

Chapter 3

Methodology

3.1 Convoluted Neural Network Background Context

Neural networks are one of the most common and successful types of machine learning algorithms in development today (Batta, 2020). They imitate natural structures of nodes and weighted links, utilizing sequential decisions to extract features from image inputs. Convolutional Neural Networks use an architecture consisting of multiple layers (hence the general classification as "deep learning" models), and convolutional kernels that enhance feature extraction. This allows them to achieve accuracy similar to human decisions on image classification tasks (Sharma, Jain and Mishra, 2018). Using a labelled dataset enables supervised learning, where errors in classification between the predictions calculated for each epoch and the actual labelled solutions backpropagates throughout the network and is used to improve the model.

In order to successfully train a supervised CNN in an agricultural context, a large dataset of labelled weed instances has to be collected. Though in general the greater the number of instances available for training a DL algorithm, the better the performance achieved, a benchmark number of 1000 has been successfully used (Olsen *et al.*, 2019). Therefore, 1000 instances was set as the minimum target for each angle class.

YOLOv5 Small¹ was chosen as the CNN of choice due to its high performance, versatility, and ease of deployment. The "small" version of the network utilizes less

¹ YOLOv5 available at <https://ultralytics.com/yolov5>

parameters, allowing for faster calculation time. This would be desirable for real-time deployment in an agricultural context, using either a small robotic platform or a large commercial machine operating at speed to locate and remove weeds.

Finally, though machine learning algorithms are becoming increasingly versatile and transferrable to new conditions and image contexts, the aim throughout the setup and testing of this project was to closely imitate a real-world scenario. This determined the plot setup, camera rig design and hardware, image labelling standards and result interpretation.

3.2 Plot Setup

The dataset collected consisted of wild radish (*Raphanus raphanistrum*) growing in a crop of winter wheat (*Triticum aestivum vulgare*).

At the University of Sydney Lansdowne research farm, Camden, four 80x2m plots were prepared using a pre-sowing knock down spray and cultivation pass. Winter wheat was sown at an industry standard rate of 80kg/ha into each plot at 2–3-week intervals from early March to late May, depending on weather and machinery availability. A tractor-drawn seeding rig set up similarly to commercial seeding units was used, with a row spacing of 25cm.

Previously collected wild radish seed was threshed from the dried seed pods, and hand sown into trays of potting mix. These were watered and fertilized and allowed to grow for 1-2 weeks in a hot house. Approximately 100 radish seedlings were sown at each interval into 3-4 trays within 3 days of each wheat plot being sown.

When wild radish seedlings had reached the two true leaf stage they were planted into the wheat plots, establishing similar growth stages between the wheat and wild radish populations. The wild radish seedlings were planted inter- and intra-row with the wheat at an even split, wild radish density being from 1-5 plants. m^{-1} . Radish plants were selectively "watered in" using a watering can after transplanting. Numbers of transplanted radish plants were 80-100 per time-of-sowing.

The plots were in some cases hand weeded to remove any excessive numbers of broadleaf weeds to reduce the complexity and similarity of the background class. In general, more than 90% of non-radish weeds were removed from the wheat plots. They were then fertilized in accordance with standard broadacre cropping practice, and primarily relied on natural rain but were irrigated to some extent when required.

Typical wheat plot and radish dispersion collection conditions are shown in Figure 1:

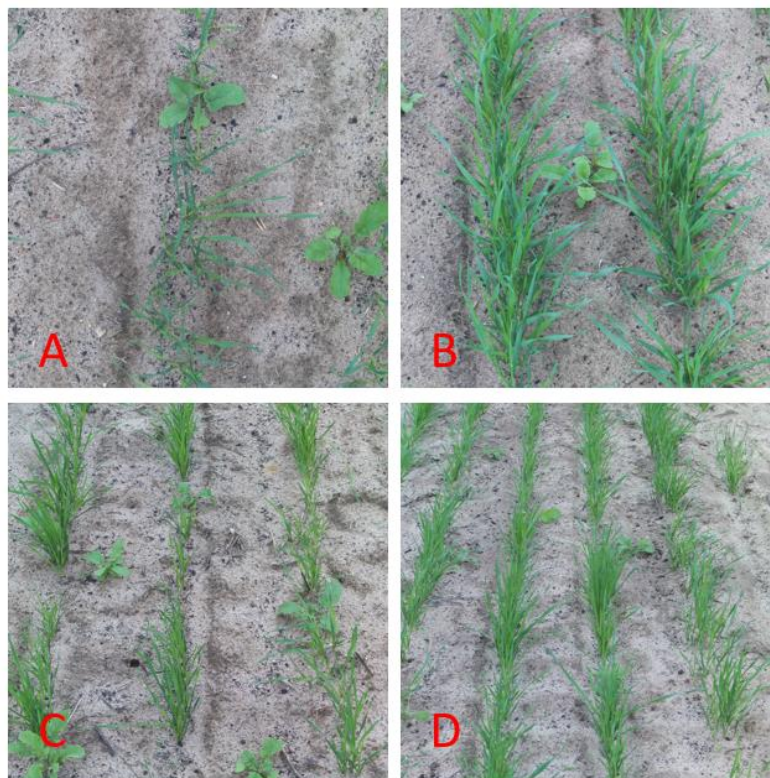


Figure 1 Radish plants growing in the wheat plots with varying camera angles of: A) 0 degrees, B) 30 degrees, C) 45 degrees, D) 60 degrees.

Depicted are ideal image collection conditions of some cloud cover giving minimal over-saturation, similar crop and weed growth stages, and minimal obscuration by crop rows or other weeds.

3.3 Camera Apparatus Design

The primary requirement while designing the data collection platform was repeatability in camera angle and height. A tripod-based camera mount was used as the camera angle could be precisely controlled by using a spirit-level on the main frame of the body, and height was easily set and fixed prior to image collection.

The secondary requirement was for the dataset to be compatible and transferrable for use in future commercial machinery mounted camera weed detection units. For this reason, a camera mount height of 1.25m was chosen, as this is compatible with normal boom operation heights for commercial spray rigs.

The image collection rig was required to collect images that avoid issues such as over-exposure or poor depth-of-field, that would limit the useful information that a deep-learning algorithm process could extract. Working in complex outdoor field environments increased these complications, as changes in cloud cover and sun angle occur often and unpredictably. For the purposes of this paper, pre-installed camera software to automatically adjust exposure levels was used, and the size of the dataset minimised the effect of unavoidable outliers. The lens was manually adjusted for each camera angle to be focused on the vegetation just above ground level. Note that the depth of field present in the 45 and 60deg camera angles made this manual adjustment hard to judge, but the images obtained proved satisfactory.

A Raspberry Pi High Quality Camera 12.3MP 1/2.3" sensor was used because of its high resolution and good accessibility and affordability. An Edmund Optics UC Series Lens with a 12mm focal length was chosen, giving a 660x500mm field of view (FOV) at 1.25m. A lens with a shorter focal length could have been used to give a larger FOV, but this would have increased image distortion. According to the specifications, the 12mm lens chosen gave a distortion of <2.5% which was deemed a suitable trade-off.

An Inca i330G tripod was modified to adapt to a custom designed 3D-printed plastic extension beam, allowing the camera to be mounted 375mm forward of the tripod centre keeping the tripod legs out of the FOV. A custom 3D-printed spring-loaded camera mount was attached to the end of the extension beam, allowing the camera angle to be easily and precisely set to 0, 30, 45 or 60 degrees.



Figure 2 Tripod Camera Rig

Figure 1 shows the tripod camera rig setup. A Raspberry Pi 4B 4GB microprocessor was used to run the RPi HQ Cam and store the collected JPEG files to a USB thumb-

drive. The image resolution was set to 2400x2400 pixels, and while this is larger than used in most image processing algorithms, the quality can easily be scaled down for specific computational requirements. A custom-written Graphical User Interface (GUI) WeedLogger program was written in Python® to provide a good user interface via a Raspberry Pi 7" LCD touch screen. A text file was set up to allow the user to provide a description of the field and weather conditions for each data collection session. The microprocessor and touch screen were housed in a custom-designed 3D-printed mount screwed onto the extension beam. Power for the apparatus was provided via an Anker PowerCore® 13000mAh 3A power bank.

Code for the GUI WeedLogger program is available at the author's GitHub repository.²

3.4 Image collection

Image collection was undertaken at approximately 2-week intervals from May through July, whenever weather and plant growth-stage conditions were suitable. Ten total data collection sessions were undertaken, resulting in approximately 40 hours of field work.

The general process was as follows: ensuring camera rig power bank is fully charged, extend tripod to working 1.25m lens height, remove lens dust covers, insert USB drive and power up the RPi by connecting power bank. Start the collection program and enter field location, crop, weed and weather conditions. Set the camera angle then use preview functionality to verify lens focus. During the session ensure to keep camera aligned with crop rows and level with reference to the spirit level on the side, minimise FOV overlap between individual radish plants, and refocus lens between changing

² Personal GitHub repository available at <https://github.com/RaynerG/RadishDataset>

camera angles. To conclude a session, exit the program, power down the RPi, remove the USB, replace the lens cover and collapse the tripod.

Care was taken to only collect images in conditions that a commercial machine would be operating in the field. This meant that extremes in weather and lighting were avoided, and image collection times generally took place between 10am and 4pm. Excessively sunny conditions were avoided as they tended to cause over-saturation, which is undesirable due to decreased pixel-wise variation leading to less usable information in each image.

3.5 Data Management and Pre-processing

The images were given a naming convention according to the following format:

`crop_weed_angle_yyyymmdd_hh-mm-ss.jpg`

In the case of this dataset, every crop parameter was "wheat", and every weed parameter was "radish". The angle parameter was set with each change in camera angle by the user. The date and timestamps were extracted from the RPi system clock.

The raw images were sorted for duplicates, overlaps, blurring or other issues. Then bounding boxes were drawn around the radish instances using a standard labelling software.³ The resulting XML Pascal VOC format files were converted to YOLO label format TXT files. Note that 4 classes were set according to growth stage, but they were merged into a single class for the main training. These classes were set as cotyledons to 1 true leaf, 2 to 4 true leaves, 5 to 8 true leaves, 9 or more true leaves. Further study should be done to determine the optimum way of splitting the classes according to growth stage in order to further improve classification accuracy. Another potentially

³ LabelImg, available at <https://github.com/tzutalin/labelImg>

important consideration is addressing the best way to label a radish instance on the edge of the image frame or partially obscured by the wheat rows, leading to only a few leaves being visible. In this dataset, these cases were labelled by deducing the probable growth stage (as detailed above) and labelling the visible leaves accordingly. Better results may be obtained by labelling single radish leaves as a unique class, but that is beyond the scope of this project.

The images were resized from 2400p to 1280p square and PNG format, in order to be directly compatible with the YOLOv5 training requirements. However, 640p was used in initial training with good success, and a range of other resolutions could be easily adapted to due to the flexibility of the YOLO label format.

After resizing and labelling the images, these PNG and TXT files had paired file names according to collection date. Using a custom python script, each pair of files was given a randomised number as its unique identifier, sorted, and placed into a dataset folder. These folders were set up to be the standard 60 / 20 / 20 percentage split between training, testing and validation subsets. Randomisation was used to give a good variation between the different conditions and growth stages seen in the different collection dates. Also, the subset splits were counted according to number of instances rather than number of images, as some variation exists in the quantity of weeds present in each image.

3.6 Training the Algorithm

Due to the high processing power required for full-scale training of a CNN on an image dataset, Amazon Web Service® (AWS) cloud computing resources were utilized. A Linux® Deep Learning (DL) machine was used, using Ubuntu® 18.04. The instance

type was g3s.xlarge with 1 GPU and 4 vCPUs, 30.5 GiB of memory and 8GiB of GPU memory. These comparatively lower specifications were deemed sufficient to train YOLOv5 Small, because it is a light algorithm. The running instance was then connected to using PuTTY® SSH (Secure Shell) key login ⁴, and Linux Screen ⁵ was used to allow the training process to be run in a detached state from the terminal, ensuring against closed SSH connection or other interferences.

The datasets were uploaded to an AWS S3 bucket, then linked to the DL instance. Training results were also linked back to the S3 bucket for download to a local machine.

The latest version of YOLOv5 was cloned from GitHub® ⁶, and the official guide was used to set up the folder structure and choose training parameters. Reference Appendix A for the folder structure, configuration file changes and training command used in this project.

The training resolution for YOLOv5s can be set to 640p or 1280p, in this case the higher resolution was used to minimise any issues radish instances at greater distances in the FOV, particularly relevant for the 60-degree case. Training was set for 200 epochs as this was shown in initial testing to be suitable for maximising the performance without excessive risk of over-fitting. Batches of 8 were used to minimise the possibility of running out of GPU RAM.

⁴ PuTTY available at <https://www.puttysshclient.com/>

⁵ Linux Screen documentation available at <https://www.gnu.org/software/screen/manual/screen.html>

⁶ YOLOv5 models available from <https://github.com/ultralytics/yolov5>

3.7 Process for Evaluation of Results

The models were initially assessed based on visual observations of their performance, and then by numerical analysis of their performance metrics.

3.7.1 Visual Performance Comparison

The visual performance comparison will be completed by running a series of unseen test images through the model for each camera angle, using the best achieved weights from the training. These test images will be chosen from a variety of plant growth stages, lighting, and background conditions.

Though minimal empirical data can be drawn from this comparison, it serves as a useful demonstration of the results, and allows observation of factors that may influence further training strategies or implementation on field machines.

3.7.2 Numerical Performance Comparison

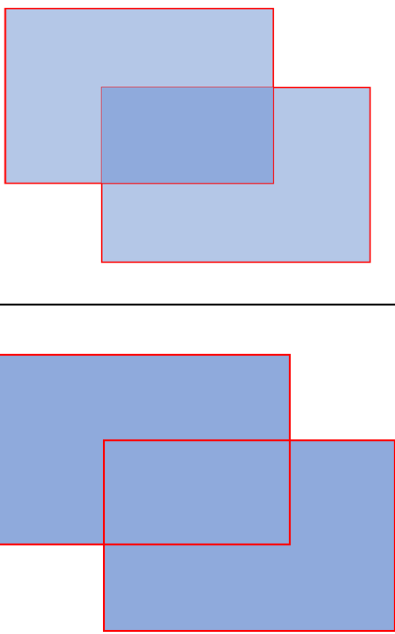
When numerically evaluating the results of a Deep Learning algorithm's predictions, there are a number of key parameters to consider. These are based upon the following matrix:

Table 1 Prediction vs Actual matrix

		Actual Class	
		1	0
Predicted Class	1	True Positive	False Positive
	0	False Negative	True Negative

In Table 1, a "1" designates a "true", or "target detected", result. Conversely, a "0" indicates a "false" result. The distinctions between predicted and actual classes can therefore be defined as true or false positives, and false or true negatives.

To first calculate whether an object that is located in the image frame by the algorithm is a true positive or a false positive, the Intersection over Union (IOU) must first be calculated. This is a threshold value, and is calculated as shown below:



$$IOU = \frac{\text{Intersecting area of predicted bounding box and labelled bounding box}}{\text{Total area of predicted bounding box and labelled bounding box}}$$

Figure 3 Calculation of the Intersection over Union

A threshold value is chosen and results over that value are set as true positives, whereas any result under the threshold indicates a false positive. If a labelled bounding box is present and there is overlapping predicted bounding box, a false negative result is given. True negative is a trivial case for any part of the image that has neither labelled nor predicted bounding boxes.

From here, the definitions of Precision and Recall can be set:

$$Precision = \frac{\text{True Positives}}{\# \text{ Predicted Positives}} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \quad (1)$$

$$Recall = \frac{True\ Positives}{\# Actual\ Positives} = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (2)$$

Precision is a comparison between the number of results predicted as positive vs the number of results that are actually positive, as shown in Eq. 1. A high decision threshold for this parameter should be set when an important decision needs to be made on when very confident, for example in producing a definitive medical diagnosis. However, this may result in some actual positives being bypassed as a negative by the algorithm.

On the other hand, recall is a comparison between the number of results that are actually positive vs the number of results correctly predicted as positive, given in Eq. 2. A high decision threshold for this parameter should be set when it is more important to minimise missed detections, i.e., avoid false negatives. As the trade-off, some actual negatives may be incorrectly identified as positive.

As previously noted in Chapter 2, broadacre cropping fields often have a small number of weeds such as radish distributed among many crop plants, in this case being wheat. The ability to identify and remove as many weeds as possible without worrying about some wheat being damaged in the process is considered the preferred performance for a green-on-green SSWC system. Therefore, high recall scores should be preferred when assessing evaluating the training results for a weed detection algorithm. For some high value per unit agricultural crops may prefer a high precision score to avoid damage.

However, a widely used standard for evaluating the performance of a machine learning algorithm is the mean Average Precision (mAP), or in the case of a single-class detection task, the Average Precision (AP). The models for each camera angle trained in this project will be compared accordingly. The standard used will be MS COCO mAP0.5:0.95, which is calculated during the training of YOLOv5. This averages the AP over IOU thresholds between 0.5 and 0.95 (Bell *et al.*, 2016).

To calculate the AP, the *Precision · Recall* pairs are first interpolated to become monotonic to give $P_{interp}(R)$. Then using the reference threshold values R , the formula for calculating AP is as follows (Padilla *et al.*, 2021):

$$AP = \sum_{k=0}^K (R(k) - R(k+1)) \cdot P_{interp}(R(k)) \quad (3)$$

Where K is the number of *Precision · Recall* pairs. As mentioned previously, mAP and AP are identical for a single class object detection task.

A determination of the best performing camera angle model was made by comparing the highest achieved precision, recall and the best mean Average Precision.