class probabilities, which makes it faster than other DL approaches that utilize sliding windows, region proposal methods, etc. Generally, detecting objects in YOLO goes through two main steps. First, the algorithm divides the image into S $\times$ S grid cells. Second, if the center of the object falls in one cell, this cell is responsible for detecting this object. Every grid cell predicts B bounding boxes, and each predicted bounding box is associated with confidence score. The predicted bounding boxes consist of five components: (x, y, w, h, c), where (x, y) are the coordinates of the box center, (w, h) are the width and height of the box relating to the whole image, and (c) is the confidence score of the boxes. The confidence score shows the following: (1) how much the model is confident that the predicted box contains the object (weed), (2) how accurate the model thinks the predicted box is the ground-truth box. The confidence score is set to zero if no object (weed) exists in that cell. However, the best confidence score is equal to the intersection over union (IoU) between the predicted box and the ground truth, as shown in Equation (3).

Since the development of official YOLO in 2016, four versions have been released: YOLOv2 [52], YOLOv3 [53], YOLOv4 [54], and YOLOv5 [48]. The version used in this study is YOLOv5 which has four models according to architecture complexity: XS, S, M, and L. In this paper, we provide the performance analysis of YOLOv5 Small (Y_S), Medium (Y_M), and Large (Y_L) models in detecting weeds and classifying their growth stages.

YOLOv5 consists of three main parts [55]: backbone, neck, and head/output part as shown in Figure 3. The backbone is mainly a CNN that is responsible for extracting important features from the given input image using multiple convolutions and pooling. It includes four layers of feature maps that are generated with the following sizes: 152 $\times$ 152, 76 $\times$ 76, 38 $\times$ 38, and 19 $\times$ 19. Neck network combines the four different-size feature maps using three fusion layers in order to acquire more contextual information and lower the loss. Neck network uses two pyramid structures (Feature Pyramid Network (FPN) and Pyramid Attention Network (PAN)) to accomplish the combining step. The first pyramid structure (FPN) carries the semantic features from the top to the lower feature maps. However, the second pyramid structure (PAN) carries the localization features from the lower to the higher feature maps. As a result of neck network, three scales of new feature maps are generated in the output part with the following sizes: 76 $\times$ 76 $\times$ 255, 38 $\times$ 38 $\times$ 255, and 19 $\times$ 19 $\times$ 255. In addition, head/output network uses the three resulting feature maps to generate the final output vectors which include bounding boxes, confidence scores, and class probabilities. ResNet101 was used by YOLOv5 to develop the cross-stage partial (CSP) network [56] that improves the inference speed and maintain the precision by reducing the model size. In YOLOv5, two types of CSP are used: one in backbone network and the other in neck. Both types of CSP are very similar but have a small difference. In the backbone network, the CSP involves one or more residual units, whereas the CSP in neck substitutes the residual units with the CBL modules. The CBL module is composed of convolution modules, normalization modules, and Leaky ReLU activation function. In the backbone network, the SPP module refers to the spatial pyramid pooling that executes the maximum pooling with different kernel size, and fuses the features by concatenating them.

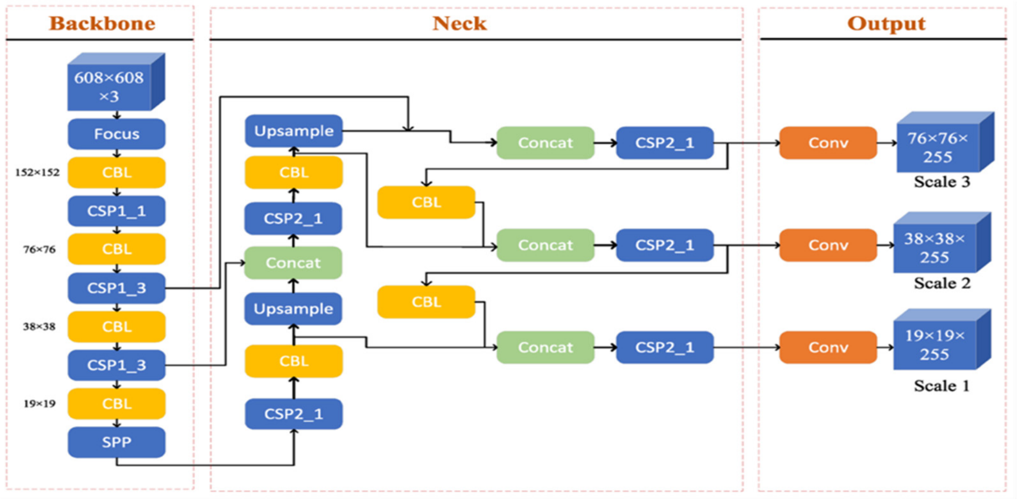


Figure 3. YOLOv5 architecture as displayed in [55].

RetinaNet

RetinaNet [49] is a one-stage object detector, which is mainly a single network that has the following components: backbone network and two task-specific subnetworks. The first component (backbone), usually an FPN [57] over a feedforward ResNet [58], is used for calculating a convolutional feature map over the entire input image, as shown in Figure 4a.

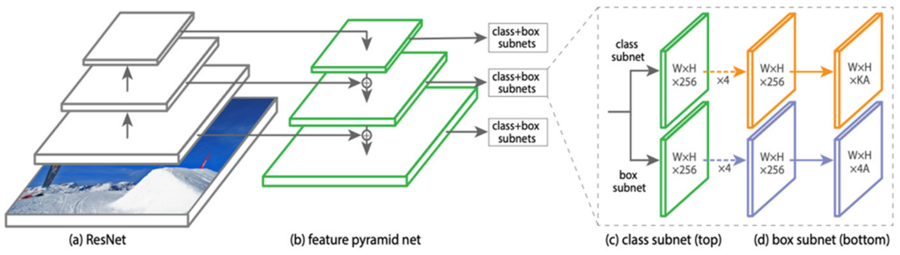


Figure 4. RetinaNet Architecture as displayed in [48]: (**a**) backbone component, (**b**) generated multi-scale convolutional feature pyramid, (**c**) first subnetwork, (**d**) second subnetwork.

The output of the backbone is used by the first subnetwork to perform convolutional object classification, as indicated in Figure 4c. Finally, the second subnetwork executes the convolutional bounding box regression, which cleared in Figure 4d. Figure 4 above illustrates the general RetinaNet structure and its components. In this study, we used the base implementation of RetinaNet with FPN and ResNet backbone to train a model that detects weeds and classifies weed growth stages. We used two types of ResNet in training: ResNet-101 and ResNet-50.

2.2.2. Two-Stage Weed Detectors

This subsection will demonstrate in detail the two-stage object detector (Faster R-CNN) used in this study to develop a model which detects and classifies weed (*Consolida regalis*) and its growth stages.

Faster R-CNN

Ren et al. [30] proposed a two-stage detector named Faster R-CNN, which is widely used in object detection. Faster R-CNN consists of two main modules: Region Proposal Network (RPN) and Fast R-CNN detector [59]. RPN module is responsible for feature extraction by using a fully Convolutional Neural Network (CNN), which proposes rectangular regions along with object-ness score. The CNN in RPN module is also known as backbone networks, such as VGG16 [60], MobileNet [61], ResNet [58], etc. The resulting feature map from the CNN/backbone network is passed into a small n × n sliding window, over which there are two fully connected layers: one for box regression and the other for box classification. In the sliding window, an anchor is placed in the center, and nine anchors are generated by default in each sliding window. Anchor default sizes are as follows: 128, 256, and 512, although these sizes can be overwritten. For each anchor, a label is assigned to classify whether the proposed bounding box contains an object or not. In this study, we trained two different combinations of backbones to generate different Faster R-CNN models. First, we used FPN [57] and ResNet [58]. ResNet-101 and ResNet-50 were used. Second, we used ResNet-101-DC5 backbone that included ResNet-101 with dilations in conv5 and Fully Connected (FC) heads for mask and prediction, respectively.

3. Training and Testing

In this section, a detailed description of the experimental environment and performance evaluation metrics used by trained models will be presented.

3.1. Setup

For training, we used a machine embedded with an Intel Core i5 CPU, GPU of NVIDIA RTX 3090 (24 GB), 32 GB memory, and Windows 10 operating system. Multiple Python3 packages were installed, such as OpenCV [62], PyTorch [63], Cudatoolkit [64], NumPy [65], and Tensorboard [66].

For YOLOv5 models, the momentum and weight decay were 0.9203 and 0.00049, respectively. Anchor size and aspect ratios were calculated dynamically. Stochastic gradient descent (SGD) optimizer and learning rate of 0.0133 were used in training. In addition, the batch size used in Y_L, Y_M, Y_S were 14, 23, and 41, respectively. Moreover, epochs number was 120 in all trained models.

For training Faster R-CNN and RetinaNet, we used Detectron2 [67], an open-source object detection platform established by the Facebook AI research team. We used pre-trained weights that loaded from the model zoo checkpoint in Detectron2. The number of workers was set to 4, and the batch size was also 4. We defined the base learning rate as 0.0003. The warm-up iteration was given a value of 1000 to warm up the base learning rate over 1000 number of SGD iterations. The maximum iterations number (Maximum number of SGD iterations) was specified to 45,000.

3.2. Performance Evaluation Metrics

To evaluate the performance of object detection models and compare the performance of different models trained on the same dataset, different metrics such as precision, accuracy, recall, and mean average precision (mAP) are used. Precision measures the model's accuracy in predicting the weed and weed growth stages, while the accuracy is the ratio of the correct detection to the total number of testing images. Recall is used to measure the model's performance in finding all the weeds in the test images. Intersection over Union (IoU) is a metric used to indicate the amount of the overlap between the predicted and ground-truth bounding boxes. Additionally, IoU affects mostly all the measures in the other metrics. In order to distinguish whether model's detection is valid or invalid, the confidence threshold w (usually w = 0.5) is used. Confidence threshold is mainly used to calculate the ratio of intersection of ground truth and prediction area to the union of ground truth and prediction area. The metrics discussed are formulated by Equations (1)–(6) below:

$$\text{Precision} = \frac{T_P}{\text{All detection}} = \frac{T_P}{T_P + F_P} \tag{1}$$

$$\text{Recall} = \frac{T_P}{\text{All ground truths}} = \frac{T_P}{T_P + F_N} \tag{2}$$

$$\text{IoU} = \frac{A \cup B}{A \cap B} \tag{3}$$

$$\text{Accuracy} = \frac{T_P + T_N}{T_P + T_N + F_P + F_N} \tag{4}$$

$$\text{AP@w} = \int_0^1 p\,(r)dr \tag{5}$$

$$\text{mAP@w} = \frac{1}{N}\sum_{i=1}^{N} \text{AP}_i \tag{6}$$

$$F_\beta - score = \left(1 + \beta^2\right)\frac{\text{Precision} \times \text{Recall}}{(\beta^2 \times \text{Precision}) + \text{Recall}} \tag{7}$$

where A and B stand for prediction bounding box and the ground truth, respectively. T_P and T_N represent the true positive and true negative, accordingly, while F_P and F_N are false positive and false negative. According to the frame-based constrains, two cases are expected: (1) if the bounding box region contains the foreground-object (weed) and IoU $\geq$ w, the frame demonstrates true positive, and (2) if the bounding box region does not contain the foreground-object (weed) and IoU < w, the frame is considered false positive. However, the frame shows false negative when the target object is missed by the bounding box. Object detection models typically show precision-recall (PR) curve, which plots precision as a function of recall to depict the trade-off between the precision and recall for varying confidence values. The area under the PR curve is represented by the average precision (AP@w) as shown in Equation (5). The mean average precision (mAP@w) represents the average of the AP@w calculated for all the classes as shown in Equation (6). F_β-score shows the weighted harmonic average of precision and recall in Equation (7). In this paper we will use F_1-score metric to evaluate the generated models.

4. Results

In this section, the performance metrics of YOLOv5, Faster R-CNN, and RetinaNet models to detect weeds and classify the weed growth stages on training set will be described in detail in Sections 4.1 and 4.2. In addition, the performance of all trained models in testing set will be shown in Section 4.3.

4.1. YOLOv5 Weed Detection and Classification Results

The performance of the three YOLOv5 models (Y_S, Y_M, and Y_L) in detecting the weeds and classifying weed growth stages are shown in Table 5. Y_L model shows the best precision, mean average precision (mAP@.5) and mAP@.5:.95 with 0.827, 0.816, and 0.382, respectively. However, Y_S achieved the best recall of 0.794 in detecting weed growth.

Table 5. Detecting weed growth results using YOLOv5 models.

YOLOv5	Precision	Recall	mAP@.5	mAP@.5:.95	F1	Size in Megabytes	Inference Time (s)	Training Time (h)
Y_s	0.788	**0.794**	0.805	0.363	0.79	**14.342 MB**	**0.0064**	9.598
Y_M	0.809	0.775	0.808	0.351	0.79	41.493 MB	0.0097	10.161
Y_L	**0.827**	0.779	**0.816**	**0.382**	**0.80**	90.957 MB	0.0142	**9.365**

Also, Table 5 shows the real-time performance in weed detection and growth stages classification of the three YOLOv5 models (Y_S, Y_M, and Y_L). As shown, the Y_S is the lightest model with approximately 14.342 MB, while Y_L is the heaviest with 90.957 MB. To assess the performance of the YOLOv5 models, different videos taken from the fields were tested using YOLOv5 detection models. Table 5 shows the inference time tested by the models. Y_S is the fastest model with an inference time of 0.0064 s. However, the Y_L is the slowest model with 0.0142 s inference time. In addition, the training time required by Y_M was the longest, 10.161 h, whereas Y_L needed 9.365 h to be trained in the dataset.

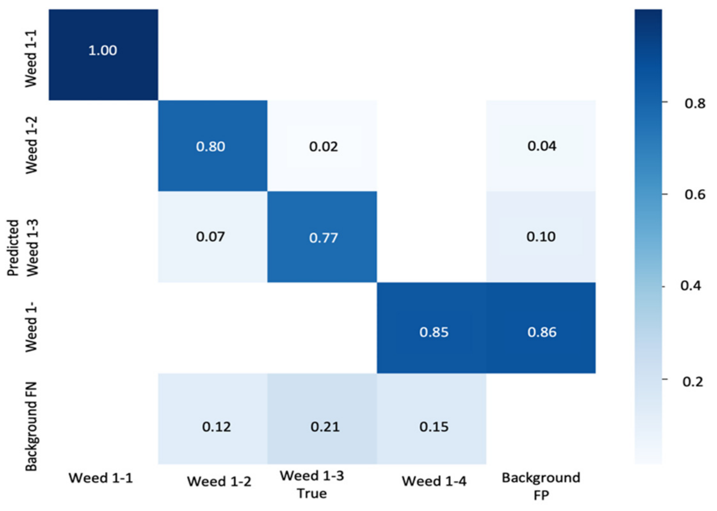
The detailed performance of YOLOv5 models in detecting weed growth stages is shown in Table 6. The names of the classes in Table 6 represent the weed with the number before the dash (*Consolida regalis*) while the number after the dash represents the growth stage: weed 1-1, weed 1-2, weed 1-3, and weed 1-4. In Table 6, Y_L shows the highest precision in detecting weed growth stages with precisions values of 0.966, 0.807, and 0.802 for stage one, two, and four, respectively. However, Y_M shows the highest precision of 0.751 in detecting the third growth stage. All three YOLOv5 models report a recall of 1 in detecting weed in the first growth stage, and that might be due to the low number of weed 1-1 samples in the dataset (64 samples). The Y_S achieves the highest recall of 0.744 and 0.749 in detecting weed in the second and fourth growth stages, respectively, whereas Y_L achieves the best recall of 0.683 in detecting weeds in the third growth stage. The best mAP@.5 is represented by Y_L with 0.995, 0.785, and 0.77 for the first, second, and fourth growth stages, respectively, while Y_S shows the best mAP@.5 of 0.716 for the third growth stage detection. In summary, Y_L shows the best precision in detecting weed and weed growth stages, while Y_S reports the best recall.

Table 6. Detecting weed growth stages using YOLOv5 models.

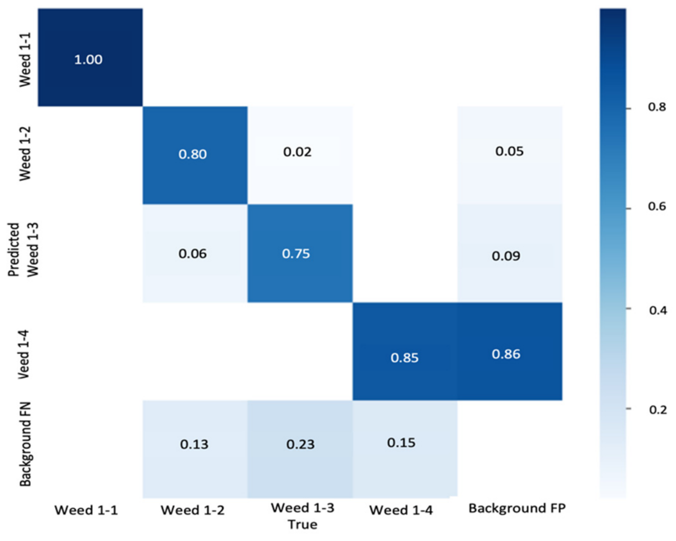
Classes	**Precision**			**Recall**			**mAP@.5**			**mAP@.5:.95**		
	$\mathbf{Y_s}$	$\mathbf{Y_M}$	$\mathbf{Y_L}$	$\mathbf{Y_s}$	$\mathbf{Y_M}$	$\mathbf{Y_L}$	$\mathbf{Y_s}$	$\mathbf{Y_M}$	$\mathbf{Y_L}$	$\mathbf{Y_s}$	$\mathbf{Y_M}$	$\mathbf{Y_L}$
Weed 1-1	0.907	0.914	**0.966**	**1**	**1**	**1**	0.976	**0.995**	**0.995**	0.495	0.443	**0.552**
Weed 1-2	0.773	0.783	**0.807**	**0.744**	0.718	0.702	0.777	0.783	**0.785**	0.373	0.38	**0.386**
Weed 1-3	0.716	**0.751**	0.733	0.682	0.659	**0.683**	**0.716**	0.696	0.715	0.305	0.303	**0.307**
Weed 1-4	0.756	0.789	**0.802**	**0.749**	0.723	0.732	0.752	0.757	**0.77**	0.278	0.277	**0.282**

Figure 5 shows the confusion matrices for the YOLOv5 models. Generally, confusion matrix is utilized to show the performance of classifying the weed growth stages in tested YOLOv5 models. The center diagonal line in the confusion matrix shows the results of prediction giving the weed growth stages classes, whereas the vertical and horizontal lines show background false negative and false positive, respectively. The values in the center line are ranging from 0 to 1, where 1 shows 100% prediction accuracy. As shown in Figure 5, the fourth growth stage of the weed suffers from the background false positive (FP) in all three models (Y_L, Y_M, Y_S), and this confusion might be because most of weed 1-4 samples contain a purple flower. However, some of weed 1-4 samples do not have the purple flower and its color is similar to the wheat and other plants color. Even though 1853 background images were added to the dataset in training, no improvement in the background FP was noticed.

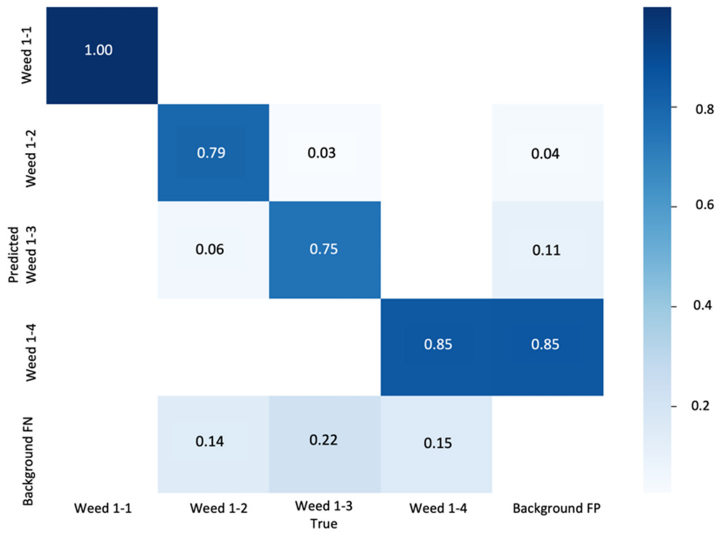
Figure 6 demonstrates the training curves of the trained Y_S, Y_M, and Y_L models. Overall, these YOLOv5 models showed highly efficient training performance in term of classification loss. Y_S was the fastest model in converging toward low classification loss, while Y_M and Y_L were converging at a close rate. At epoch 120, all models stopped training because of low classification loss. Moreover, Figure 7 shows the F1 score curve, which combines the precision and recall in one metric, varying with the confidence threshold score for Y_L model as an example. Figure 7 depicts the performance of Y_L model in terms of classifying the weed growth stages with specific confidence scores. The Y_L model showed the highest F1 score of 0.80 at 0.552 confidence threshold in almost all weed growth stages classifications.



(**a**)



(**b**)



(**c**)

Figure 5. Confusion matrixes for YOLOv5 models: (**a**) Y_L, (**b**) Y_M, and (**c**) Y_S.

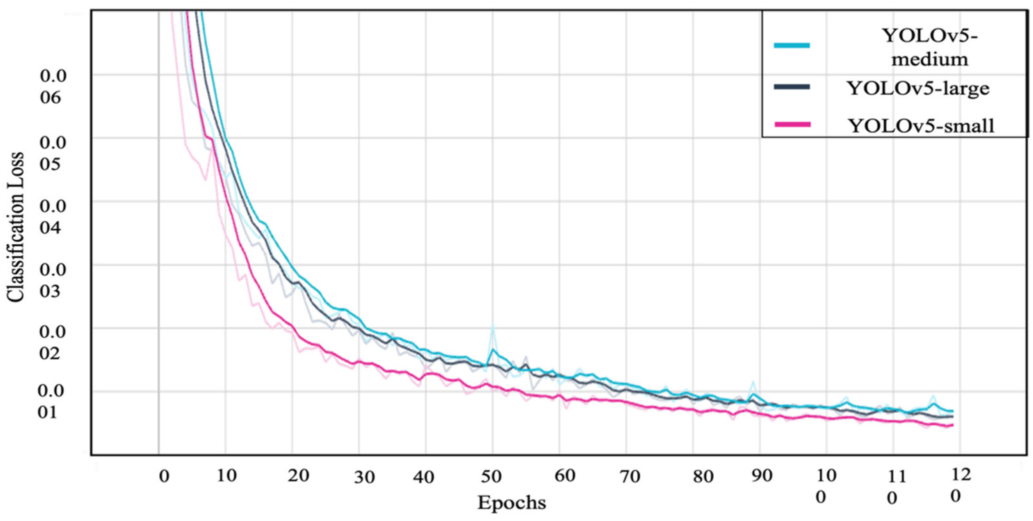


Figure 6. Training and classification loss of YOLOv5 models.

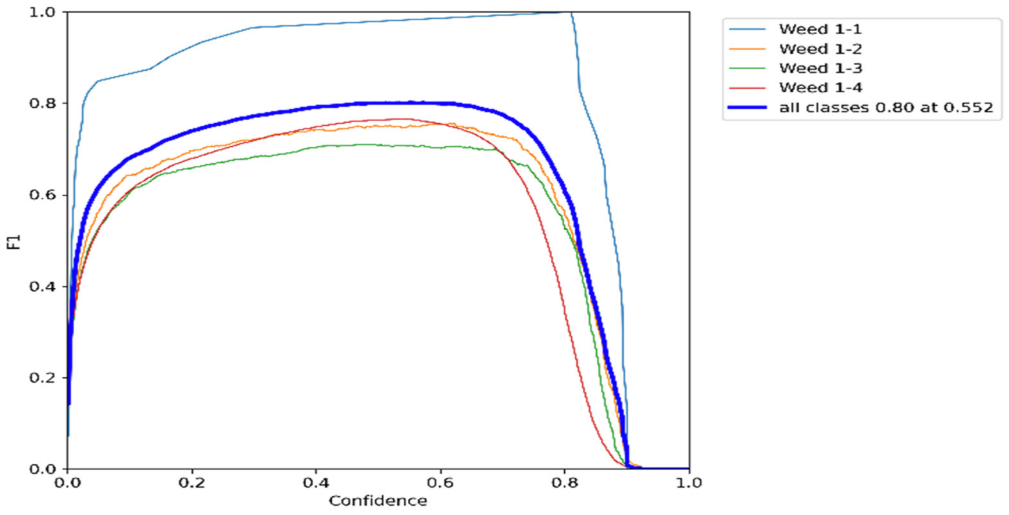


Figure 7. F1 score curve for Y_L model.

4.2. Faster R-CNN and RetinaNet Weed Detection and Classification Results

The performance of Faster R-CNN and RetinaNet, using Detectron2 in training, in detecting weed growth stages is shown in Table 7 below. RetinaNet with ResNet-101 and FPN backbone shows the best average precision (AP@.5), AP@.5:.95, recall of 87.457, 49.991, and 65.5, respectively. Additionally, Table 7 shows Faster R-CNN and RetinaNet performance in real-time weed detection and growth stages classification. As presented, RetinaNet with ResNet-50 and FPN backbone is the lightest model with 283.574 MB, while Faster R-CNN with ResNet-101-DC5 backbone is the heaviest with 1438.6999 MB. For the purpose of evaluating the performance of the models trained in Detecton2, different images, uninvolved in the training process, were tested using both Faster R-CNN and RetinaNet detection models. Furthermore, Table 7 shows the inference time during which the models detected weed growth stage instances in the tested images. Faster R-CNN with ResNet-101and FPN backbone is the fastest model with an inference time of 0.08727 s. However, Faster R-CNN with ResNet-101-DC5 is the slowest model with 0.0924 s inference time.

Table 7. Detecting weed growth stages using Faster R-CNN and RetinaNet.

Model	Backbone	AP@50	AP@.5:.95	Recall	Size in Megabytes	F1	Inference Time (s)	Training Time (h)
Faster R-CNN	R-101-DC5 [1]	86.32	37.464	60.037	1438.699	0.70	0.0924	11:52:57
Faster R-CNN	R-101-FPN [2]	86.763	47.170	62.1	815.742	0.72	**0.08727**	16:18:44
Faster R-CNN	R-50-FPN [3]	84.165	38.039	54.5	322.410	0.66	0.0917	9:11:18
RetinaNet	R-101-FPN [2]	**87.457**	**49.991**	**65.5**	432.056	**0.74**	0.09130	9:19:56
RetinaNet	R-50-FPN [3]	86.404	45.156	62.4	**283.574**	0.72	0.0889	**8:38:46**

[1] ResNet-101 backbone with dilations in conv5; [2] ResNet-101 with FPN backbone; [3] ResNet-50 with FPN backbone.

For the time needed to train models in Detectron2, the training time required by Faster R-CNN with ResNet-101and FPN backbone was the longest: 16.18 h. However, RetinaNet with ResNet-50 and FPN backbone needed 8.38 h to be trained on the dataset.

Table 8 shows the performance of Faster R-CNN and RetinaNet models in detecting weed growth stages. The names of the classes in Table 8 represent the same names of the classes explained for Table 6. All the numbers shown in Table 8 represent the average precision (AP) of each model detecting each weed growth stage. In Table 8, RetinaNet with ResNet-101 and FPN backbone shows the highest performance in detecting weed growth stages with average precision values of 55.149, 55.809, and 31.910 for stage one, three, and four, respectively. However, Faster R-CNN with ResNet-101 and FPN backbone shows the highest average precision of 58.817 in detecting the second growth stage.

Table 8. Detecting weed growth stages using Faster R-CNN and RetinaNet (Average Precision).

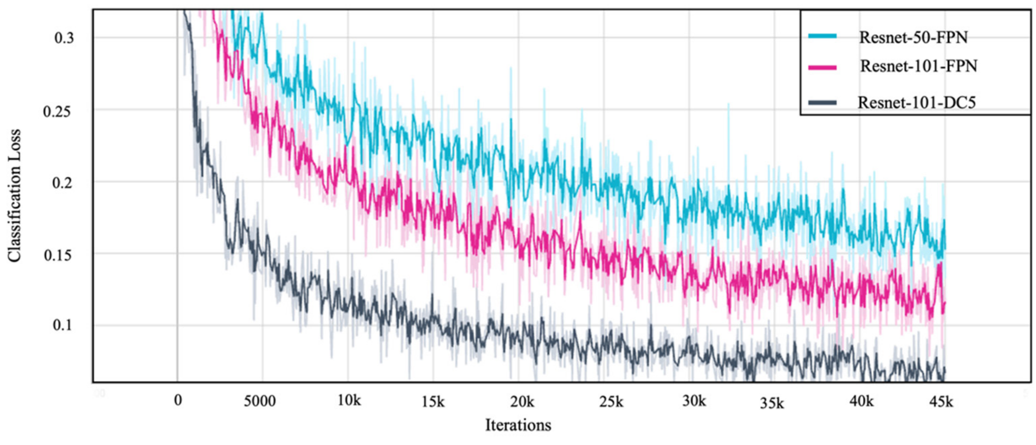
Model	Backbone	Weed 1-1	Weed 1-2	Weed 1-3	Weed 1-4
Faster R-CNN	R-101-DC5	42.624	58.022	47.045	30.216
Faster R-CNN	R-101-FPN	50.099	**58.817**	48.419	31.346
Faster R-CNN	R-50-FPN	35.050	52.184	38.732	26.192
RetinaNet	R-101-FPN	**55.149**	57.098	**55.809**	**31.910**
RetinaNet	R-50-FPN	45.149	55.730	50.702	29.043

Figure 8a displays the training curves of all trained models of Faster R-CNN with three different backbones. Faster R-CNN models showed acceptable training performance according to the classification loss. Faster R-CNN with Resnet-101-DC5 backbone was the fastest model in convergence toward low classification loss while the other Faster R-CNN models with Resnet-101-FPN and Resnet-50-FPN backbones were converging at a close rate. All Faster R-CNN models stopped training at 45k iteration. Figure 8b demonstrates the training curves for RetinaNet models with two backbones: Resnet-101-FPN and Res-net-50-FPN. All RetinaNet models showed closely similar performance in training with respect to classification loss. Similar to Faster R-CNN models, all RetinaNet models stopped training at 45k iteration.

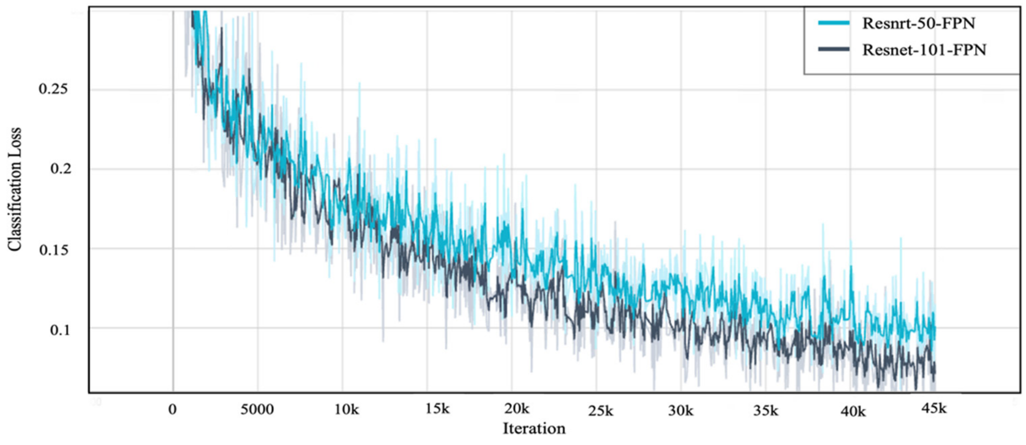
4.3. Weed Detection and Growth Stages Classification Results

To show the performance of all three trained DL models in detecting weeds and classifying weed growth stages in unseen images, we tested all the three models in testing set, which is not included in the training of the models. As shown in Figure 9 below, we tested an image with two instances of weed 1-2 and fourteen instances of weed 1-3, as shown in Figure 9a, by all the trained models in this study. As shown in Figure 9, only trained models Faster R-CNN and RetinaNet with different backbones were able to detect all the weed instances in the image except one instance of weed 1-3, as shown in Figure 9b–f.

However, Y_s, Y_M, Y_L models succeeded in detecting 62.5%, 37.5%, and 25% of the weed instances in the tested image, respectively.

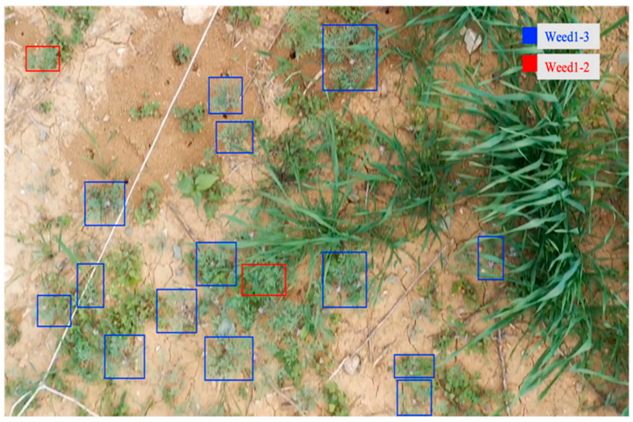


(**a**)



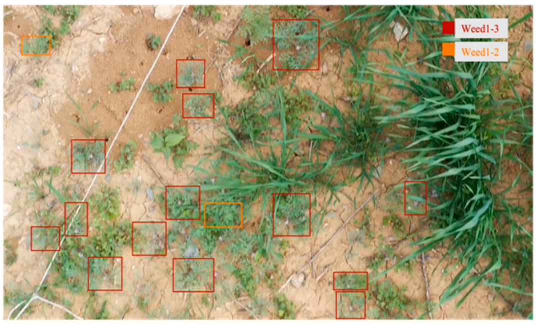
(**b**)

Figure 8. Training and classification loss of (**a**) Faster R-CNN models and (**b**) RetinaNet models.

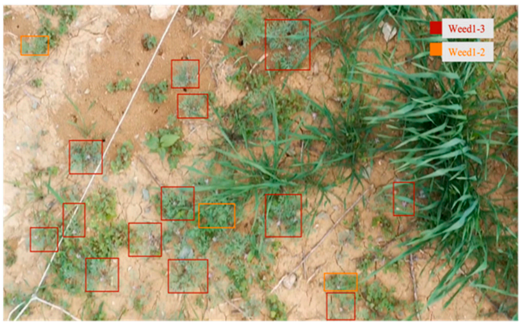


(**a**) Ground Truth Labels

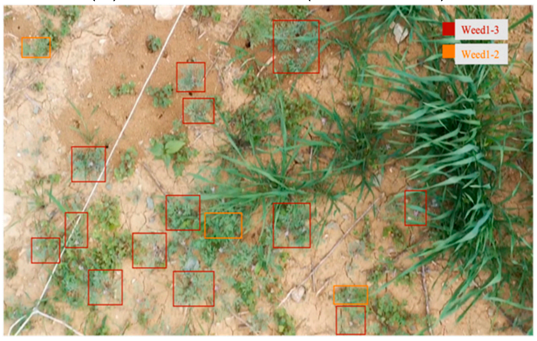
Figure 9. *Cont*.



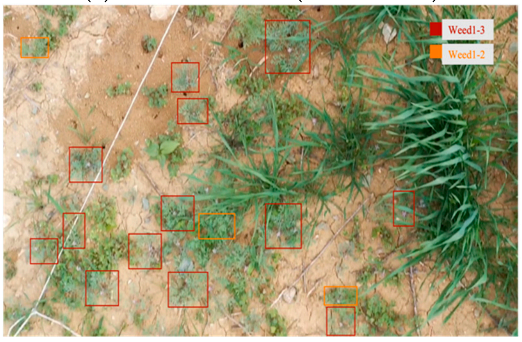
(**b**) Faster R-CNN (R-101-DC5)



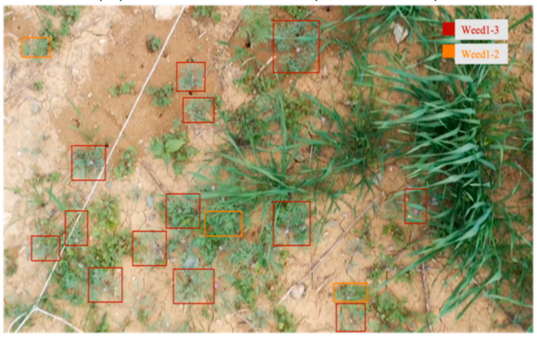
(**c**) Faster R-CNN (R-101-FPN)



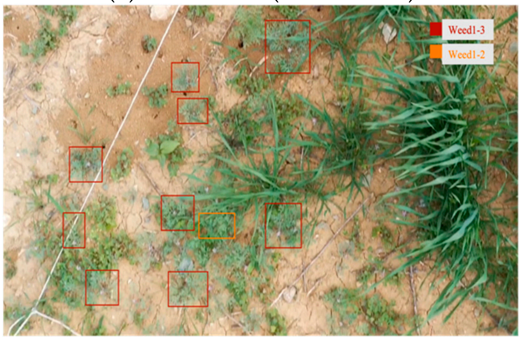
(**d**) Faster R-CNN (R-50-FPN)



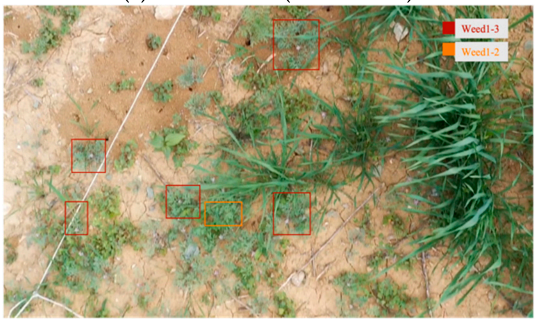
(**e**) RetinaNet (R-101-FPN)



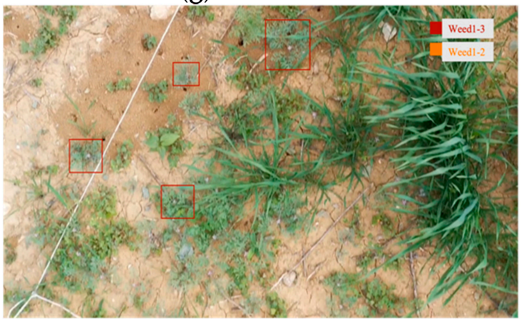
(**f**) RetinaNet (R-50-FPN)



(**g**) Yolov5 -Y_S



(**h**) Yolov5 -Y_M



(**i**) Yolov5 -Y_L

Figure 9. Sample image from detection using (**a**) Ground Truth Labels, (**b**) Faster R-CNN (R-101-DC5), (**c**) Faster R-CNN (R-101-FPN), (**d**) Faster R-CNN (R-50-FPN), (**e**) RetinaNet (R-101-FPN), (**f**) RetinaNet (R-50-FPN), (**g**) Y_S, (**h**)Y_M, and (**i**)Y_L.

5. Discussion

In this study, two DL models of one-stage object detractor (YOLOv5 and RetinaNet) and one DL model of two-stage object detector (Faster R-CNN) were trained for detecting *Consolida regalis* weed and classifying its four growth stages. According to the results in the previous section, RetinaNet model with ResNet-101-FPN backbone showed the highest average precision of 87.457 in detecting weed comparing to all trained models. Likewise, Faster R-CNN with ResNet-101-FPN backbone showed an adequate average precision of

86.763 in detecting weed. With regards to YOLOv5 models, Y_L showed the best precision of 0.827 in detecting weed, but not better than that of RetinaNet and Faster R-CNN models mentioned above.

In terms of classifying the *Consolida regalis* weed growth stages, Y_L showed the highest precision in classifying almost all the weed growth stages. However, the low number of samples of the first growth stage results in a classification precision of 1 in all YOLOv5 models.

The comparison of the real time performance in detecting weed and classifying its growth stages shows that Y_S presented the highest recall and the lowest inference time of 0.0064 s. However, when Faster R-CNN models are compared with the other trained models, YOLOv5 and RetinaNet, all Faster R-CNN models show adequate recall and F1 score results.

In conclusion, we recommend using RetinaNet with ResNet-101-FPN model since it achieves the trade-off between the cost (inference time) and the benefits (accuracy) of detecting and classifying weed growth stages.

6. Conclusions

Automating weed detection and weed growth stages classification is important for protecting and supporting the agriculture production and the expected agricultural expansion, considering an increase in demand. This paper provided a feasibility study for the agriculture community by developing recent DL models with different backbones to address the problem of the automation of weed growth stages classification. As a result, we developed three different deep-learning models with different backbones for automatic detection of one type of *Consolida regalis* weeds and classification of its four growth stages. In this study, we collected a weed dataset in different times of the year to capture the weed in different growth stages. The weed dataset was trained using YOLOv5 models (Y_S, Y_M, Y_L), Faster R-CNN with three different combinations of backbones, and RetinaNet with two types of backbones. The experiment results show that RetinaNet with ResNet-101-FPN achieved the highest average precision in detecting weed, while Y_L showed the highest precision in classifying the growth stages. However, Y_S achieved the highest recall in detection and classification. The developed algorithm presented in this paper succeeded in the classification of the growth stages of *Consolida regalis*, which could be considered as a support tool when conducting field-based weed control. Finally, the recommended model for detecting the growth stages of *Consolida regalis* weeds is RetinaNet with ResNet-101-FPN since it achieves the trade-off between the accuracy and real-time performance.

In our future research, we will collect more data on different weed species and more growth stages. The accuracy in classifying weed growth stages can be further improved by using training datasets with a balanced number of samples for all growth stages. Moreover, in the perspective of quantification of weed amount in different growth stages, we are planning to develop deep-learning models for the instance segmentation task. Finally, we plan to experimentally investigate the proposed recommendations for developing a complete weed detection and control system to identify the exact weed location and growth stages of different weed species using DL, and then remove or kill weeds with robotic arms or lasers, respectively, under real-world conditions.

Author Contributions: All authors contribute evenly to conceptualization, methodology, software, validation, and writing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy.

Acknowledgments: The Article is a revised and expanded version of the paper entitled "An Efficient Deep Learning Technique for Detecting and Classifying the Growth of Weeds on Fields" which was presented at Future Technologies Conference 2023, Vancouver, Canada on 20–21 October 2022. Also, we would like to thank Mustafa Guzel for providing the weed dataset and for his generous help and guidance in the labeling task.

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

References

1. Sylvester, G. *E-Agriculture in Action: Drones for Agriculture*; Food and Agriculture Organization of the United Nations: Bangkok, Thailand, 2018.
2. Lee, W.S.; Alchanatis, V.; Yang, C.; Hirafuji, M.; Moshou, D.; Li, C. Sensing Technologies for Precision Specialty Crop Production. *Comp. Elect. Agric.* **2010**, *74*, 2–33. [CrossRef]
3. Swanton, C.J.; Nkoa, R.; Blackshaw, R.E. Experimental Methods for Crop–Weed Competition Studies. *Weed Sci.* **2015**, *63*, 2–11. [CrossRef]
4. Patel, D.D.; Kumbhar, B.A. Weed and Its Management: A Major Threats to Crop Economy. *J. Pharm. Sci. Biosci. Res.* **2016**, *6*, 753–758.
5. Pimentel, D.; Zuniga, R.; Morrison, D. Update on the Environmental and Economic Costs Associated with Alien-Invasive Species in the United States. *Ecol. Econ.* **2005**, *52*, 273–288. [CrossRef]
6. Ghardea, Y.; Singha, P.K.; Dubeya, R.P.; Gupta, P.K. Assessment of yield and economic losses in agriculture due to weeds in India. *Sci. Direct* **2018**, *107*, 12–18. [CrossRef]
7. Methods of Weed Control. Available online: https://www.larimer.gov/naturalresources/weeds/control (accessed on 25 September 2022).
8. Holt, J.S. Principles of Weed Management in Agroecosystems and Wildlands. *Weed Technol.* **2004**, *18*, 1559–1562. [CrossRef]
9. Llewellyn, R.; Ronning, D.; Clarke, M.; Mayfield, A.; Walker, S.; Ouzman, J. *Impact of Weeds on Australian Grain Production—The Cost of Weeds to Australian Grain Growers and the Adoption of Weed Management and Tillage Practices*; Grains Research and Development Corporation and the Commonwealth Scientific and Industrial Research Organisation: Canberra, Australia, 2016.
10. Bàrberi, P. Weed Management in Organic Agriculture: Are We Addressing the Right Issues? *Weed Res.* **2002**, *42*, 177–193. [CrossRef]
11. Lameski, P.; Zdravevski, E.; Kulakov, A. A Short Review of the Environmental Impact of Automated Weed Control. In *ICT Innovations 2018*; Engineering and Life Sciences; Springer: Cham, Switzerland, 2018; pp. 132–147.
12. Paikekari, A.; Ghule, V.; Meshram, R.; Raskar, V. Weed Detection Using Image Processing. *Inter. Res. J. Eng. Technol.* **2016**, *3*, 1220–1222.
13. Tang, J.-L.; Chen, X.-Q.; Miao, R.-H.; Wang, D. Weed Detection Using Image Processing under Different Illumination for Site-Specific Areas Spraying. *Comput. Elect. Agric.* **2016**, *122*, 103–111. [CrossRef]
14. Burgos-Artizzu, X.P.; Ribeiro, A.; Guijarro, M.; Pajares, G. Real-Time Image Processing for Crop/Weed Discrimination in Maize Fields. *Comput. Elect. Agric.* **2011**, *75*, 337–346. [CrossRef]
15. Hameed, S.; Amin, I. Detection of Weed and Wheat Using Image Processing. In Proceedings of the 2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS), Bangkok, Thailand, 22–23 November 2018; pp. 1–5. [CrossRef]
16. Pérez, A.J.; López, F.; Benlloch, J.V.; Christensen, S. Colour and Shape Analysis Techniques for Weed Detection in Cereal Fields. *Comput. Elect. Agric.* **2000**, *25*, 197–212. [CrossRef]
17. Tejeda, A.J.I.; Castro, R.C. Algorithm of Weed Detection in Crops by Computational Vision. In Proceedings of the 2019 International Conference on Electronics, Communications and Computers (CONIELECOMP), IEEE, Cholula, Mexico, 27 February–1 March 2019; pp. 124–128. [CrossRef]
18. Parra, L.; Torices, V.; Marín, J.; Mauri, P.V.; Lloret, J. The Use of Image Processing Techniques for Detection of Weed in Lawns. In Proceedings of the Fourteenth International Conference on Systems (ICONS 2019), Valencia, Spain, 24–28 March 2019; pp. 24–28.
19. Bini, D.; Pamela, D.; Prince, S. Machine vision and machine learning for intelligent agrobots: A review. In Proceedings of the 2020 5th International Conference on Devices, Circuits and Systems (ICDCS), Coimbatore, India, 5–6 March 2020; pp. 12–16.
20. César Pereira Júnior, P.; Monteiro, A.; Da Luz Ribeiro, R.; Sobieranski, A.C.; Von Wangenheim, A. Comparison of Supervised Classifiers and Image Features for Crop Rows Segmentation on Aerial Images. *Appl. Artif. Intell.* **2020**, *34*, 271–291. [CrossRef]
21. Pantazi, X.E.; Tamouridou, A.A.; Alexandridis, T.K.; Lagopodi, A.L.; Kashefi, J.; Moshou, D. Evaluation of Hierarchical Self-Organising Maps for Weed Mapping Using UAS Multispectral Imagery. *Comput. Elect. Agric.* **2017**, *139*, 224–230. [CrossRef]
22. Pantazi, X.-E.; Moshou, D.; Bravo, C. Active Learning System for Weed Species Recognition Based on Hyperspectral Sensing. *Biosyst. Engin.* **2016**, *146*, 193–202. [CrossRef]
23. Binch, A.; Fox, C.W. Controlled Comparison of Machine Vision Algorithms for Rumex and Urtica Detection in Grassland. *Comput. Elect. Agric.* **2017**, *140*, 123–138. [CrossRef]
24. Alam, M.; Alam, M.S.; Roman, M.; Tufail, M.; Khan, M.U.; Khan, M.T. Real-Time Machine-Learning Based Crop/Weed Detection and Classification for Variable-Rate Spraying in Precision Agriculture. In Proceedings of the 2020 7th International Conference on Electrical and Electronics Engineering (ICEEE), IEEE, Antalya, Turkey, 14–16 April 2020; pp. 273–280. [CrossRef]
25. Islam, N.; Rashid, M.M.; Wibowo, S.; Xu, C.-Y.; Morshed, A.; Wasimi, S.A.; Moore, S.; Rahman, S.M. Early Weed Detection Using Image Processing and Machine Learning Techniques in an Australian Chilli Farm. *Agriculture* **2021**, *11*, 387. [CrossRef]

26. Gao, J.; Nuyttens, D.; Lootens, P.; He, Y.; Pieters, J.G. Recognizing Weeds in a Maize Crop Using a Random Forest Machine-Learning Algorithm and near-Infrared Snapshot Mosaic Hyperspectral Imagery. *Biosyst. Eng.* **2018**, *170*, 39–50. [CrossRef]
27. Liu, L.; Ouyang, W.; Wang, X.; Fieguth, P.; Chen, J.; Liu, X.; Pietikäinen, M. Deep Learning for Generic Object Detection: A Survey. *Int. J. Comput. Vis.* **2020**, *128*, 261–318. [CrossRef]
28. 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), IEEE, Los Alamitos, CA, USA, 27–30 June 2016; pp. 779–788. [CrossRef]
29. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Proceedings of the Computer Vision—ECCV 2016, Amsterdam, The Netherlands, 11–14 October 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 21–37. [CrossRef]
30. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Neural Netw.* **2017**, *39*, 1137–1149. [CrossRef]
31. Yu, J.; Sharpe, S.M.; Schumann, A.W.; Boyd, N.S. Deep Learning for Image-Based Weed Detection in Turfgrass. *Eur. J. Agron.* **2019**, *104*, 78–84. [CrossRef]
32. Veeranampalayam Sivakumar, A.N.; Li, J.; Scott, S.; Psota, E.; Jhala, J.A.; Luck, J.D.; Shi, Y. Comparison of Object Detection and Patch-Based Classification Deep Learning Models on Mid- to Late-Season Weed Detection in UAV Imagery. *Remote Sens.* **2020**, *12*, 2136. [CrossRef]
33. Dyrmann, M.; Jørgensen, R.N.; Midtiby, H.S. RoboWeedSupport—Detection of Weed Locations in Leaf Occluded Cereal Crops Using a Fully Convolutional Neural Network. *Adv. Anim. Biosci.* **2017**, *8*, 842–847. [CrossRef]
34. Sharpe, S.M.; Schumann, A.W.; Yu, J.; Boyd, N.S. Vegetation Detection and Discrimination within Vegetable Plasticulture Row-Middles Using a Convolutional Neural Network. *Precis. Agric.* **2020**, *21*, 264–277. [CrossRef]
35. Junior, L.C.M.; Ulson, J.A.C. Real Time Weed Detection Using Computer Vision and Deep Learning. In Proceedings of the 2021 14th IEEE International Conference on Industry Applications (INDUSCON), IEEE, São Paulo, Brazil, 16–18 August 2021; pp. 1131–1137. [CrossRef]
36. Salazar-Gomez, A.; Darbyshire, M.; Gao, J.; Sklar, E.I.; Parsons, S. Towards Practical Object Detection for Weed Spraying in Precision Agriculture. *arXiv* **2021**. [CrossRef]
37. Subeesh, A.; Bhole, S.; Singh, K.; Chandel, N.S.; Rajwade, Y.A.; Rao, K.V.R.; Kumar, S.P.; Jat, D. Deep Convolutional Neural Network Models for Weed Detection in Polyhouse Grown Bell Peppers. *Artif. Intell. Agric.* **2022**, *6*, 47–54. [CrossRef]
38. Saleem, M.H.; Potgieter, J.; Arif, K.M. Weed Detection by Faster RCNN Model: An Enhanced Anchor Box Approach. *Agronomy* **2022**, *12*, 1580. [CrossRef]
39. Quan, L.; Feng, H.; Lv, Y.; Wang, Q.; Zhang, C.; Liu, J.; Yuan, Z. Maize Seedling Detection under Different Growth Stages and Complex Field Environments Based on an Improved Faster R-CNN. *Biosyst. Eng.* **2019**, *184*, 1–23. [CrossRef]
40. Teimouri, N.; Jørgensen, R.N.; Green, O. Novel Assessment of Region-Based CNNs for Detecting Monocot/Dicot Weeds in Dense Field Environments. *Agronomy* **2022**, *12*, 1167. [CrossRef]
41. Chen, D.; Lu, Y.; Li, Z.; Young, S. Performance Evaluation of Deep Transfer Learning on Multi-Class Identification of Common Weed Species in Cotton Production Systems. *Comput. Elect. Agric.* **2022**, *198*, 107091. [CrossRef]
42. Mahmudul Hasan, A.S.M.; Sohel, F.; Diepeveen, D.; Laga, H.; Jones, M.G.K. Weed Recognition Using Deep Learning Techniques on Class-Imbalanced Imagery. *Crop Pasture Sci.* **2022**, A–Q. [CrossRef]
43. Jin, X.; Bagavathiannan, M.; Maity, A.; Chen, Y.; Yu, J. Deep Learning for Detecting Herbicide Weed Control Spectrum in Turfgrass. *Plant Methods* **2022**, *18*, 94. [CrossRef] [PubMed]
44. Teimouri, N.; Dyrmann, M.; Nielsen, P.R.; Mathiassen, S.K.; Somerville, G.J.; Jørgensen, R.N. Weed Growth Stage Estimator Using Deep Convolutional Neural Networks. *Sensors* **2018**, *18*, 1580. [CrossRef]
45. Mishra, A.M.; Harnal, S.; Mohiuddin, K.; Gautam, V.; Nasr, O.A.; Goyal, N.; Alwetaishi, M.; Singh, A. A Deep Learning-Based Novel Approach for Weed Growth Estimation. *Intell. Autom. Soft Comput.* **2022**, *31*, 1157–1173. [CrossRef]
46. Hasan, A.S.M.M.; Sohel, F.; Diepeveen, D.; Laga, H.; Jones, M.G.K. A Survey of Deep Learning Techniques for Weed Detection from Images. *Comput. Elect. Agric.* **2021**, *184*, 106067. [CrossRef]
47. Almalky, A.M.; Ahmed, K.R.; Guzel, M.; Turan, B. An Efficient Deep Learning Technique for Detecting and Classifying the Growth of Weeds on Fields. In *Proceedings of the Future Technologies Conference (FTC) 2022, Volume 2*; Arai, K., Ed.; Lecture Notes in Networks and Systems; Springer International Publishing: Cham, Switzerland, 2023; pp. 818–835. [CrossRef]
48. Jocher, G. Yolov5. Available online: https://github.com/ultralytics/yolov5 (accessed on 25 March 2022).
49. Lin, T.; Goyal, P.; Girshick, R.; He, K.; Dollar, P. Focal Loss for Dense Object Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 318–327. [CrossRef]
50. Lin, T. LabelImg. Available online: https://github.com/tzutalin/labelImg (accessed on 20 April 2022).
51. Chen, X.-W.; Lin, X. Big Data Deep Learning: Challenges and Perspectives. *IEEE Access* **2014**, *2*, 514–525. [CrossRef]
52. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525. [CrossRef]
53. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**. [CrossRef]
54. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**. [CrossRef]

55. Zhu, L.; Geng, X.; Li, Z.; Liu, C. Improving YOLOv5 with Attention Mechanism for Detecting Boulders from Planetary Images. *Remote Sens.* **2021**, *13*, 3776. [CrossRef]
56. Wang, C.-Y.; Mark Liao, H.-Y.; Wu, Y.-H.; Chen, P.-Y.; Hsieh, J.-W.; Yeh, I.-H. CSPNet: A New Backbone That Can Enhance Learning Capability of CNN. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE, Seattle, WA, USA, 14–19 June 2020; pp. 1571–1580. [CrossRef]
57. Lin, T.-Y.; Dollar, 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), IEEE, Honolulu, HI, USA, 21–26 July 2017; pp. 936–944. [CrossRef]
58. 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), IEEE, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
59. Girshick, R. Fast R-CNN. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), IEEE, Santiago, Chile, 11–18 December 2015; pp. 1440–1448. [CrossRef]
60. Liu, S.; Deng, W. Very Deep Convolutional Neural Network Based Image Classification Using Small Training Sample Size. In Proceedings of the 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), IEEE, Kuala Lumpur, Malaysia, 3–6 November 2015; pp. 730–734. [CrossRef]
61. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**. [CrossRef]
62. Bradski, G. The OpenCV Library. *Dr. Dobb's J. Softw. Tools Prof. Program* **2000**, *25*, 120–123.
63. Ketkar, N. *Deep Learning with Python: A Hands-on Introduction*, 3rd ed.; Apress: New York, NY, USA, 2017; pp. 195–208, ISBN 978-1-4842-2765-7.
64. Kirk, D. NVIDIA Cuda Software and Gpu Parallel Computing Architecture. In Proceedings of the 6th International Symposium on Memory Management, Montreal, QC, Canada, 21–22 October 2007; Association for Computing Machinery: New York, NY, USA, 2007; pp. 103–104. [CrossRef]
65. Oliphant, T.E. *Guide to NumPy*, 2nd ed.; CreateSpace Independent Publishing Platform: Scotts Valley, CA, USA, 2015.
66. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv* **2016**. [CrossRef]
67. Wu, Y.; Kirillov, A.; Massa, F.; Lo, W.-Y.; Girshick, R. Detectron2. 2019. Available online: https://github.com/facebookresearch/detectron2 (accessed on 14 September 2022).