University of Adelaide Biometry Hub Technical Report Series: SS001

Applying the YOLO object detection framework to identifying seedling emergence in aerial farm images

– Report for the Data Science research project (part B) –

Tuan Minh Nguyen
email: a1847275@adelaide.edu.au

November 24, 2023

Contents

Abstract

To support the detection of early emergence of plants in the field of agronomy, sophisticated technologies such as computer vision and image processing have been widely implemented to make use of aerial photographs captured of crop fields. Many applications have been developed to harness the power of these innovations, but traditional approaches like the use of Convolutional Neural Networks (CNNs) for image classification and the sliding window techniques often hinder the practicality of those toolsets due to inaccurate outputs and elongated processing time. With a core component of enhancing the workflow currently executed by the Biometry Hub at the University of Adelaide for detecting and counting early emergence of faba beans, this paper experiments multiple versions of a specialised object detection framework called You Only Look Once (YOLO) to determine the most effective model, which will be proposed for deployment.

After five candidate YOLO models were trained and validated, the eighth YOLO version (YOLOv8) was nominated with a validation Mean Average Precision (mAP) of 95.9%. The chosen model has resolved all the major issues of the old algorithm, including the huge runtime and computational demands, as well as inaccurate counting results caused by clumping plants. The final evaluation results also showed an outstanding performance of this model with a testing mAP of 94.6% and it only miscounted around 6 plants on average in full-size plot images, reducing the plant-counting errors by 76.9% compared with the old program. The model also proved its efficiency by taking less than 20 seconds, equal to only 4% of the old processing time, to produce an output.

1 Introduction

As one of the most crucial but vulnerable determinants of crop development, seedling emergence should receive special care and judicious investments from plant breeders and growers (Lamichhane, Messéan & Ricci, 2019). The future crop health and fitness of plants might be substantially impacted by short delays in the emergence time determination (Verdú & Traveset, 2005). However, with large agricultural crops, manually conducting in-field investigations to monitor seedling emergence is onerous, time-consuming, and virtually unviable. Also, the time it takes breeders to count emerged plants by hands will not let the window of observations to hold and consequently side-by-side comparisons cannot be performed. To respond to these issues, agriculturists have considered the idea of crop scouting with the support of aerial imagery because image acquisition devices operated aerially at various altitudes are capable of capturing a massive amount of useful information about seedling emergence and other crop characteristics in the field (Jin et al., 2017). Particularly, with the increasing prevalence of unmanned aerial vehicles (UAVs), widely known as drones, taking photos of surface objects from a bird's-eye view, it has been recognised that such aerial images could be taken on crop fields and become an invaluable source of data for streamlining the identification and analyses of not only seedling emergence but also other phenotypic traits of plants.

Many cutting-edge technologies have been developed to harness aerial crop images to assist agriculturists in automatically identifying early emergence of plants, especially the instruments that employ computer vision and computational image processing. The advent of machine learning tools and artificial neural networks, in conjunction with recent improvements in imagery resolutions, has given rise to the implementation of pixel-based classifiers that can detect pixels containing emerged plants and count visible seeding emergence in farm images, then comparisons can be made with the ground-truthed counts (van der Merwe et al., 2020). However, traditional approaches to image classifications faced some problems that are idiosyncratic to agriculture; for example, multiple objects might exist in a single input image with large variability, or clustered growth of objects might cause severe occlusions, impeding the resulting accuracy of yield estimation (Wosner, Farjon & Bar-Hillel, 2021). As one of the remarkable advancements in computer vision, object detection tools have played a crucial role in a diverse range of plant-growing domains, and counting tasks like the identification and estimation of plant emergence is not an exception, with the initial goal of achieving greater accuracy (Wosner et al., 2021).

To efficiently identify plant emergence and measure the counts of seedlings on crop images, this paper aims to propose the implementation of a popular object detection framework called YOLO (You Only Look Once), based on an experiment of faba beans germination. With a view to enhancing a workflow executed by the Biometry Hub group at the University of Adelaide for identifying faba beans early emergence in UAV-taken images, the experimental outputs of this paper are optimistically applicable in production. Different YOLO algorithms will be trialled to choose the most effective model, which will then be put through a final performance evaluation test and some comparisons with the current programs before a formal implementation. The paper is structured as follow: Section 2 begins with some introductory information about the data to be utilised for analyses; Section 3 elaborates on the problems with the current workflow and the reasons for conducting these experiments; Section 4 provides a summary of previous research that might be related to this topic; Section 5 follows with some technical descriptions of the YOLO models to be experimented; Section 6 will display the experimental results of model performances, including the chosen model and its performance; Section 7 will give some general discussion points revolving around the results and some potential limitations of the model proposed; and Section 8 will conclude the project.

2 The dataset supplied

As the property of the Analytics for the Australian Grains Industry (AAGI) and the Biometry Hub, the data used to produce the outcomes in this paper has a collection of plot images for the early emergence of faba beans from a complete two-year seedling experiment sponsored by the Grains Research and Development Corporation (GRDC) in Australia. The crop field is located in Strathalbyn, South Australia and the aerial photographs of its plots were taken in 2022 by the Matrice 300 drone equipped with a Real Time Kinetic (RTK) photogrammetry surveying package and a Zenmuse P1 Camera with a 50mm lens. The entire field is divided into 99 rows and 12 columns to make up a total of 1188 plots, each of which corresponds to an image with a rough dimension of

2000-pixel width and 6000-pixel height, as visualised in Figure 1. Each individual plot was photographed on three different days, resulting in certain differences in the plant sizes and the visibility of plant emergence. Within the scope of this project, only the plot images in the third day will be used because they exhibit abundant visible plants that started to emerge, which makes the plant identification and the experimental results less influenced by human errors or by subjective visual perception of seedling emergence.

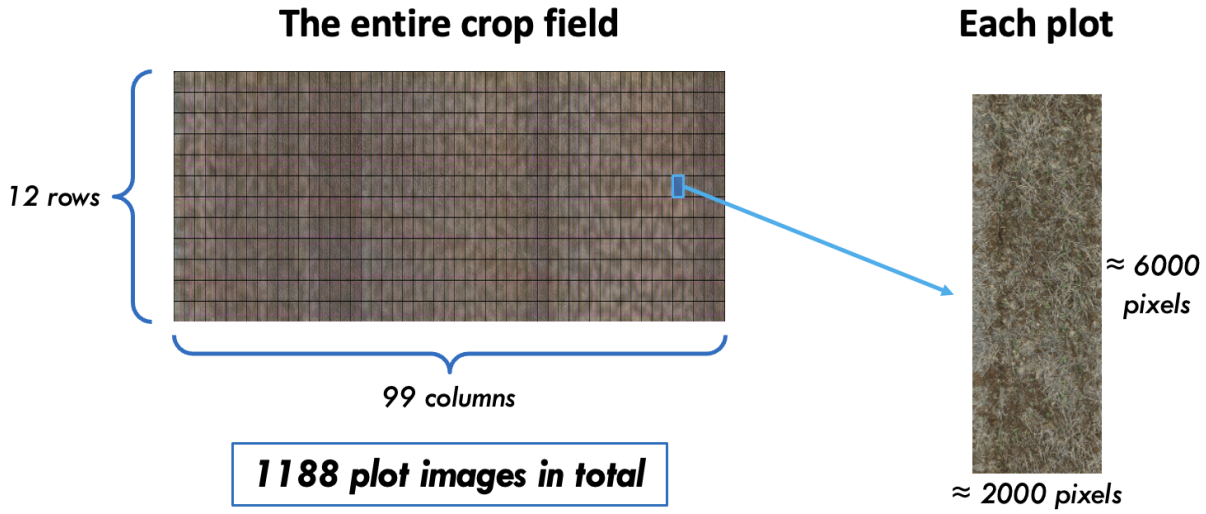

Figure 1: An illustration of how the faba beans crop field is divided into plots

The detection models built in this paper fall under the supervised learning paradigm, so they will require many annotation inputs that have to be manually created by users. The manual effort exerted in preparing those annotations is tremendous, but scanning through numerous plot images with such a huge dimension is not a necessity. Hence, not all of the data will be used, rather a subset of it will be taken, with an assumption that the selected plot images are sufficiently representative of the entire crop field. The data preparation stage will be described with more details in Section 5.3.1.

3 Problem identification

3.1 How the current programs work

Targeted to solving specific problems with the workflow executed in the Biometry Hub at present, an illustration is given about how the current toolsets function, how they have caused considerable issues, and the motivation for seeking potential improvements like object detection models. All the information regarding this workflow is retrieved from the documentation of the programs as well as my personal examination of the code scripts. The current workflow structure is composed of three main Python applications, each of which is responsible for a particular task in producing the final counts or estimation of emerged plants (i.e., faba beans). The first program, named **peascription.py**, takes a collection of plot images, in which users have manually marked the positions of each plant, generates many cutouts of size 64x64 pixels around each plant annotation, and randomly selects other 64x64 patches containing only the background, in order to prepare

the data input for training an image classification model. The second program, named **peatrain.py**, is where the model-training happens, it takes all the 64x64-pixel cutout annotations generated by **peascription.py** to train a Convolutional Neural Network (CNN) model that can classify whether a 64x64 cutout has an emerged plant (i.e., the positive class) or does not have a plant (i.e., the negative class). To give further technical details, the deep learning model set up in **peatrain.py** is a 10-layer neural network with three 2D convolutional layers containing 32, 64, and 128 filters respectively, a kernel size of 5x5 pixels, an epoch number of 30, a learning rate of 0.001 and a batch size of 64. Finally, in the third program, named **peatear.py**, users can input a full plot image, then the program will use the trained CNN model to make classifications on numerous 64x64 cutouts on that image, before producing the count of emerged plants.

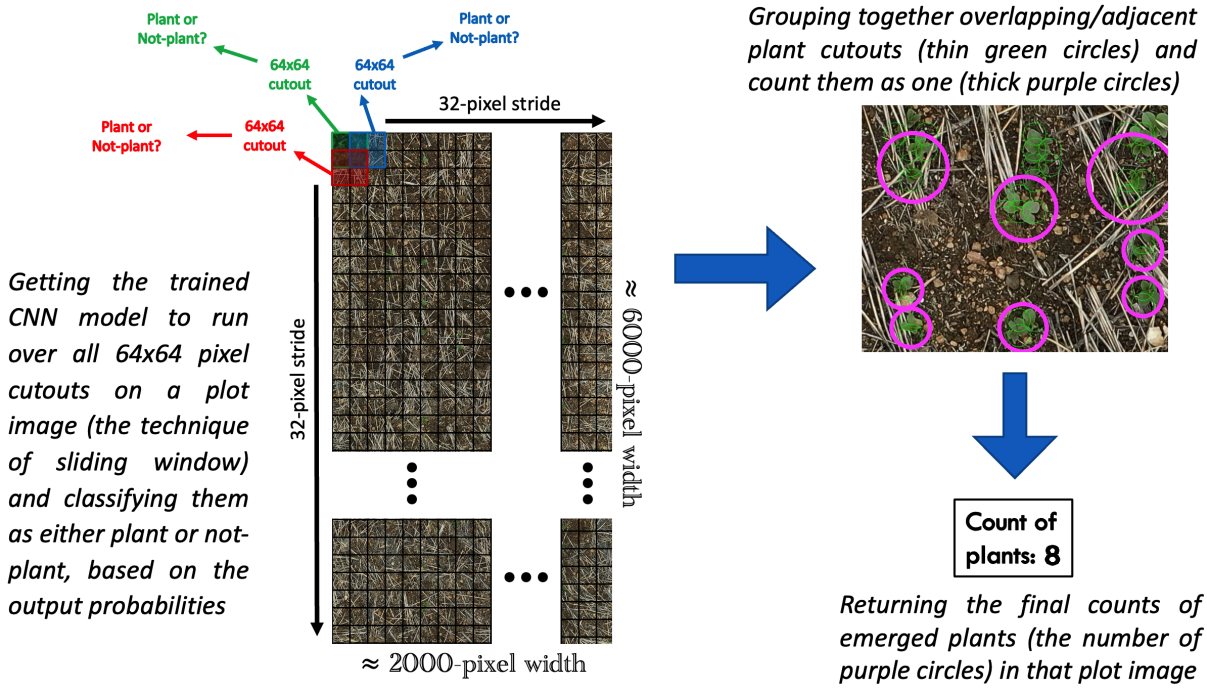

Figure 2: How the program peatear.py functions to produce the plant counts in crop images

To further explain the logic behind the program **peatear.py**, because the CNN model is trained to make classifications on 64x64 cutout images only, it cannot accept an input of a larger size than that. Meanwhile, a full plot image has an average size of 2000-pixel width and 6000-pixel height, so the model is unable to just take a full plot image and directly produce the count of emerged plants in it. Rather, the mechanism of that third program is based on the methodology of sliding window and grouping overlapping objects, as depicted in Figure 2. The former technique involves choosing a window of suitable fixed size to perform an inspection over a target image, using a trained binary image classifier (Helwan & Ozsahin, 2017). As an illustration with the Biometry Hub's toolset, when a full plot is entered, the program will divide the image into a multitude of tiny square fragments of size 32x32 pixels, and iteratively run the trained CNN model on each and every 64x64 cutout, discriminating whether it contains a plant or not based on prediction probabilities of the two classes, with a fixed step size of window movements, starting from

the top-left corner to the bottom-right corner. The window size (i.e., the dimension of a cutout input) is fixed at 64x64 pixels, and the stride is set to be 32 pixels, which means that the window will move 32 pixels at each step, and any two consecutive 64x64 cutouts to be classified will be halfway overlapping. This use of the sliding window technique was also referred to as an exhaustive search over the target image (Verschae & Ruiz-del-Solar, 2015). During that iterative process, every time the CNN classifier determines a 64x64 cutout as belonging to the positive class, it will draw a circle inscribed in that square to denote the predicted presence of an emerged plant. However, because a full plot image is divided into too many patches, a single plant may lie across multiple 64x64 cutouts, or a single detected plant may be denoted by multiple circles, so the count of plants cannot simply be the number of cutouts classified as positive. Instead, overlapping or adjacent cutouts with positive instances are grouped together and counted as one, two circles will be considered overlapping if the distance between their centers is smaller than the sum of their radii. This operation is carried out recursively, small circles can be merged to make bigger circles, big circles can continue to be merged to make even bigger ones, as long as they meet the criterion of overlapping, and only the last circle in that recursive operation is counted for the final estimation of emerged plants.

3.2 Problems with the current approach

There are at least four primary concerns arising from the current approach in the Biometry Hub's workflow. First of all, the plants in plot images cannot be precisely localised as the circles drawn by **peatear.py** are just inscribed in positive cutouts, they just denote the potential existence of a plant, not the precise position of it. This demerit can be attributed to the fact that the CNN classifier only learned to discriminate between plant and not-plant instances but was not trained to pinpoint exactly where the plants are in the annotation images inputted in **peatrain.py**. As an example, in Figure 3a, a plant lies across three to four positive cutouts, causing the resulting green circles to be overlapping; consequently, the big purple circles to merge those overlapping green circles contain expansive background and fail to define the exact boundary of the plants. Without the precise locations of plants, the practicability of the program will mostly be confined to estimating early emergence only or will hardly be extensible to other phenotypic applications such as inspecting plant sizes or measure plant growth by overlaying crop images taken at different timestamps with various transparency levels.

In addition, the accuracy of plant counts is diminished because of the sliding-window and object-grouping methods above. The criterion to determine overlapping circles does not take into account the occlusion caused by plant clumping, and overlapping plants are not necessarily reflected by overlapping cutouts. Consequently, despite the profound ability of the CNN classifier to discriminate between plant and not-plant instances, the final counts of emerged plants are made erroneous and can even generate misleading results in further analyses of emergence estimation. The difficulty of acute occlusion caused by overlapping objects in agriculture was previously recognised in a research study by Wosner et al. (2021). In Figure 3b, although the CNN model was able to spot out almost all of positive cutouts (i.e., thin green circles), the final counts of emerged plants are just one (i.e., thick purple circles), while the true numbers should be at least four or five.

(a) Each purple circle denoting a plant contains a wide area of background

(b) Inaccurate counts of the emerged plants (equal to the number of thick purple circles) returned after grouping overlapping cutouts classified as positive (denoted by thin green circles)

Figure 3: Typical examples of the first two problems with the current workflow

Associated with the second problem, another issue could be recognised, only a part of the workflow, which is the model to classify 64x64 cutouts, is trainable. Meanwhile, the operation of translating those classifications into the final plant-emergence estimation is still manually manipulated with user-specified parameters like the window size, the window-moving step size, or the condition of overlapping cutouts. Not only does it undermine the efficiency of the overall workflow but it also acts as a catalyst exacerbating the inaccurate estimation of faba beans emergence.

Moreover, the sliding window technique is notorious for its elongated computational time and heavy memory usage, which could be deemed the most considerable problem, especially in this circumstance where the window size (i.e., 64x64 pixels) is only a small fraction of the full plot image dimension (i.e., approximately 2000x6000 pixels). When a classifier is applied to every single tiny patch on the target image like that, the program execution time will explode (Giusti et al., 2013), compromising the its feasibility in real-time applications. It is calculated that for a 64x64 window moving over a 2000x6000 image with a 32-pixel step size, there are $\frac{2000}{32} * \frac{6000}{32} \approx 11718$ classifications to be made by the CNN model. CNN architectures work on the idea of getting a lot of kernels (of size 5x5 in this case) to convolve around an input image (i.e., a 64x64 cutout), so as to extract feature maps that define the appearance of the object of interest (i.e., emerged plants). Thus, 11718 classifications can create a massive number of computations to be performed at the backend, so along with excessive computational time, memory usage is also pushed to the extreme. According to Gouk & Blake (2014), if a CNN classifier contributes to a larger system while requiring such a prohibitively colossal memory allocation, the applications may end up suffering disruptions on machines with limited computing resources. This situation is exactly similar to the current workflow of the Biometry Hub, it was observed that when the **peatear.py** program was experimented to run multiple full plot images in a row, it kept crashing after processing just 12 to 15 items, whereas there were hundreds to thousands of full-size images in total queued to be run. In turn, more labours had to be invested in monitoring the code execution and in refreshing the

program when it crashed, causing the amount of manual intervention to increase.

In brief, the overarching challenge that needed to be addressed is how to efficiently transform the classification outputs of the CNN model into the positions and the final count of emerged plants in a full plot image, or how to combine the task of 'element-wise' image classification with the technique of object localisation to produce more accurate outputs with fewer computational resources. Object detection models like the YOLO framework are a perfect answer to that challenge.

4 Related works

In the past, object detection methods have been examined in relation to a variety of agronomic activities such as disease control, quality inspection, fruit damage diagnosis, and so on (Nawaz et al., 2022). Among them, crop-counting and crop-scouting tasks to investigate plant phenotypes like seedling emergence have attracted great attention from researchers, many of those tasks also involve the use of drone images as the inputs. A typical example is the research conducted by Hasan et al. (2018), who exploited the Region-based-CNN (R-CNN) object detection algorithm to detect and count wheat spikes in images for the estimation of one-season yield on a wheat field trial. Having a more complex architecture, the Faster R-CNN algorithm was modified by Liu et al. (2020) to automatically count the number of maize tassels on images taken by a UAV device. Also enhancing the Faster R-CNN model, Deng et al. (2021) introduced the feature pyramid network to accelerate the job of counting grains in rice panicles with various visibility conditions. A more recent example is the invention of EmergeNet, a deep learning model that incorporates three different backbone image-processing networks to identify and record the emergence of maize coleoptiles in video sequences (Das et al., 2023).

In the publications listed above, object detection methods were the core element guiding the research objectives, but none of them involved YOLO models. As a popular object detection framework, YOLO has also been an emerging topic of interest in the literature of many plant-breeding domains. For instance, the fourth and fifth versions of YOLO were implemented to equip fruit-picking robots with the right harvesting mechanisms by visually recognising different types of kiwifruits and of apple growth forms in orchards (Suo et al., 2021; Lv et al., 2022). Other than fruit cultivation, Zhang, Zhu & Wen (2023) proposed the combination of a Transformer deep learning technique with the fourth version of YOLO to label densely distributed maize tassels in UAV-captured images. To accomplish the same goal, Pu et al. (2023) devised the Tassel-YOLO model, originated from the seventh version of YOLO, by employing a novel cost function to improve training speeds and inference accuracy. On top of the YOLO X version, Xiang et al. (2023) developed the YOLO POD model to predict the number of soybean pods by adopting a multi-tasking structure design with a modified loss calculation.

From a narrower perspective where YOLO models are studied with regards to counting seedling emergence, the third and fourth versions of YOLO were utilised by Oh et al. (2020) and Tan et al. (2022) respectively on UAV imagery as well as high-resolution videos

to count cotton plants at their early crop-establishment phase. Lin et al. (2022) chose to apply the fifth version of YOLO to the detection of peanut seedlings and observed that the model returned an extraordinary accuracy while reducing 80% of the time it would have taken humans to count. In a similar sense, Hamidon & Ahamed (2023) applied the seventh YOLO model to spotting out and counting defective seedlings of lettuce under different indoor farming conditions. These studies appear to be most analogous with what this paper aims to accomplish as not only did it explore the use of a YOLO model but its topic was also related directly to plant counting at the early emergence phase.

Despite that analogy, no study about YOLO or object detection in general has been found to target specifically at the early emergence of faba beans by the time this report is written. Meanwhile, the optimisation of object detection models are varied for different scenarios, and there is no one-size-fits-all solution or no general approach for applying object detection to the plant science (Lu et al., 2023). Therefore, with faba beans as the detection subject in UAV imagery, this paper is expected to contribute to the initial stage of investigating and implementing object detection methods for the phenotyping study of this plant, not just at its early emergence phase.

5 Methods

5.1 Object detection

As suggested by its name, object detection is a method seeking to identify, classify and localise the objects of interest in the input images, it usually involves the placement of rectangle bounding boxes around each object as the final output to define the object's boundary (Sanaeifar et al., 2023). In the experiment with faba beans, as emerged plants are the only object of interest for detection, this paper is dealing with a one-class detection problem, as opposed to multi-class object detection. One of the universally accepted expressions with numerical values for bounding boxes is to specify the coordinates of its top-left and bottom-right corners on the image, using the format (x_1, y_1, x_2, y_2) (where x_1, y_1 represent the top-left coordinates and x_2, y_2 represent the bottom-right coordinates) (Szegedy, Toshev & Erhan, 2013). Hence, to perform predictions on numerical values for the bounding box coordinates, contemporary object detection models also adopt regression approaches, along with image classification.

Compared with image classification like basic CNN models, object detection goes one step further by determining the positions of objects. In fact, CNN architectures are employed at the backbone network for extracting feature maps from the input images, there are two widely recognised model types to describe how CNNs are used in object detection models: two-stage and one-stage, as summarised in Figure 4. In the former, candidate object regions are proposed (i.e., region proposals), then CNNs will help extract feature maps from these regions to classify them into different object classes and to perform regression for generating bounding boxes separately (Jiang & Li, 2020). The sliding window technique is also applied in two-stage object detection models but on the proposed regions rather than on all the cutouts on the image. In spite of that, object

detection frameworks based on region proposals still need separate training for their two components, object classification and bounding box regression, so their processing time remains a significant hindrance, as maintained by Zhao et al. (2019).

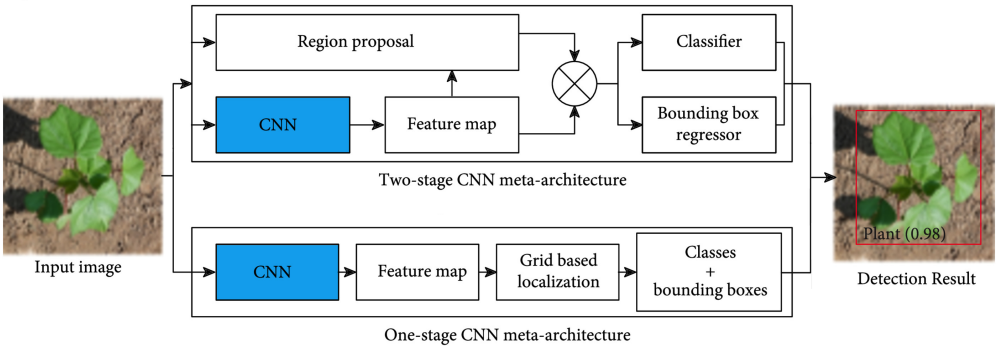

Figure 4: The internal operation in one-stage and two-stage models (Jiang & Li, 2020)

In contrast, one-stage models blend those two components into one single module, learning to map image pixels directly to bounding box coordinates and assign class probabilities. Due to the elimination of region suggestions and separate training processes, these models offer faster processing speed with a simpler model architecture, accompanied by the possibility of compromising accuracy (Sanaeifar et al., 2023). As computational efficiency is a major concern with the current workflow, one-stage models appear to be an appropriate solution. YOLO, a typical algorithm series of object detection methods that are grounded on this single-stage mechanism, would be the main focus of this paper.

5.2 The YOLO series algorithm

Introduced by Redmon et al. (2016), You Only Look Once (YOLO) is an object detection algorithm that converts target detection into a regression problem, making use of the topmost feature map on the target image to predict the position and the category of objects with a confidence score attached to each bounding box at the final outputs. As exhibited in Figure 5, YOLO divides a target image into a SxS grid, if the midpoint of an object falls into a grid cell, the cell will be used to detect that object by generating bounding boxes and corresponding confidence scores, then encoding them together with the class probabilities (Redmon et al., 2016). The CNN architecture in the original YOLO model, or the first version of YOLO (YOLOv1), consists of 24 convolutional layers followed by 2 fully connected layers, as displayed in Figure 6. By abolishing the intermediate tasks of region proposals as in a single-stage object detector, YOLO series models offer faster processing speed and they are thus suggested to be more suitable for agricultural applications, particularly in real-time scenarios (Chen et al., 2021).

With only one feature map on the entire image, the YOLO model originally received some skepticism about its ability to detect small objects and objects that appear in groups or in clusters, but successive versions and revisions of YOLO have endeavoured to address this issue with substantial modifications in its backbone and structure characteristics (Jiang et al., 2022). Over the years, different YOLO versions have attempted to make improvements

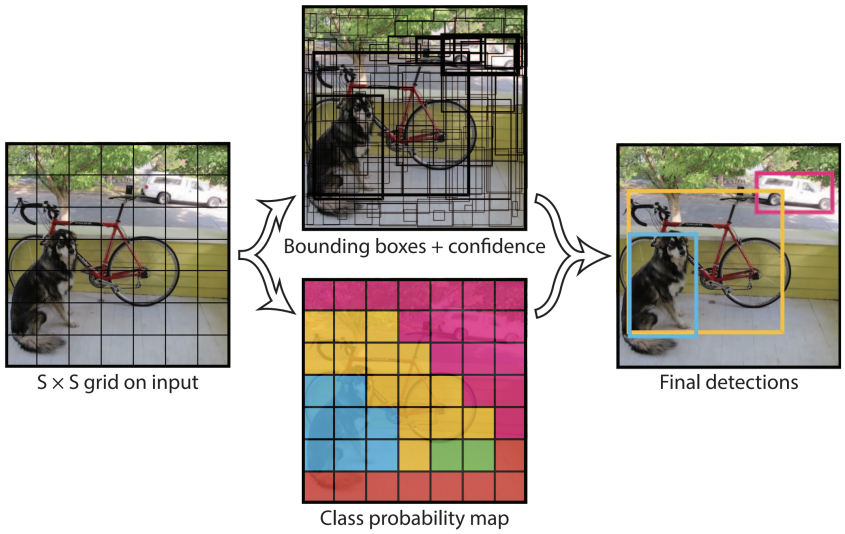

Figure 5: The basic mechanism of how a YOLO model works (Redmon et al., 2016)

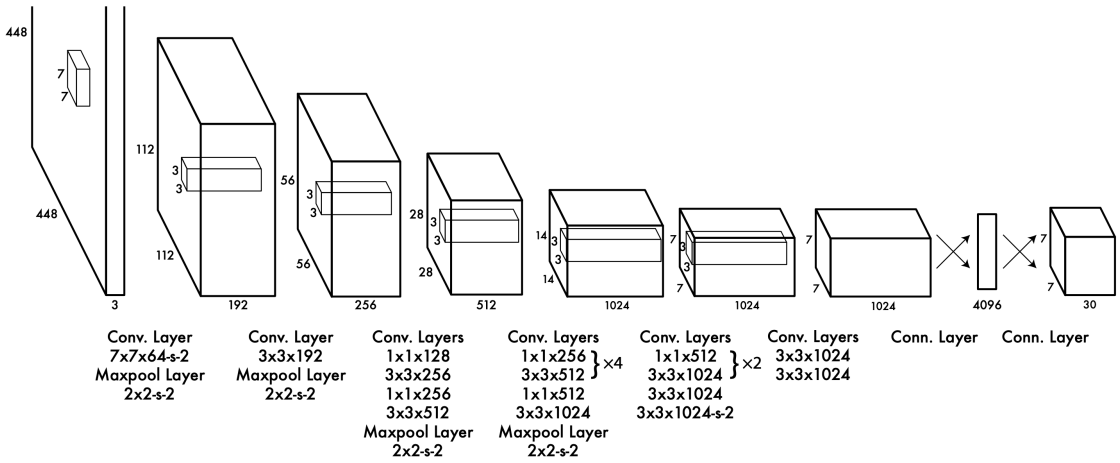

Figure 6: The original CNN architecture constructed in YOLOv1 (Redmon et al., 2016)

with an emphasis on the balance between performance and efficiency. There have been eight main YOLO versions (named from YOLOv1 for the first version to YOLOv8 for the latest version) and some other revised-limited variants such as YOLO-LITE or PP-YOLO. In this paper, the five most recent YOLO versions will be trialled, including:

- **YOLOX** released by Ge et al. (2021),
- **YOLO version 6 (YOLOv6)** released by Li et al. (2022),
- **YOLO version 7 (YOLOv7)** released by Wang, Bochkovskiy & Liao (2022),
- **YOLO version 8 (YOLOv8)** released by Jocher, Chaurasia & Qiu (2023) in the software company Ultralytics, and
- **YOLO Neural Architecture Search (YOLO-NAS)** released by the research team in the software company Deci AI (Aharon et al., 2021).

The technical designs and architectures of these models could be found in Appendix B.

5.3 Experimental procedure

To build and experiment the above YOLO models, this paper adheres to a standard machine learning workflow, starting with data preparation data preprocessing, and data splitting, then moving on to model training, model selection, and a final evaluation of the chosen model. Some of these steps are detailed below.

5.3.1 Data preparation & Data splitting

As mentioned, to reduce manual tasks, only a part of all the plot images taken for the faba beans experiment will be selected for the training and validation of the YOLO models. Out of the 1188 plot images, 30 of them are randomly selected to compose the entire dataset that will be used for the training procedures of YOLO models. Synonymously, these 30 plots are assumed to simulate the only data available on hand and to be representative of the entire faba beans crop field. In these 30 plot images, bounding box annotations are manually placed onto each and every single emerged plant to create the ground-truth references, as shown in Figure 7. Using the open-source software Roboflow[1], there are a total of 5315 bounding boxes that have been annotated on the images. The image and annotation files are formatted contingent on the annotation convention of each YOLO model to be trained, and Roboflow takes care of all these formatting tasks.

Figure 7: An example of manually annotating bounding boxes onto each emerged plant in the selected plot images to create the ground-truth labels, using the Roboflow software

The 30 plot images selected are divided into three parts called the training, validation and testing datasets. The training set containing 20 images with 3432 bounding boxes is used directly for training the YOLO models. The validation set containing 5 images with

[1]Roboflow: https://roboflow.com

907 annotations is used for keep track of the generalisability of each model during the training process and for comparing their performances for model selection. After a YOLO model is chosen, the testing set containing 5 images with 976 annotations will resemble the plot photos that has not been encountered in model-training, it is used to obtain an unbiased estimation of how the nominated model performs on unseen data.

In the faba beans experiments, the ultimate objective of implementing innovative computer-vision and image-processing models is to count the number of emerged plants in plot images. So, besides the above 30 plots that are used specifically for the task of plant detection, another 30 plot images are also selected randomly to assess the plant counts returned by the nominated model. To emphasise, these 30 plot images will only be used at the final evaluation stage, after the most effective model is nominated, and they are only for evaluating the final plant counts, not for evaluating the detection performance. To differentiate between these 30 plots and the 5 annotated images in the testing data used for evaluating models' detection ability, the former will be referred to as the count-testing plot images. With these 30 count-testing plot images, the ground-truth counts have been recorded and made ready to compare against the model-made counts.

5.3.2 Data preprocessing - Image tilting

For the faba beans experiment, the sizes of emerged plants are drastically small in proportion to the target image size, so apart from using recent YOLO versions with better performances on small object detection, a preprocessing step of tiling images will be applied on the input data (Ünel, Özkalayci & Çiğla, 2019). This preprocessing step can also accommodate the issue of input size with the YOLO models. The five YOLO models to be experimented are designed to accept an input size 640x640 pixels by default, which means that the images to be fed into the models are expected to have a square shape with each side being 640-pixel long. If an image with a larger size than that is entered, it will be scaled down and squeezed into a 640x640 square, causing the loss of potentially useful information as well as the distortion of objects (i.e., emerged plants) in the image (Plastiras, Kyrkou & Theocharides, 2018). Although that input size is adjustable, if it deviates too much from the default specification, it may not work in harmony with the components constructed in the models and the model may occupy more memory space (Hashemi, 2019), so plot images of size 2000x6000 pixels should not be input directly into the models. However, if an image is separated into too many tiles, the problem of an emerged plant lying across multiple tiles may become severe, similar to what has already been suffered by the current toolsets. Therefore, it is decided that image-tiling will be applied in conjunction with slightly adjusting the input size of YOLO models. More specifically, each plot will be equally divided into 12 non-overlapping tiles by one vertical split and five horizontal splits so that each tile is a square with an approximate size of 1000x1000 pixels. With this preprocessing technique, the training data has 20 x 12 = 240 image tiles, the validation data has 5 x 12 = 60 image tiles, and the testing data has 5 x 12 = 60 image tiles. Simultaneously, the input size of all five models is increased from 640 to 1000, and the bounding boxes annotated before will be mapped accordingly.

This preprocessing step of image tiling is supposed to be applied on all three dataset splits as well as in both the training and inference phases. When the model is deployed on a full-size plot image, the 12 tiles can simply be stitched back together using image-processing libraries to produce the final counts of emerged plants.

5.3.3 Model training

To ensure the comparability among the five candidate YOLO models, each of them is trained over 30 epochs with a batch size of 16 and an input size of 1000. This input size represents the dimension of square images to be fed into the model for training, and it is set to 1000 to correspond with the size of the square image tiles obtained from the aforementioned data preprocessing step. An early-stopping mechanism of 15-step patience is also applied, which means the training process will get terminated if the performance scores do not show any improvement over 15 consecutive epochs.

5.4 Evaluation metrics

5.4.1 Mean Average Precision at Intersection-over-Union threshold 50% (mAP50)

The main evaluation metric used for comparing YOLO models and for reporting their overall performance is mean average precision (mAP), which is a condensation of three model performance measurements: Recall, Precision, and Intersection over Union (IoU).

a. Intersection over Union (IoU)

IoU measures the precision of the predicted bounding boxes in comparison with the ground-truth annotations and it can help determine whether a detection is a True Positive (TP) or False Positive (FP). Ranging from zero (denoting no intersection at all) to one (denoting a perfect match), IoU is calculated as the ratio between the area of overlapping and the total area made up by the ground-truth and predicted bounding boxes, as shown in Figure 8 (Padilla et al., 2021). Usually, a threshold of IoU is specified to characterise detections in a classification context. For example, if the IoU threshold is chosen to be 50%, then any detections with IoU values greater than or equal to 0.5 will be considered TP cases, otherwise they will be considered FP cases (Everingham et al., 2010).

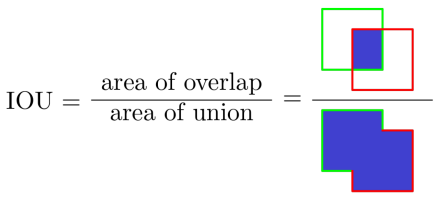

Figure 8: The formula to calculate IoU (Padilla et al., 2021)

b. Recall and Precision

With a pre-specified IoU threshold, recall and precision can be computed from the number of TPs and FPs obtained for all detections. As indicated in Equation 1 and Equation 2,

recall measures how well a model can identify the positive class by returning the fraction of TPs or correct plant detections over the total number of ground-truth plant instances, whereas precision measures the reliability of the detections made by a model for the positive class by returning the proportion of TPs or correct detections over the number of all positive predictions that have been made (Padilla, Netto & Silva, 2020).

$$Recall = \frac{TPs}{all\ ground\_truth\ plant\ instances} \tag{1}$$

$$Precision = \frac{TPs}{all\ plant\ detections} \tag{2}$$

c. The Precision-Recall (PR) curve

A model is considered good if it can identify all ground-truth positive instances (i.e., aiming for high recall) while also ensuring the relevance of its detections (i.e., maintaining high precision); in other words, there is always a trade-off between recall and precision, depending on the chosen confidence level (Padilla et al., 2021). Recall and precision values at different confidence thresholds can be plotted on a 2-dimensional graph, called the Precision-Recall (PR) curve, as a decreasing function of the confidence level. To encapsulate these varying precision and recall values, the area under the PR curve is computed as a single metric. In practical cases, a PR curve often shows a zig-zag pattern, so intermediate points are interpolated to facilitate the calculation of AUC (Oksuz et al., 2018). Figure 9 provides an example of the PR curve and the point interpolation used for computing the PR-AUC, and average precision (AP) is computed as the area under this interpolated PR curve (Henderson & Ferrari, 2017).

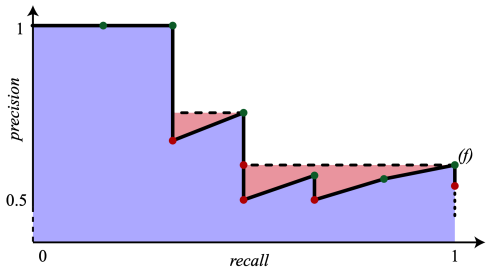

Figure 9: A PR curve plotted at a sequence of different confidence levels. The original curve is the solid black line, the pink area is the result of point interpolation for smoothing the PR curve. The average precision (AP) is the area under the interpolated curve, which equals to the total area of blue and pink regions in the plot (Henderson & Ferrari, 2017)

d. mAP50 and model selection

mAP is the mean of AP scores obtained individually for all classes in an object-detection problem. In the faba beans experiment, emerged plants are the only subject of detection,

so the mAP is just the AP computed for the 'plant' class. Also, the IoU threshold is chosen to be 50% in this case, or the evaluation metric to be computed is the mAP at IoU of 50%, widely abbreviated as mAP50. In model selection, the YOLO model with the highest mAP50 score on the validation data will be nominated for the final evaluation stage on the testing data for further analysis on its true detection performance.

5.4.2 Root Mean Squared Error (RMSE) on the count-testing plot images

In the final evaluation stage, in parallel with the use of mAP50 to evaluate the detection performance of the nominated model, it is also necessary to assess the plant-count errors of that model because counting emerged plants in plot images is the ultimate goal of this project. Thus, on the 30 count-testing plot images, the chosen model will be set up to predict the count of emerged plants by obtaining the number of plant detections made. Subsequently, the predicted counts will be collated with the ground-truth counts to get 30 count errors, which will then be condensed into a single score, using the Root Mean Squared Error (RMSE). As stated by Jierula et al. (2021), RMSE measures the average magnitude of error between the predicted (model-made) plant count and the actual (manual) plant count, the lower an RMSE score is, the more effectively a model performs. Jierula et al. (2021) also pointed out that RMSE was highly interpretable relative to the variable being studied as they have the same measurement unit.

The formula to calculate RMSE is $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$, where n is the number of data points to calculate residuals, $\hat{y}_i$ is the predicted value, and y_i is the actual value. When this formula is applied to evaluating the model performance on the count-testing data, n is equal to 30 as there are 30 count-testing plots, $\hat{y}_i$ and y_i are the model-made count and the true manual count of plants for a plot respectively.

6 Results

6.1 Model comparison and model selection

In this experiment, five candidate YOLO models have been trained on the training data, then evaluated on the validation data to get an estimation of their performance, based on the mAP50 metric. The mAP50 scores obtained on the validation data are presented in Table 1. It can be clearly seen that the YOLOv8 model came out on top with the highest

Table 1: The mAP50 scores obtained by the five candidate YOLO models on the validation data (Arranged in a descending order of validation mAP50 scores)

Model	Validation mAP50
YOLOv8	0.959
YOLO-NAS	0.939
YOLOv7	0.897
YOLOX	0.852
YOLOv6	0.763

validation mAP50 score of 95.9%. At the second place, the YOLO-NAS model returned a validation mAP50 score of 93.9%, followed by the YOLOv7 model with an mAP50 score of 89.7%. In contrast, with a validation mAP50 score of only 76.3%, the YOLOv6 model appeared to be left far behind all the other models. Therefore, YOLOv8 is nominated as the most optimal model for detecting emerged plants in the faba beans plot images and will be proceeded to the final evaluation stage on the testing data.

The learning curves of the YOLOv8 model derived from its training log are plotted in Figure 10. The YOLOv8 model is trained to optimise three different losses called classification loss, bounding box loss, and distributional focal loss (DFL). As can be seen in Figure 10, over the 30 epochs of training, all the three validation losses exhibited a decline, the validation mAP50 exhibited a steady increase, and they started to stabilise from the 20^{th} epoch. It indicates that 30 epochs are sufficient for the YOLOv8 model to find the minimum losses during its optimisation process. Across the 30 epochs, the curves of training losses and validation losses decreased consistently almost in analogous patterns, which could be indicative of the likelihood that the YOLOv8 model has succeeded in avoiding the pitfall of overfitting, as advised by Domhan, Springenberg & Hutter (2015). It is numerically observed that the best validation mAP50 was achieved by YOLOv8 at the 30^{th} epoch, so the model weights obtained in this epoch were saved for its parameters.

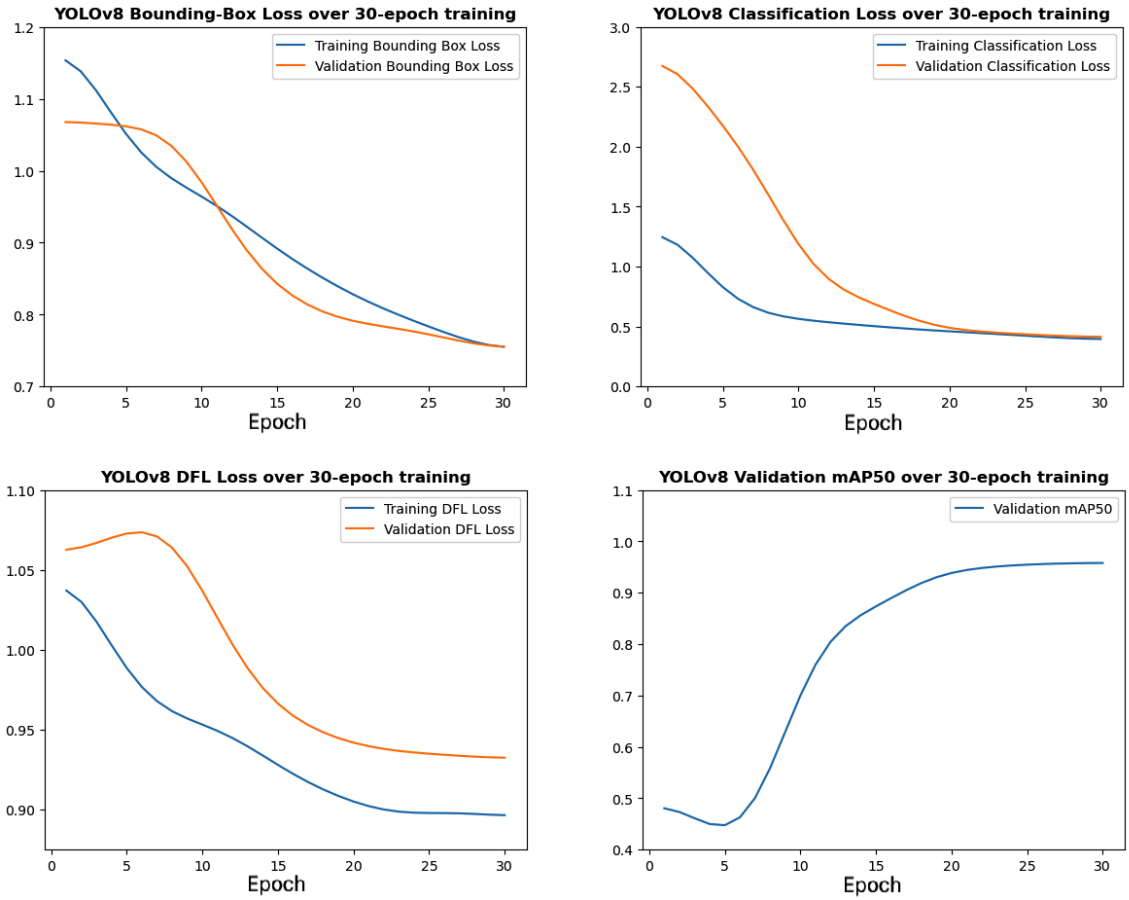

Figure 10: The smoothened learning curves from the YOLOv8 model's training log

6.2 Final evaluation of the chosen model (YOLOv8)

After getting nominated as the optimal model for the faba beans experiment, YOLOv8 has been tested on the five testing plot images to unbiasedly assess its true performance in detecting emerged plants. As shown in Figure 11, the testing mAP50 it achieved on the testing images was 94.6%. This number is very close to the model's validation mAP50 of 95.9%, so it could once again confirm that the YOLOv8 model did not overfit the training data. With the PR curve reaching nearly the top-right corner of the plot and the testing mAP50 approaching 100%, the YOLOv8 model has proved its outstanding performance on unseen data and its robustness in this object-detection task.

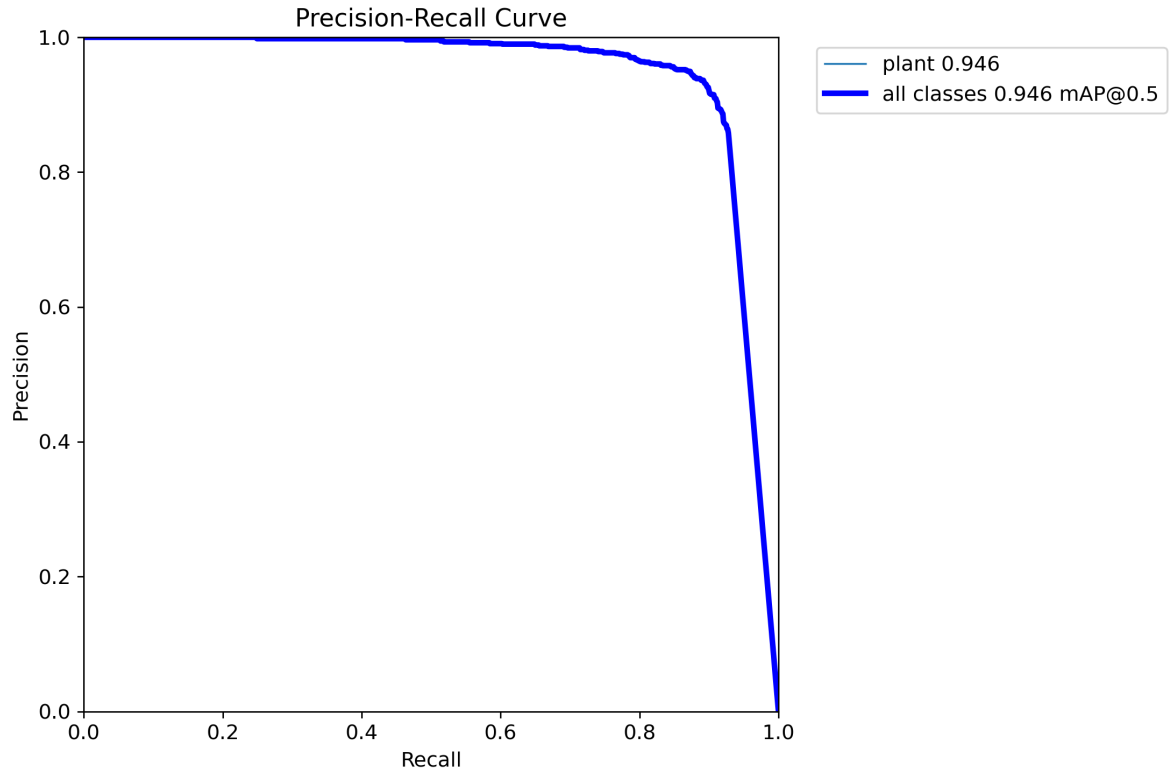

Figure 11: The PR curve and mAP50 achieved by the YOLOv8 model on the testing data

6.3 Evaluating the plant-counting ability of the chosen model (YOLOv8)

Apart from assessing the detection performance of the chosen model, its ability to count emerged plants should also be evaluated to address the initial objective of plant-counting. After being trained, the YOLOv8 model has been set up in a program that performs a complete process of detecting and counting emerged plants in full-size plot images, from tiling the images for preprocessing and running the trained model on each tile, to stitching those tiles back together and returning the counts of plants (i.e., the number of detections). At this final evaluation stage, the program will be tested on the 30 count-testing plot images selected before to examine how accurate the plant counts would be, in comparison with the ground-truth counts. Table 2 gives the comparisons of plant counts

between the ground truth, the old algorithm (i.e., the old program with the sliding window and object-grouping techniques), and the new program built upon YOLOv8.

Table 2: The plant counts returned on 30 count-testing images by the new program (built on YOLOv8) and by the old algorithm (peatear.py) & the RMSE scores computed accordingly

Plot Image	Ground-truth plant count	Plant count made by the old algorithm (peatear.py)	Plant count made by the new program (built on YOLOv8)
col2row14	121	96	126
col2row48	146	125	154
col3row2	138	102	143
col3row45	154	127	158
col3row73	132	99	138
col3row86	111	86	118
col4row27	160	131	167
col4row34	145	117	147
col5row2	142	126	150
col5row18	114	91	121
col5row28	137	98	143
col6row16	139	118	147
col6row81	134	103	132
col7row21	176	142	181
col7row27	117	90	113
col7row62	159	110	168
col7row90	123	106	126
col8row9	131	114	138
col8row35	62	57	64
col8row82	118	95	127
col9row15	133	105	142
col9row18	108	89	116
col9row22	122	103	125
col10row10	116	90	123
col10row18	106	87	102
col10row20	135	107	141
col10row69	119	99	126
col10row77	140	115	137
col11row88	162	123	170
col12row48	193	168	200
		RMSE = 27.113	**RMSE = 6.261**

It can be seen that the YOLOv8-powered program returned the plant counts that were very close to the ground truth, the differences are no more than 10 plants for each plot image. Meanwhile, the **peatear.py** program always returned a lower number of plants than the ground-truth counts, the differences even reach to 30 or 40. These prediction errors were condensed into two RMSE scores for an easier comparison. On average, the

YOLOv8 model miscounted just over 6 plants in those count-testing plot images, improving the plant-count accuracy by $\frac{27.113-6.261}{27.113} \approx 76.9\%$, compared to the old algorithm.

6.4 Comparing the runtime of the new program with the old one

Table 3: Comparisons of the runtime between the new program (built upon YOLOv8) and the old one (peatear.py) on 30 count-testing plot images (For comparability, both programs were run on the CPU processor of an Apple Macbook Pro M1 13.3' laptop)

Plot Image	The processing time of the old algorithm (peatear.py) in seconds	The processing time of the new program (built on YOLOv8) in seconds
col2row14	459	19
col2row48	436	18
col3row2	400	16
col3row45	430	18
col3row73	447	17
col3row86	432	18
col4row27	413	17
col4row34	427	18
col5row2	403	16
col5row18	465	18
col5row28	447	17
col6row16	447	17
col6row81	447	18
col7row21	452	18
col7row27	416	17
col7row62	445	17
col7row90	445	17
col8row9	400	17
col8row35	445	16
col8row82	416	19
col9row15	442	18
col9row18	475	19
col9row22	445	18
col10row10	462	18
col10row18	458	18
col10row20	430	17
col10row69	474	17
col10row77	447	18
col11row88	458	17
col12row48	447	18
	Average = 440.333 seconds	**Average = 17.533 seconds**

When the two programs were being runned on the 30 count-testing images, the execution time was recorded to compare their efficiency. As reported in Table 3, the new program

took an average of 17.5 seconds to produce the output for each full-size plot image, and no image required more than 20 seconds to be processed. On the same device, the **peatear.py** program always spent at least 400 seconds running through each single image, sometimes it could even take more than 470 seconds. Compared with **peatear.py**, the new program has reduced the average execution time by $\frac{440.333}{17.533} \approx 96$ times. Not only has the YOLOv8 model proved its accuracy improvements in detecting and counting plants but it also helps mitigate the processing time and computational power required.

6.5 Some examples of the inferences made by the YOLOv8 model

The screenshots from Figure 12 to Figure 18 below are some of the outputs produced by the YOLOv8 model, in parallel with the ground-truth annotations, to epitomise the accuracy it has achieved in the task of counting emerged faba beans in plot images.

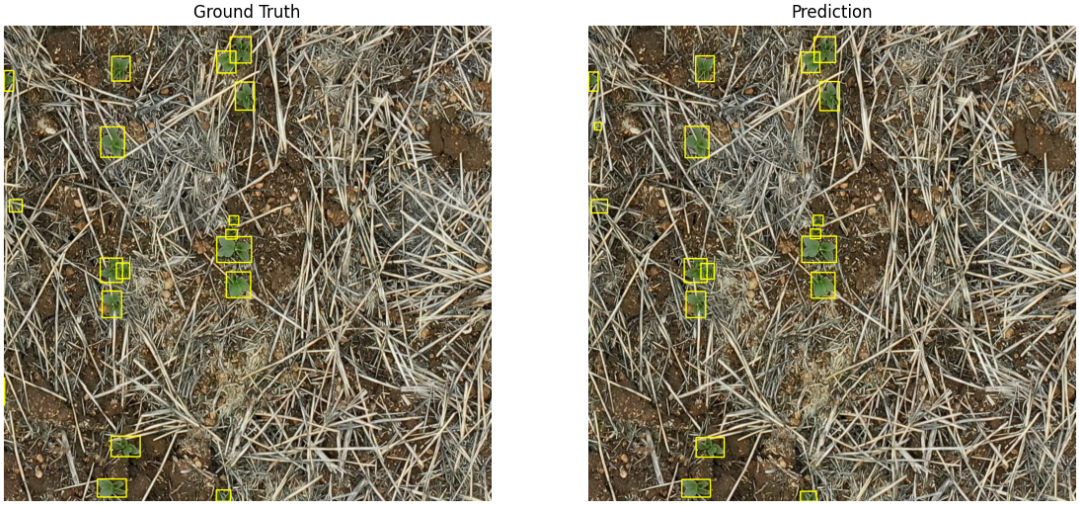

Figure 12: Example (1) of the YOLOv8 output, compared with the ground truth

Figure 13: Example (2) of the YOLOv8 output, compared with the ground truth

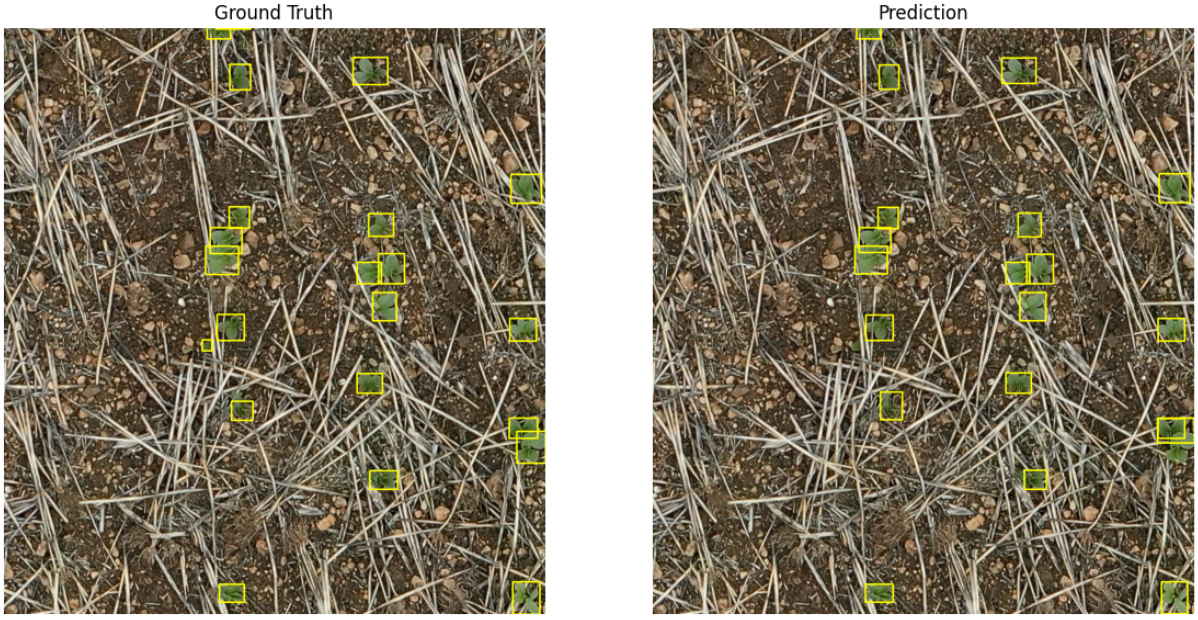

Figure 14: Example (3) of the YOLOv8 output, compared with the ground truth

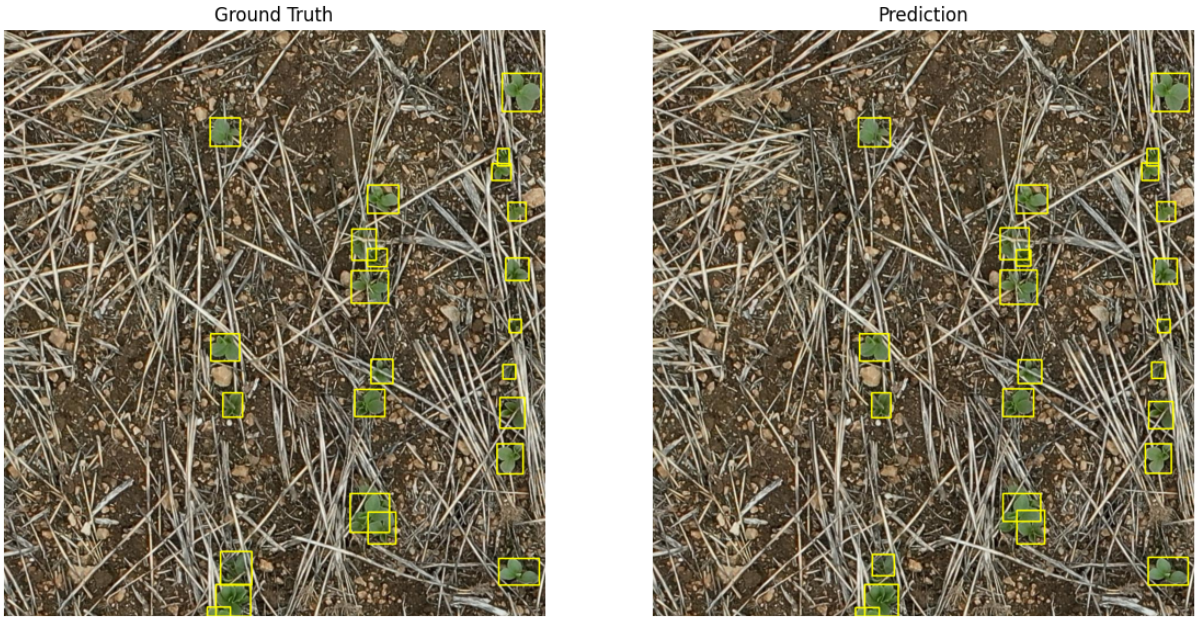

Figure 15: Example (4) of the YOLOv8 output, compared with the ground truth

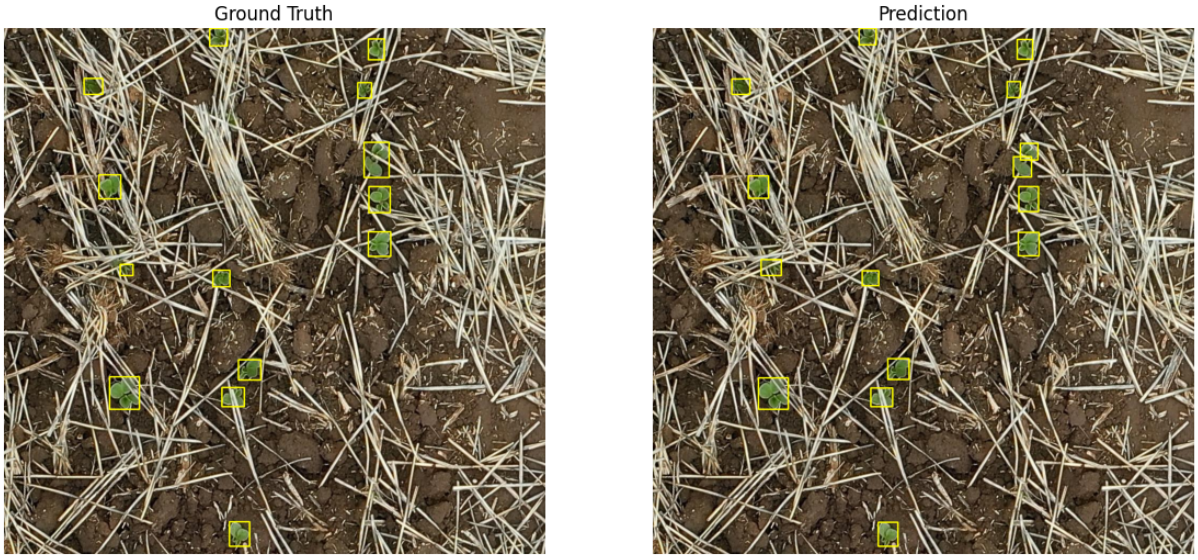

Figure 16: Example (5) of the YOLOv8 output, compared with the ground truth

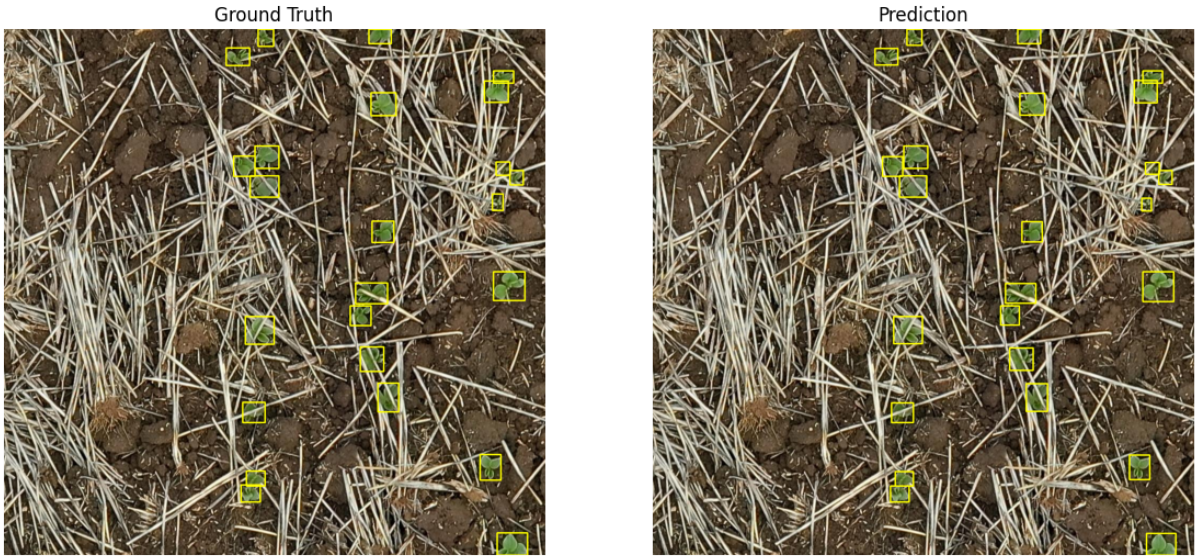

Figure 17: Example (6) of the YOLOv8 output, compared with the ground truth

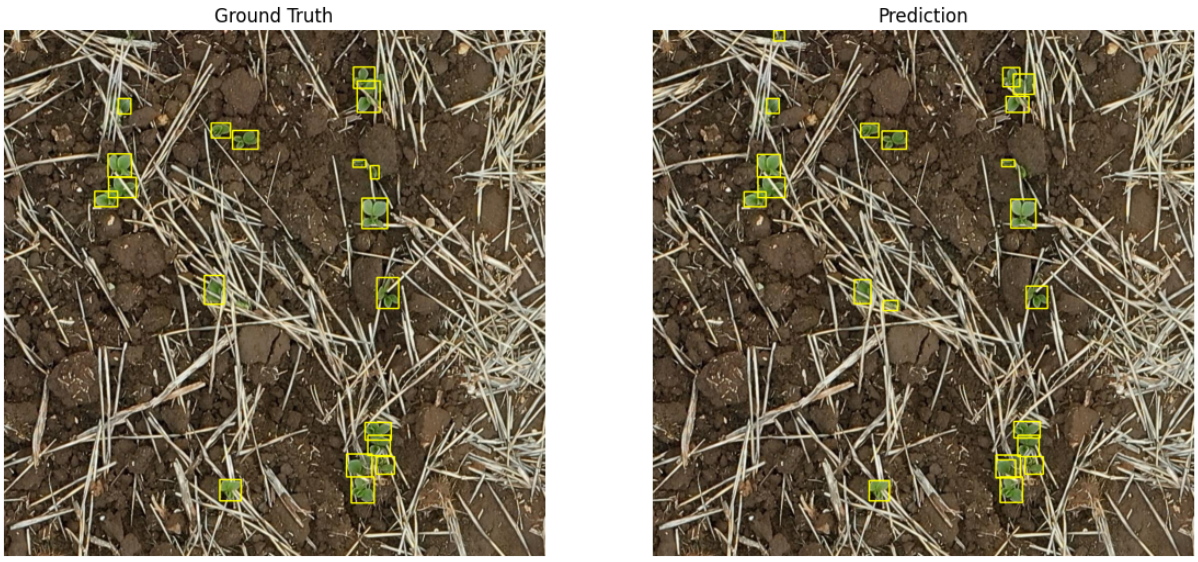

Figure 18: Example (7) of the YOLOv8 output, compared with the ground truth

Looking closely at the examples above, there are some rare cases where the YOLOv8 model made mistakes, which could be False Positives, False Negatives, or over-separating clumping plants, as illustrated in Figure 19 below. However, the frequency of these mistakes happening was observed to stay very low, and the number of these mistakes are trivial, considering the large numbers of emerged plants to be detected.

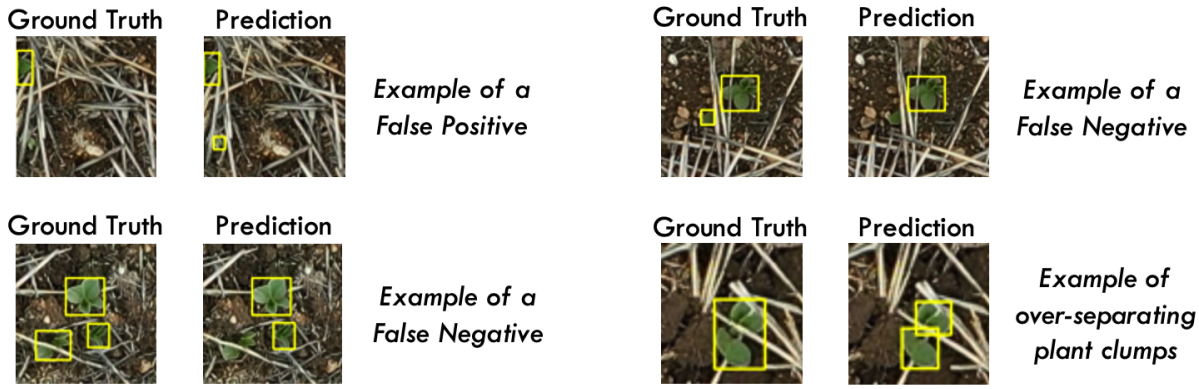

Figure 19: Some close-up examples of minor mistakes the YOLOv8 model made

Additionally, to highlight how effective the YOLOv8 model was at identifying and localising individual plants in clumping, a few more examples are provided in Figure 20 below, including the collation between the outputs produced by YOLOv8 and by the old program. As can be clearly seen from there, thanks to the ability to detect individual plants, the YOLOv8 model has greatly improved the accuracy of plant counting, especially in hard cases like clumping. Instead of grouping overlapping or adjacent plant cutouts together, the YOLOv8 model learns the exact appearance of plants and detect each of them separately, even when some plants are occluded behind others.

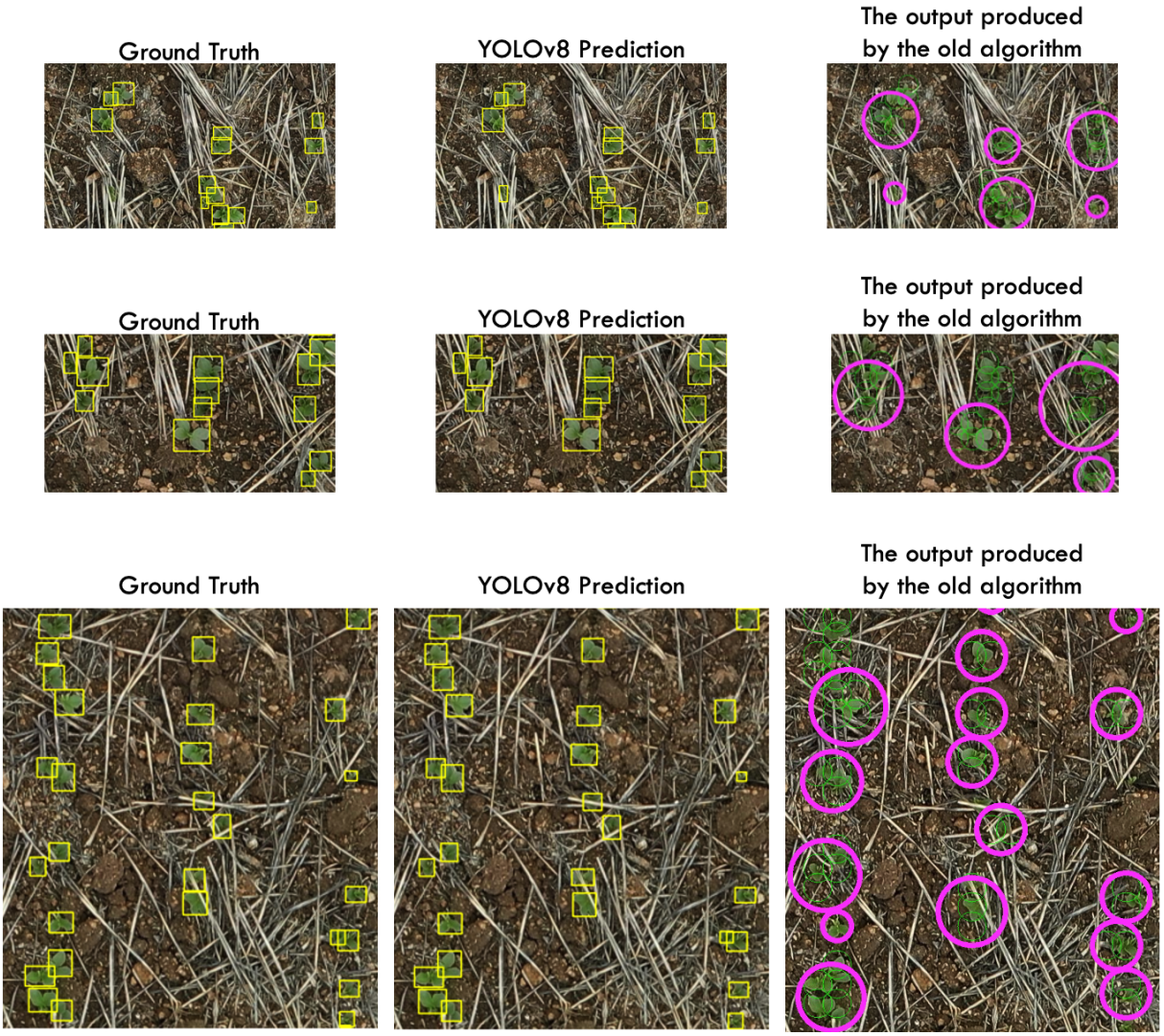

Figure 20: The YOLOv8 model has rectified the clumping issue in the old algorithm, being capable of detecting each individual plant separately, even when they are severely overlapping.

7 Discussions and limitations

7.1 An empirical judgement on the experimental results

In this paper, after five recent versions of YOLO were experimented based on the mAP50 evaluation metric, YOLOv8 was nominated as the most effective model for detecting early emergence of faba beans in plot images. Although the YOLO-NAS model was released after YOLOv8, YOLOv8 still showed a better performance on this faba beans dataset. It implies that a YOLO version released earlier does not necessarily mean it would perform worse than a later-released version. This finding is in alignment with a comprehensive review made by Terven & Cordova-Esparza (2023), who also concluded that YOLO models do not focus solely on improving detection accuracy over time, and thus the performance score of a YOLO version might be lower than a previously developed version. However, their findings simultaneously support and contradict the model comparison results in this

paper to a small extent because the authors found that YOLOv8 also performed better than YOLO-NAS but YOLOv7 even outperformed all the other versions on one of the most popular large-scale object-detection datasets called Microsoft Common Objects in Context (MS COCO) (Terven & Cordova-Esparza, 2023).

For the faba beans experiment, the YOLOv8 model has shown an extraordinary performance in both aspects of plant detection and plant counting with a testing mAP50 of 94.6% and a plant-count RMSE of only 6.261. From the experimental results, YOLOv8 could help tackle all of the four problems that happened with the old program **peatear.py**. Firstly, with bounding-box annotations, all the detected plants are localised, or the boundary of each object detected are specified both visually on the plot image and numerically with the coordinate values. The annotations for each plant no longer contains a lot of background like before. Secondly, not being affected by the methods of sliding window and grouping overlapping objects anymore, the new program built with YOLOv8 can now return much more accurate plant counts to better assist in the analysis of early seedling emergence. Thirdly, together with the elimination of the sliding window technique, the single-stage mechanism of this YOLO model has also resolved the issue with computational time and memory. With YOLOv8, users can now just spend 4% of the time it would have taken them with the old algorithm on processing a full-size plot image, while still being reassured with the model's quality of plant counting. Finally, unlike the old approach that can only set up the training of a CNN model for binary classifications and has to resort to the sliding window technique for generating the outputs, YOLOv8 and other YOLO models in general can be optimised end-to-end, or the model can learn both classification and localisation of emerged plants at the same time (Redmon et al., 2016).

7.2 Limitations

In spite of the robust performance obtained by the YOLOv8 model, there are several limitations with the experiments of YOLO in this paper. Due to the more complex architecture constructed internally, YOLO models often require more time to be trained than a simple CNN for classification purpose only like in the **peatear.py** program. During the code execution for the five YOLO models, it was recorded that their training process often took 20 minutes on a Tesla T4 GPU device, while the training of the CNN model in the old program took almost the same amount of time on just a CPU device (model training on GPU devices was claimed to be at least four to five times faster than on CPU devices, according to Buber & Diri (2018)). Nevertheless, this large training time is completely compensated by much faster processing speed in the deployment stage when a trained YOLO model is applied to producing plant counts full-size plot images, so faster deployment is totally worthy of the sacrifice of greater training time.

Another constraint that should be taken into consideration is that the preparation of bounding box annotations took a colossal amount of time, despite the help from the Roboflow software. If users want good-quality data to train object-detection models, they have to manually draw the ground-truth bounding boxes as precisely as possible to facilitate the model training, so the task of placing bounding-box annotations on plot

images would occupy the majority of time to get the data ready for experimentation. Also, under the scope of this project, the YOLO models were trained entirely on plot images captured on the third day of drone-flying, the emerged plants in these plot images might have notable morphological differences from the first day and the second day, such as different sizes and shapes. For this reason, if a user desires to implement YOLOv8 across all three days of drone-flying in the faba beans experiment, they would need to retrain the model with images and annotations on Day 1 and Day 2; otherwise, they can enlarge the dataset with more plot images taken on these two days.

Besides, the preprocessing step of tiling images is aimed at shrinking down the dimension of an input to around 1000 pixels, so diving a target image into 12 tiles was pertinent to the current size of full plot images. If there is any change in the arrangement of plots in the crop field, the plot size on images could be altered, and the model will probably need to be trained again with different preprocessing configurations modified according to the new plot size. YOLO models were said to be unsuitable for detecting small objects, which is the case of plant detection in this faba beans experiment, so if the drone is to be flown at a higher altitude and the size of emerged plants is made smaller, the YOLOv8 model might need to be enhanced or further data-preprocessing steps might need to be taken.

8 Conclusion

Computer vision and computational image processing methods are expected to aid plant breeders and agronomists in many agricultural tasks, including the monitoring of seedling emergence across large crop fields through UAV-enabled photography. In this work, object detection was sought to make some improvements to a workflow implemented by the Biometry Hub for identifying early emergence of faba beans in aerial plot images, by enhancing their toolsets to perform both object classification and localisation. For this purpose, as one of the most sophisticated frameworks in object detection, multiple versions of YOLO models were experimented and compared on a set of faba beans plot images captured aerially. The initial goal with this paper has been accomplished, after training the five candidate models and comparing their validation mAP50 scores, YOLOv8 was chosen as the optimal model for spotting emerged plants in plot images with a testing mAP50 of 94.6% and a testing plant-count RMSE of 6.261. Compared with the old algorithm, the new program built upon YOLOv8 was able to achieve 76.9% higher accuracy in plant counting with prediction errors being no more than 10 plants for each plot.

Not only does the nominated model show an excellent performance in detection and counting accuracy but it also proves to be more computationally efficient than the old program. The implementation of YOLOv8 is expected to address all the major issues with the old algorithm, especially the localisation of plants growing in clumps and the accessibility to an end-to-end trainable algorithm. Despite some drawbacks, the experimental results in this paper might potentially have certain research significance and bring some practical benefits to the implementation of YOLO and object detection models in the domain of counting and detecting seedling emergence for faba beans cultivation.

Acknowledgements

By completing this report, I wish to express my immense gratitude to my supervisors, Dr. Olena Kravchuk, Peter Kasprzak, and Russell Edson for stimulating my interest in this research direction and for providing supportive guidance and helpful feedback along the way. Over two trimesters, they have encouraged me to follow my research idea with confidence, helped me to address any technical issues arising during my project, and provided me with valuable advice about how to conduct the experiments and how to improve my final products. Peter Kasprzak has my deepest thanks for giving me pastoral supervision, for directly reviewing my preliminary work, and for pitching me through the project goals and all the relevant aspects of the faba beans experiment, including the workflow and toolsets currently implemented at the Biometry Hub.

Also, I would like to extend my great appreciation to the Grains Research and Development Corporation (GRDC) for sponsoring the faba beans experiments as well as the Analytics for the Australian Grains Industry (AAGI) for sharing the data of crop photos with me. This research project could not be made possible without the availability of those plot images. A special recognition also goes to the drone operators who set up drone-flying sessions, I understand that taking those aerial images required a lot of time, effort, and investments to guarantee high-resolution good-quality photos.

Appendices

Appendix A - Supplementary data and the code script

All the plot images in the faba beans dataset are the property of the AAGI and the Biometry Hub, any request to access the data should be sent to them. For the purpose of validating and verifying the results in this paper, the code scripts to run the experiments are made available and could be found on this Github repository: https://github.com/minhnguyen061200/faba_beans.git

Appendix B - The technical design of the five YOLO models experimented

Below are the illustrations of each YOLO model that has been experimented in this paper, as provided by Terven & Cordova-Esparza (2023). The YOLOX, YOLOv6, YOLOv7, YOLOv8 and YOLO-NAS model architectures are presented in Figure 21, Figure 22, Figure 23, Figure 24, Figure 25 respectively.

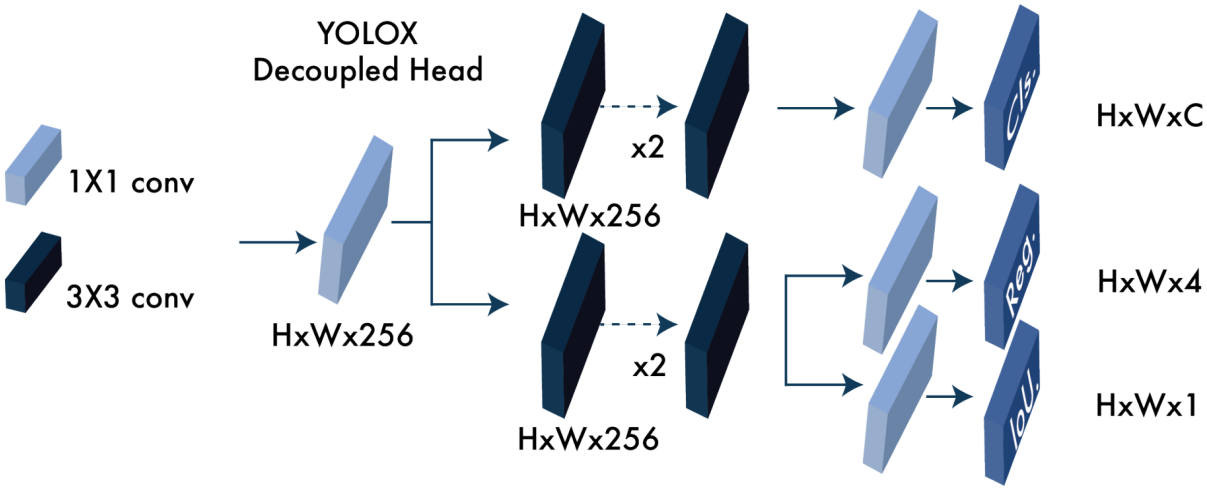

Figure 21: The YOLOX model architecture (Terven & Cordova-Esparza, 2023)

8 Conclusion

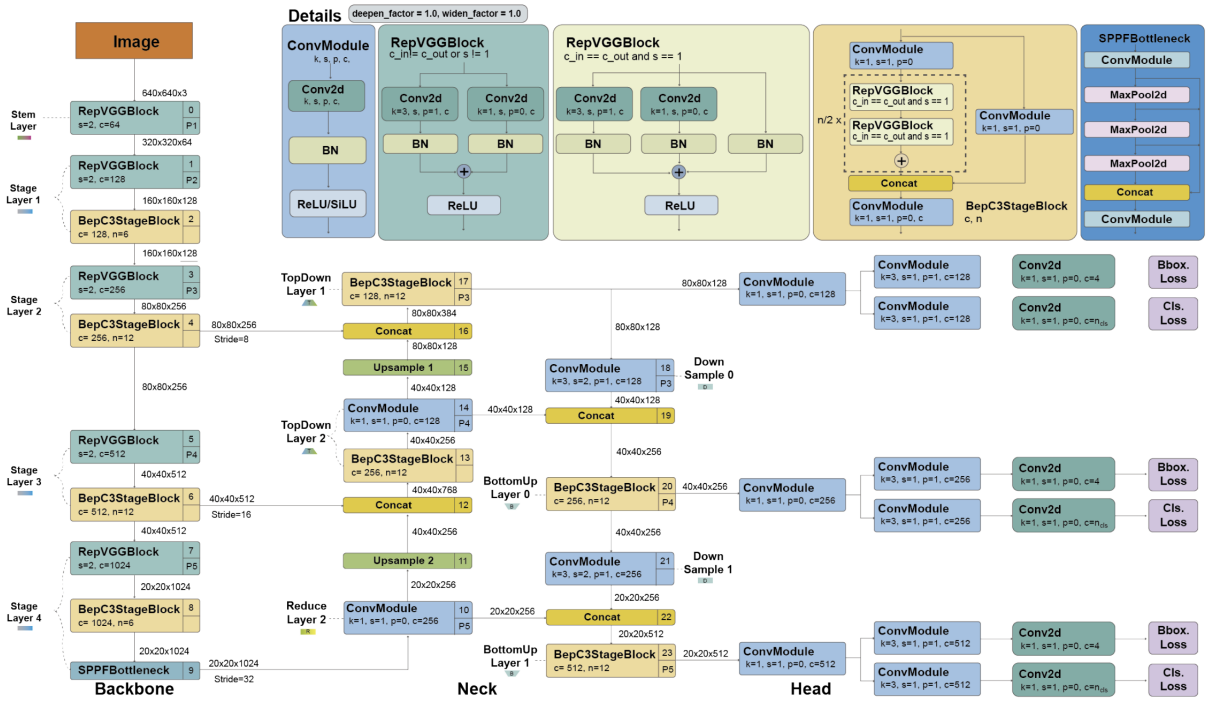

Figure 22: The YOLOv6 model architecture (Terven & Cordova-Esparza, 2023)

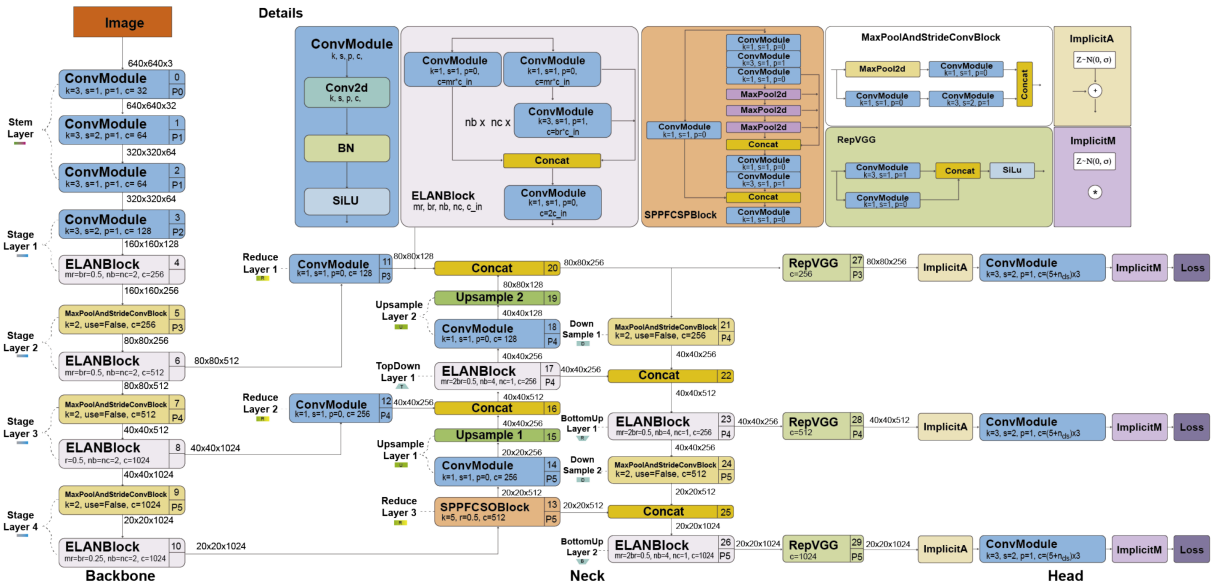

Figure 23: The YOLOv7 model architecture (Terven & Cordova-Esparza, 2023)

8 Conclusion

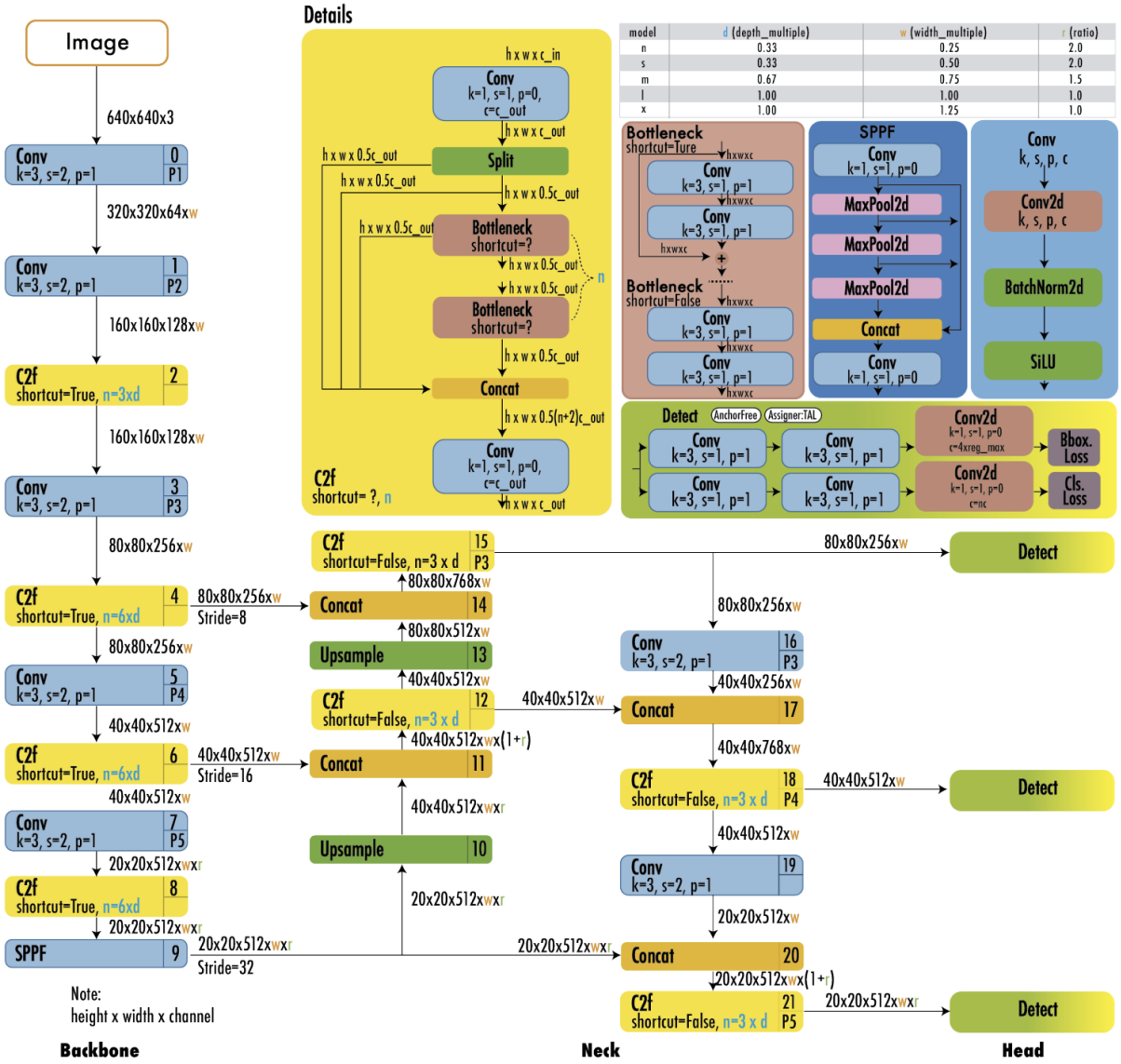

Figure 24: The YOLOv8 model architecture (Terven & Cordova-Esparza, 2023)

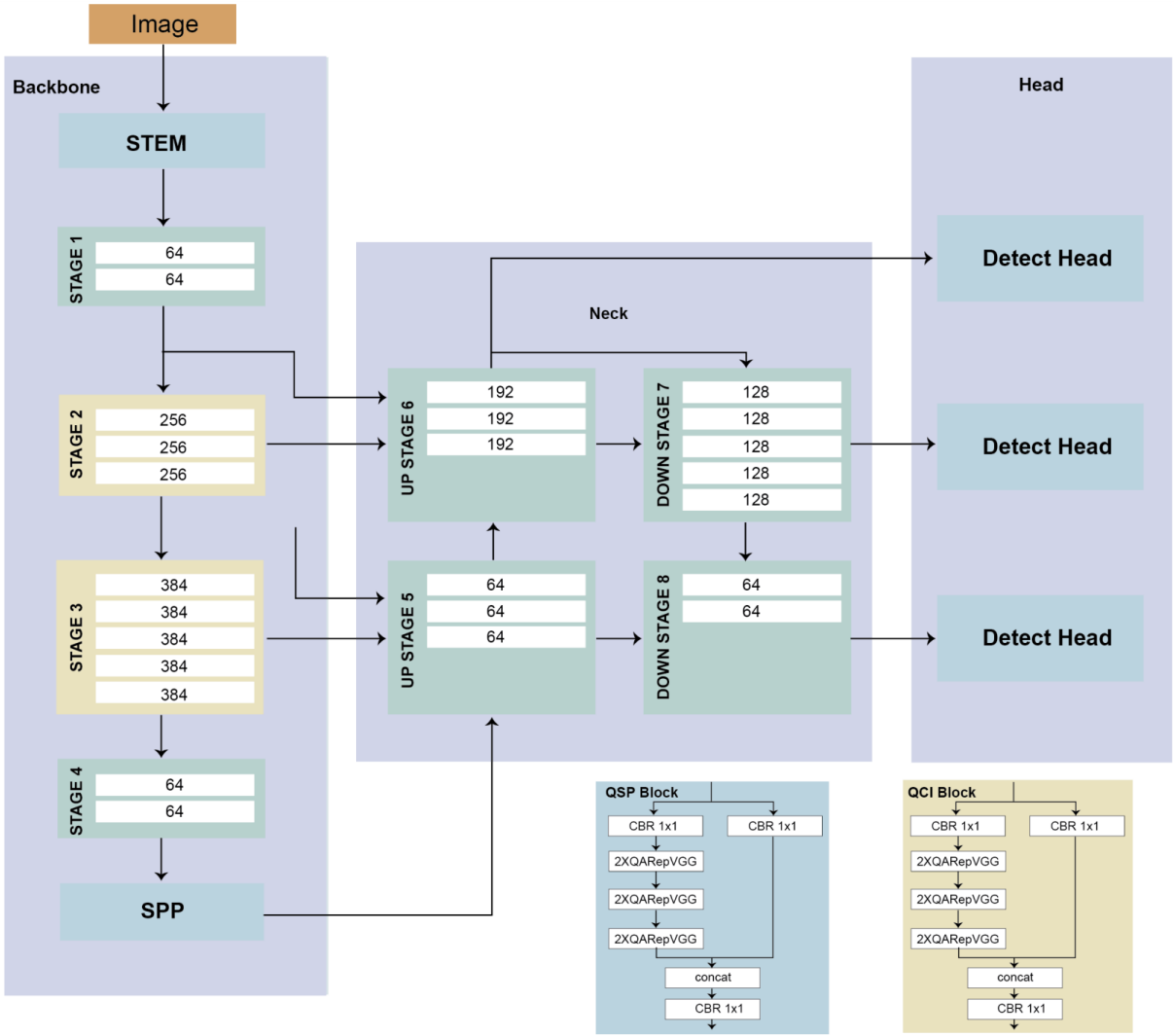

Figure 25: The YOLO-NAS model architecture (Terven & Cordova-Esparza, 2023)

References

10 Aharon, S., Louis-Dupont, Ofri Masad, Yurkova, K., Lotem Fridman, Lkdci, Khvedchenya, E., Rubin, R., Bagrov, N., Tymchenko, B., Keren, T., Zhilko, A. & Eran-Deci. (2021) Super-gradients. *GitHub repository.*

Buber, E. & Diri, B. (2018) Performance analysis and CPU vs GPU comparison for deep learning. *2018 6th international conference on control engineering & information technology (CEIT 2018)* pp. 13–18. IEEE, Istanbul, Turkey.

Chen, J., Wang, Z., Wu, J., Hu, Q., Zhao, C., Tan, C., Teng, L. & Luo, T. (2021) An improved Yolov3 based on dual path network for cherry tomatoes detection. *Journal of Food Process Engineering*, **44**, article e13803.

Das, A., Choudhury, S., Das, A., Samal, A. & Awada, T. (2023) EmergeNet: A novel deep-learning based ensemble segmentation model for emergence timing detection of coleoptile. *Frontiers in Plant Science*, **14**, article 1084778.

Deng, R., Tao, M., Huang, X., Bangura, K., Jiang, Q., Jiang, Y. & Qi, L. (2021) Automated counting grains on the rice panicle based on deep learning method. *Sensors*, **21**, article 281.

Domhan, T., Springenberg, J.T. & Hutter, F. (2015) Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. *Proceedings of the 24th international conference on artificial intelligence* IJCAI'15. (ed M.W. Qiang Yang), pp. 3460–3468. AAAI Press, Buenos Aires, Argentina.

Everingham, M., Van Gool, L., Williams, C.K., Winn, J. & Zisserman, A. (2010) The pascal visual object classes (VOC) challenge. *International journal of computer vision*, **88**, 303–338.

Ge, Z., Liu, S., Wang, F., Li, Z. & Sun, J. (2021) YOLOX: Exceeding YOLO series in 2021. *ArXiv*, **abs/2107.08430**.

Giusti, A., Cireşan, D.C., Masci, J., Gambardella, L.M. & Schmidhuber, J. (2013) Fast image scanning with deep max-pooling convolutional neural networks. *2013 IEEE international conference on image processing (ICIP 2013)* pp. 4034–4038. IEEE, Melbourne, Australia.

Gouk, H.G.R. & Blake, A.M. (2014) Fast sliding window classification with convolutional neural networks. *IVCNZ '14: Proceedings of the 29th international conference on image and vision computing new zealand* pp. 114–118. Association for Computing Machinery, Hamilton, New Zealand.

Hamidon, M.H. & Ahamed, T. (2023) Detection of defective lettuce seedlings grown in an indoor environment under different lighting conditions using deep learning algorithms. *Sensors*, **23**, article 5790.

Hasan, M.M., Chopin, J., Laga, H. & Miklavcic, S. (2018) Detection and analysis of wheat spikes using convolutional neural networks. *Plant Methods*, **14**, article 100.

Hashemi, M. (2019) Enlarging smaller images before inputting into convolutional neural network: Zero-padding vs. interpolation. *Journal of Big Data*, **6**, article 98.

Helwan, A. & Ozsahin, D.U. (2017) Sliding window based machine learning system for the left ventricle localization in MR cardiac images. *Applied Computational Intelligence and Soft Computing*, **2017**, article 3048181.

Henderson, P. & Ferrari, V. (2017) End-to-end training of object class detectors for mean average precision. *Computer vision - ACCV 2016* pp. 198–213. Springer Cham, Taipei, Taiwan.

Jiang, P., Ergu, D., Liu, F., Cai, Y. & Ma, B. (2022) A review of yolo algorithm developments. *Procedia Computer Science*, **199**, 1066–1073.

Jiang, Y. & Li, C. (2020) Convolutional neural networks for image-based high-throughput plant phenotyping: A review. *Plant Phenomics*, **2020**, article 4152816.

Jierula, A., Wang, S., OH, T.-M. & Wang, P. (2021) Study on accuracy metrics for evaluating the predictions of damage locations in deep piles using artificial neural networks with acoustic emission data. *Applied Sciences*, **11**, article 2314.

Jin, X., Liu, S., Baret, F., Hemerlé, M. & Comar, A. (2017) Estimates of plant density of wheat crops at emergence from very low altitude UAV imagery. *Remote Sensing of Environment*, **198**, 105–114.

Jocher, G., Chaurasia, A. & Qiu, J. (2023) Ultralytics YOLOv8.

Lamichhane, J.R., Messéan, A. & Ricci, P. (2019) Research and innovation priorities as defined by the ecophyto plan to address current crop protection transformation challenges in france. *Advances in agronomy* (ed D.L. Sparks), pp. 81–152. Academic Press, Cambridge, Massachusetts.

Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., Li, Y., Zhang, B., Liang, Y., Zhou, L., Xu, X., Chu, X., Wei, X. & Wei, X. (2022) YOLOv6: A single-stage object detection framework for industrial applications. *arXiv*.

Lin, Y., Chen, T., Liu, S., Cai, Y., Shi, H., Zheng, D., Lan, Y., Yue, X. & Zhang, L. (2022) Quick and accurate monitoring peanut seedlings emergence rate through UAV video and deep learning. *Computers and Electronics in Agriculture*, **197**, article 106938.

Liu, Y., Cen, C., Che, Y., Ke, R., Ma, Y. & Ma, Y. (2020) Detection of maize tassels from UAV RGB imagery with faster r-CNN. *Remote Sensing*, **12**, article 338.

Lu, D., Ye, J., Wang, Y. & Yu, Z. (2023) Plant detection and counting: Enhancing precision agriculture in UAV and general scenes. *IEEE Access*, **11**, 116196–116205.

Lv, J., Xu, H., Han, Y., Lu, W., Xu, L., Rong, H., Yang, B., Zou, L. & Ma, Z. (2022) A visual identification method for the apple growth forms in the orchard. *Computers and Electronics in Agriculture*, **197**, article 106954.

Nawaz, S.A., Li, J., Bhatti, U., Muhammad Usman, S. & Raza, M. (2022) AI-based object detection latest trends in remote sensing, multimedia and agriculture applications. *Frontiers in Plant Science*, **13**, article 1041514.

Oh, S., Chang, A., Ashapure, A., Jung, J., Dube, N., Maeda, M., Gonzalez, D. & Landivar, J. (2020) Plant counting of cotton from UAS imagery using deep learning-based object detection framework. *Remote Sensing*, **12**, article 2981.

Oksuz, K., Cam, B.C., Akbas, E. & Kalkan, S. (2018) Localization recall precision (LRP): A new performance metric for object detection. *Computer vision - ECCV 2018* pp. 521–537. Springer Cham, Munich, Germany.

Padilla, R., Netto, S. & Silva, E. da. (2020) A survey on performance metrics for object-detection algorithms. *2020 international conference on systems, signals and image processing (IWSSIP)* pp. 237–242. IEEE, Niteroi, Brazil.

Padilla, R., Passos, W.L., Dias, T.L.B., Netto, S.L. & da Silva, E.A.B. (2021) A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, **10**, article 279.

Plastiras, G., Kyrkou, C. & Theocharides, T. (2018) Efficient ConvNet-based object detection for unmanned aerial vehicles by selective tile processing. *Proceedings of the 12th international conference on distributed smart cameras* ICDSC '18. pp. article 3. Association for Computing Machinery, New York, NY, USA.

Pu, H., Chen, X., Yang, Y., Tang, R., Luo, J., Wang, Y. & Mu, J. (2023) Tassel-YOLO: A new high-precision and real-time method for maize tassel detection and counting based on UAV aerial images. *Drones*, **7**, article 492.

Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (2016) You only look once: Unified, real-time object detection. *Proceedings of the 29th IEEE conference on computer vision and pattern recognition (CVPR 2016)* pp. 779–788. Las Vegas, Nevada.

Sanaeifar, A., Guindo, M.L., Bakhshipour, A., Fazayeli, H., Li, X. & Yang, C. (2023) Advancing precision agriculture: The potential of deep learning for cereal plant head detection. *Computers and Electronics in Agriculture*, **209**, article 107875.

Suo, R., Gao, F., Zhou, Z., Fu, L., Song, Z., Dhupia, J., Li, R. & Cui, Y. (2021) Improved multi-classes kiwifruit detection in orchard to avoid collisions during robotic picking. *Computers and Electronics in Agriculture*, **182**, article 106052.

Szegedy, C., Toshev, A. & Erhan, D. (2013) Deep neural networks for object detection. *NIPS'13: Proceedings of the 26th international conference on neural information processing systems* (eds C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani & K.Q. Weinberger), pp. 2553–2561. Curran Associates Inc., Lake Tahoe, Nevada.

Tan, C., Li, C., He, D. & Song, H. (2022) Towards real-time tracking and counting of seedlings with a one-stage detector and optical flow. *Computers and Electronics in Agriculture*, **193**, article 106683.

Terven, J. & Cordova-Esparza, D.-M. (2023) A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond.

Ünel, F.Ö., Özkalayci, B.O. & Çiğla, C. (2019) The power of tiling for small object detection. *2019 IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW)* pp. 582–591. IEEE, Long Beach, California.

van der Merwe, D., Burchfield, D., Witt, T., Price, K. & Sharda, A. (2020) Drones in agriculture. *Advances in agronomy* (ed D.L. Sparks), pp. 1–30. Academic Press, Cambridge, Massachusetts.

Verdú, M. & Traveset, A. (2005) Early emergence enhances plant fitness: A phylogenetically controlled meta-analysis. *Ecology*, **86**, 1385–1394.

Verschae, R. & Ruiz-del-Solar, J. (2015) Object detection: Current and future directions. *Frontiers in Robotics and AI*, **2**, article 29.

Wang, C.-Y., Bochkovskiy, A. & Liao, H.-Y. (2022) YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors.

Wosner, O., Farjon, G. & Bar-Hillel, A. (2021) Object detection in agricultural contexts: A multiple resolution benchmark and comparison to human. *Computers and Electronics in Agriculture*, **189**, article 106404.

Xiang, S., Wang, S., Xu, M., Wang, W. & Liu, W. (2023) YOLO POD: A fast and accurate multi-task model for dense soybean pod counting. *Plant Methods*, **19**, article 8.

Zhang, X., Zhu, D. & Wen, R. (2023) SwinT-YOLO: Detection of densely distributed maize tassels in remote sensing images. *Computers and Electronics in Agriculture*, **210**, article 107905.

Zhao, Z.-Q., Zheng, P., Xu, S.-T. & Wu, X. (2019) Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, **30**, 3212–3232.