Faculty of Engineering

Segmentation and Classification of Weeds

Final Research Report submitted for EGH400-2 Semester 2 2022

Student Name:	Isaac TAN
Supervisor:	Jasmine Banks, Khamael Al Dulaimi
Submission Date:	28 October 2022

Abstract — The ability to manage invasive weed species is vital to the outcome of crop harvests and the agriculture industry. Advancing software systems are creating solutions that help this industry optimise weed management so that it is less labour intensive, chemically less harmful to the environment, and more profitable. This report investigates the use of such software systems, convolutional neural networks (CNNs), for the task of automatically segmenting and detecting the Rumex obtusifolius weed species amongst grass. The motivation for this is that the software system will be used in the future to automatically detect and precision spray these weeds. The project followed two main goals, semantically segmenting images of the Rumex weed, and then classifying with localisation (object detection). The semantic segmentation model was built upon the U-Net architecture and the object detection model was built with the Mask R-CNN architecture. The results of this report compared evaluation metrics of each system under different constraints and parameters, to ultimately assess the viability of using CNNs for weed detection. Findings showed that the semantic segmentation system achieved accuracies and IoU scores of 85% and 73.87% respectively while operating at 21 fps. Results from the object detection system revealed a mAP of 52.28% and an F1 score of 54.23% while taking 846 ms per prediction on a 256 x 256-pixel image.

CONTENTS

INTRODUCTION

The vital industry of agriculture continues to evolve and adapt to sustain the demands of the ever-growing population. Farmers and researchers continue to investigate different approaches of optimising agriculture to maximise yield while minimising loss due to ever-present factors like weeds, pests, and natural disasters. Research is now pointing towards machine learning and artificial neural networks (ANNs) to aid the industry and overcome some of these obstacles.

This project investigated the problem of weeds, specifically the *Rumex obtusifolius*, and aimed to construct a solution in the form of a software system to assist in the detection of such weeds. The motivation for this was to provide a proof of concept that the task of treating these weeds could be automated.

The requirement for an automated method of weed treatment stems from a huge annual loss of crop to weeds reported worldwide. Studies published in Australia and India reported losses of up to $5 billion and $11 billion respectively (McLeod, 2018; Talaviya, Shah, Patel, Yagnik, & Shah, 2020). Furthermore, current weed management methods are cost inefficient, labour intensive and damaging to the environment (Hlaing & Khaing, 2014). Automated weed management systems that use precision spraying of specific herbicides on weeds theoretically solve all the above issues. These automated systems can be divided into the software detection system, and the physical task of spraying the weeds. This report solely focuses on detecting and classifying the weeds.

To this end, this report analysed the contemporary literature of machine learning, ANNs and their current application to the agriculture industry relating to weeds. It delves into the method used in this project to both semantically segment and classify objects in images of the Rumex weed. It aimed to prove the viability of an automated weed management system attached to an unmanned ground vehicle (UGV) or similar device.

For this proposed UGV to be viable in terms of the detection system, two main conditions were assessed:

- *What is the accuracy and precision of a detection system when detecting Rumex amongst grass?*
- *Can this detection system be deployed at a high enough speed to be able to run in real-time?*

The report further discussed the results of the implementations in this project with regard to these two research questions. Finally, the risks of such a software system are examined concerning ethics and sustainability within the agriculture industry.

1 LITERATURE REVIEW

Prior to development of the software system, current approaches to weed management and the technologies proposed for such a task were thoroughly investigated. This review delves into the workings of artificial neural networks and investigates different approaches and applications of them. This section of the report will also review the use of ANNs in similar problem spaces. The literature explored in this review will be contained to sources written in relevant agricultural, computer science or engineering journals or articles from January 2012 onwards, though older sources are referenced to compare methods. These sources will be obtained through online databases such as IEEE Xplore, Scopus and Inspec.

1.1 MOTIVATION

Current methods of weed treatment include tedious manual labour of hand picking or spraying with a targeted herbicide, and broader spraying an entire field with a general herbicide. Failure to eliminate a weed with broader spraying causes the species to adapt and become resistant to the herbicide (Kamath, Balachandra, Vardhan, & Maheshwari, 2022). Additionally, Australian research shows that herbicides used on sugarcane farms wash off the farm and into rivers and finally the Great Barrier Reef, polluting the natural wonder (Sugar Research Australia, 2022). Automating weed treatment to precision spray volumes of targeted herbicide proportional to the amount of weed helps reduce the negative impacts of weeds.

The motivation of applying artificial neural networks to this particular problem is drawn from an abundance of literature showing the potential for detection and classification tasks, but lack of literature pertaining to the Rumex obtusifolius.

1.2 ARTIFICIAL NEURAL NETWORKS OVERVIEW

The technology that shows the most potential in identifying weeds is machine learning and artificial neural networks. Artificial neural networks (ANNs) are algorithms mimicking the biological structure of the human brain (Wen, Yihui, Ying, & Jiaxu, 2022). They make decisions based upon the output of a stream of connections between neurons from an input. Neurons are separated into layers and connected to one another through weight, bias, and an activation function. Information is passed between layers with the weights applied and assessed under the activation function. If the output of the activation function meets the bias or threshold, then the data passed forward to the next layer. This process repeats until the output layer, at which the ANN arrives at a decision (Dave & Dutta, 2014). A simplified diagram of this can be seen in Figure 1.1.

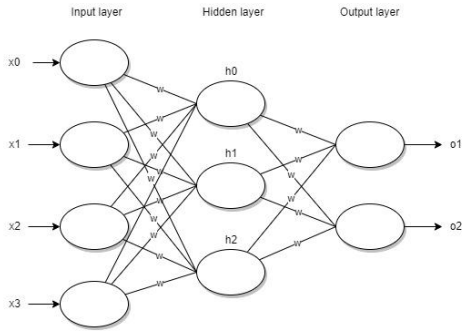


Figure 1.1 Neural Network Diagram. Adapted from *(Shiruru, 2016)*

Neural networks are made more advanced and complex as more layers are added to their architecture, forming deep neural networks (DNNs) (O'Mahony, et al., 2019). A subclass of DNNs, convolutional neural networks (CNNs) are particularly useful for pattern recognition in images. CNNs feature a convolutional filter which when repeated creates a feature map of the input (Grace, Anitha, Sivaramakrishnan, & Sivakumari, 2021). This feature map removes the manual task of feature extraction and applying the weights which differentiates ANNs from DNNs. Manually performing feature extraction and applying the weights between neurons is called supervised learning and creating convolutions that automatically adjust these weights is termed unsupervised learning (Abiodun, et al., 2018).

The goal of both learning types is to arrive at predictions closer to the expected outcome or known truth with each iteration of training, known as an epoch. This distance between

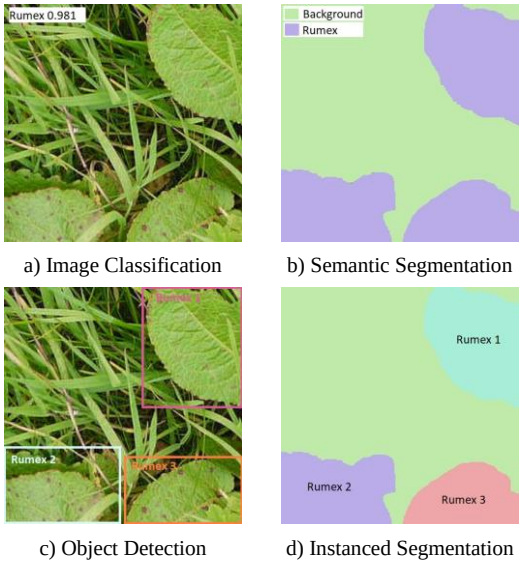
predicted outcome and expected outcome or ground truth is measured using a loss function and with each step of training, the algorithm adjusts its feature weights to reduce the loss. As this process is repeated the algorithm eventually optimises the loss function and increases accuracy (Yu, Wang, Zou, & Wang, 2020). Convolutions create linear functions of the output of the previous layer and feed it forward, eventually reaching a final activation function that map results according to the function type. During the training phase, the CNN is constantly updating feature weights to minimise the value loss. However, if the model is trained for too long, it over optimises feature weights specifically for the training dataset causing something called overfitting. This means that when the model is evaluated on the testing dataset, the overall accuracy is lowered as the model has been trained to perform well specifically for the training dataset (Narasinga Rao, Venkatesh Prasad, Sai Teja, Zindavali, & Phanindra Reddy, 2018).

Neural networks can be used to complete different functions of image processing. These functions can be classed into four main groups: image classification, semantic segmentation, object detection and instanced segmentation. Image classification describes an entire image by one label or class. Its output is a vector the length of the number of classes used in its training with corresponding prediction probabilities (Yamashita, Nishio, Do, & Togashi, 2018).

Semantic segmentation involves describing every pixel in an image as a label from a list of classes (Sheikh, et al., 2020). In the case of this project the two classes were Rumex and background. The output of the CNN is a feature map the same dimensions as the input image x the number of classes (Liu, Deng, & Yang, 2018).

Object detection is an extension of image classification that detects multiple occurrences of objects in a photo and classifies them as separate instances. The output of these CNNs can vary depending on the image and the architecture used, but ultimately describes an object by a class label and a corresponding bounding box outlining its localisation.

Finally, combining aspects of both detection and segmentation, instanced segmentation describes pixels as a label class as before, but maintains occurrences and labels them as separately (Vayssade, Jones, Gée, & Paoli, 2022). Figure 1.2 shows example outputs of the four objectives described above.



a) Image Classification b) Semantic Segmentation

c) Object Detection d) Instanced Segmentation

Figure 1.2 Image processing functions: a) Image Classification, b) Semantic Segmentation, c) Object Detection, d) Instance Segmentation. Adapted from (Hafiz & Bhat, 2020)

The neural networks used in this project focus on semantic segmentation and object detection. Each work slightly different from one another and have different use cases.

1.3 Semantic Segmentation

Binary classification like the classes of Rumex and not-Rumex (background) in this project is achieved by mapping the final output of a CNN as a number between 0 and 1 through a SoftMax activation function where numbers closer to 1 indicate higher probability that the prediction is Rumex. However, for semantic segmentation, which creates a mask this final output must be either 0 or 1. As such a sigmoid activation function is used as the last layer of the network to fit the decision to these values (Li, Yang, Peng, & Zhou, 2021).

1.4 Object Detection

Unlike the method and architecture used by semantic segmentation networks, which are fully connected CNNs followed by either a sigmoid or SoftMax activation, object detection requires a different approach. Since the number of occurrences of an object varies with each image, the object detection output length varies. To account for this and detect multiple instances of an object in a single image, images are divided into regions and input into a classic CNN. There are multiple implementations of this method.

Rudimentary approaches involved taking equal sized windows out of the original image and applying a CNN to classify each window (Rowley, Baluja, & Kanade, 1998). The problem with this was that objects may span over different windows or have aspect ratios, so the success of the system was largely dependent on the number of windows analysed per image. As the window size decreased and the number of windows increased, so did the computational cost of running such a system.

R-CNN, a solution to this problem was proposed by Girshick, et al, where a selective search algorithm is used to extract 2000 regions from an image (Girshick, Donahue, Darrell, & Malik, 2014). These regions were warped to be equally sized, and a CNN then classified each region. While state of the art at the time of proposal, there were still some downfalls. The large number of regions, often overlapped, meaning that the same pixels were passed through the system which repeated computations. R-CNN was reported to take up to 2.5 days to train on the PASCAL VOC07 training set, and took 47 seconds per image to test, both tasks completed with a GPU (Girshick, Fast R-CNN, 2015).

Girshick solved these issues with his proposal of Fast R-CNN. Fast R-CNN optimised training and testing time by generating a convolutional feature map from the input with a CNN. This feature map is used to propose regions of interest and then pass them forward to a region of interest (RoI) pooling layer. This extracts a fixed-length feature vector from the feature map which is passed to a sequence of fully connected layers and then to two output layers. One output layer produces a probability of an object and its class, and the other outputs four numbers corresponding to the bounding box positions of the object (Girshick, Fast R-CNN, 2015). The reason this is faster than R-CNN is the convolutional operation is only done once per image instead of once for each of the 2000 proposed regions in R-CNN. This change was reported to increase speed by a factor of 213 at test time and took only 2.5 hours to train.

Ren et al further improved Fast R-CNN with Faster R-CNN. Faster R-CNN followed a similar process to Fast R-CNN but replaced the selective search algorithm used to extract regions of interest, with a separate network that predicted region proposals. The region proposal network (RPN) takes inputs from the convolutional feature map feeds them through a series of fully connected layers to two output layers, outputting proposed regions to the classifier (Ren, He, Girshick, & Sun, 2015). These improvements meant that around 300 region proposals were made per image on the VOC07 dataset and allowed Faster R-CNN to perform at 5 fps on a GPU. This enables real-time object detection.

Extending from Faster R-CNN, He et al proposed Mask R-CNN which added a parallel branch for predicting object masks in addition to bounding boxes. This adds a very small computational expense with the benefit of more complex output information. The system is still reported to run at 5 fps (He, Gkioxari, Dollár, & Girshick, 2017). He et al suggested that this mask information be used for estimating human poses, however in the case of this project and its future work, this showed potential for estimating the positioning of the Rumex weed and locating its centre or where its roots are relative to the leaves.

The original backbone CNN architecture for Fast R-CNN and thereby Mask R-CNN was VGG-16 (Girshick, Fast R-CNN, 2015). VGG-

16 is a 16-layer deep CNN that aimed to increase depth therefore improving performance on very small convolutional filters (Zisserman & Simonyan, 2015). However, He et al compares the Mask R-CNN performance with both VGG-16 and a new proposed residual net, ResNet101. They report that the significant increase in depth from 16 layers to 101 layers while still having lower complexity allows Mask R-CNN to perform with a mean average precision (mAP) increase of 7.3% on the Microsoft Common Objects in Context (MS COCO) dataset (He, Zhang, Ren, & Sun, 2016).

These findings influenced an implementation of Mask R-CNN by Matterport which builds an easy-to-use library for Python on Keras and TensorFlow (Abdulla, 2018). This implementation encourages the use of transfer learning with the pre-trained weights MS COCO dataset.

Transfer learning is the application of previously learned knowledge to new scenarios. In the case of CNNs, it is taking pre-trained weights on a dataset of large volume and using them to train a CNN on a smaller dataset (Weiss, Khoshgoftaar, & Wang, 2016). This is beneficial as it reduces training time, expense of manually annotating data, and improves performance. (Sonntag, et al., 2017; Yang, et al., 2022)

1.5 CASE STUDIES

Before the software systems were created for this project, case studies on similar projects were investigated.

When classifying tobacco crop and weeds researchers compared region and grid-based CNNs using R-CNN and YOLOv5 respectively. The study published in Applied Sciences found that the R-CNN resulted in a 98% accuracy and the system using YOLOv5 yielded a 94% accuracy. Additionally, it stated that integrating a system that automatically detected and precision sprayed weeds decreased the use of herbicide by 52% (Alam, et al., 2022).

Another study published in the International Conference on Artificial Intelligence and Smart Systems (ICAIS) compared a series of CNNs to other decision-making algorithms with a varied segmentation and feature extraction techniques to classify crops and weeds. It found that the CNNs were more effective than the other machine learning algorithms and that of the CNNs focusing on segmentation, SegNet and U-Net were the most accurate with results of 95%+ (Veeragandham & Santhi, 2021). It also detailed the common challenges of classifying weed species using automated systems such as the varying light conditions. The study states that these problems can be overcome with different image pre-processing techniques.

Delving deeper into Segnet and U-Net, a study published in IEEE International Conference on Signal, Information and Data Processing (ICSIDP) investigated an application of image segmentation for the blind and compares the results of SegNet and U-Net. Their method found that for segmentation, U-Net achieved the better result of 84.32% accuracy against 70.54% with SegNet. Additionally, their system was fast, taking less than 0.5 seconds to segment any random image (Liu, Wang, & Zhao, 2019).

Additionally, Ronneberger et al. who first introduced U-Net, states that for segmentation, U-Net achieves very good performance on varying images when they investigated its use in neuronal structures and electron-microscopic recordings. They found that it achieved 92.03% accuracy taking less than one second to segment a 512x512 pixel image. (Ronneberger, Fischer, & Brox, 2015). The appeal of using U-Net was that it concatenates the features from encoding-decoding with higher resolution features from before max pooling. This allowed for contextual awareness when up-scaling.

Focusing specifically on Rumex obtusifolius, work by Kounalakis et al investigates using predefined spatial regions to perform object detection. They used an unmanned ground vehicle to capture images of the weed at a constant distance. Results reported precision between 4.73% and 59.27% with recall ranging 82.97% down to 23.81%. They found that when

changing parameters to balance their dataset and make improvements in precision came at the cost of recall. Additionally, their detection system did not run in real-time, disqualifying its use in this project (Kounalakis, Malinowski, Chelini, Triantafyllidis, & Lazaros, 2018).

Another report on Rumex by Lam et al uses an unmanned aerial vehicle to capture images of the weed at between 10 - 20 meters. Their proposed method achieved accuracies of up to 92.1% with F1 scores of 78.7%, however did not operate in real-time. (Lam, et al., 2021).

Finally, one of the pioneering studies into detection of Rumex by van Evert et al uses Fast Fourier Transform and thresholding to detect weeds. Their model achieved accuracies of up to 89% (van Evert, et al., 2010).

There is a gap in the existing literature on the detection of Rumex. Current studies use detection methods that have significant processing times, so make use of the information by processing it externally and then constructing a flight or vehicle path for a UAV or UGV such that it can traverse it and treat the weeds. However, detection methods on other plant species and outside of the field of agriculture showed potential for real-time image detection that can be processed onboard a UAV or UGV. As such this project aimed to implement some of these proposed methods on the Rumex obtusifolius.

2 METHOD

2.1 GENERAL APPROACH:

The approach to the project was to construct two different CNN models, one for semantic segmentation, and another that applied a RPN for instanced classification of weeds. The general method of image segmentation and classification follows five steps: image acquisition, image pre-processing, image segmentation, feature extraction and classification (Vasavi, Punitha, & Rao, 2022). However, for this project, deep neural networks were used thus removing the need for manual feature extraction.

The segmentation software system was developed in Python 3.10.0 with TensorFlow 2.10.0. and the Keras 2.10.0 API to interface.

The object detection software was developed in Python 3.7.13 with TensorFlow 1.15.3, Keras 2.2.4 and the Matterport implementation of the Mask R-CNN library (Abdulla, 2018). The justification for using different versions of Python and libraries is discussed later in *2.6 Risks.*

2.2 ACQUISITION AND PRE-PROCESSING

The images used for this project were acquired as part of a dataset of Rumex obtusifolius with a grass background that was used by Van Evert, et al (van Evert, Polder, van der Heijden, & Kempenaar, 2009).

Typically, images in a dataset are of different sizes and are randomly cropped and rescaled in pre-processing to a pre-determined size so that all images are of equal shapes. However, this dataset was comprised of 540 square images all 256x256 pixels. The images were randomly separated into training and testing images at a ratio of 75% training and 25% testing. The training dataset was then split further to allow 10% of it to be used as validation. To decrease the volume of data and thus decrease training and testing time, images were downscaled in the semantic model. The effect of this is discussed in *3.1 Semantic Segmentation*.

The literature review revealed that one obstacle to automated classification is light conditions varying throughout the day. This is often alleviated with image pre-processing with varied light conditions. However, since the scope of this project is contained to the images in the dataset, which were all taken under similar lighting conditions, this step was purposefully omitted here for sake of the project timeline. Future works may implement some form of light condition pre-processing.

The dataset also included annotations of the images which contained bounding box information of Rumex locations in the images.

The structure and hierarchy of the annotation file can be seen found in the appendix Figure 5.1 where the bounding box information has been highlighted.

A Python program was created that parsed all the annotation files, located bounding box information, and created a mask image with the information. An example output of this Python program can be seen on the right of Figure 2.1 next to its corresponding dataset image. For visualisation, the mask output has been made red for boxes containing Rumex and left green for areas containing grass.

Figure 2.1 Image from dataset (left), output mask from Python program (right)

2.3 SEGMENTATION

The semantic segmentation neural network was built upon the U-Net architecture. This was so that it took features at a higher resolution and applied them with deeper convolutions. The architecture took an input frame and applied two 3x3 convolutions followed by a 2x2 max pooling operation. This process was repeated for a total of 4 times with the number of feature channels of the convolutions doubling after each max pooling. An up-sampling filter of 2x2 was then applied and concatenated with the feature map of the same size from before the max pooling operation. This was then followed by two more 3x3 convolutions and the process was repeated for four up-samples. Each filter application used a ReLU activation function. The final filter was a 1x1 convolution with a sigmoid activation function to determine the binary output of weed or not weed. A diagram of this architecture can be seen below in Figure 2.2 where the grey boxes are the feature maps, and the white boxes are the copied and concatenated feature maps from before the max pools.

The 405 images from the training dataset were used to train and validate the neural network on this architecture. The model was then compiled with the Adam optimiser. Since the task was one of binary segmentation, a loss function of binary cross entropy was used when compiling. The model was initialised for training with an arbitrary number of 15 epochs. When fitting the model however, a call back function was created to combat the overfitting problem discussed in the literature review. This call back function monitored the validation loss at the end of each epoch and if the validation loss had not improved for more than 5 epochs then training would be stopped, and the previous feature weights would be restored.

2.3.1 TRAINING

Following training, the model was then evaluated with the testing dataset of 135 images. The results of this are discussed later in *Results, Analysis & Discussion*. To visualise these

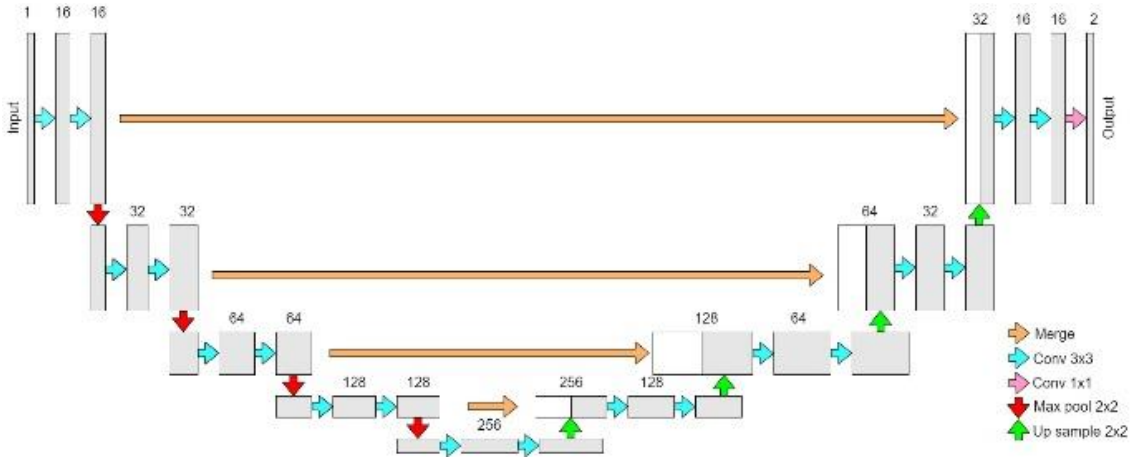


Figure 2.2 U-Net Architecture. Adapted from (Ronneberger, Fischer, & Brox, 2015)

results, the output feature maps for each image were converted back to masks.

The literature review found that over training a model for a specific dataset is a common issue with CNNs. To mitigate this, training was stopped early if the validation loss didn't improve after a number of epochs. This resulted in the model being trained for 9 epochs instead of the proposed 15 that the training was initiated with. This can be seen in Table 2.1.

Table 2.1 Training summary of U-Net which stops training early due to overfitting

Epoch	Loss	Accuracy	Validation Loss	Validation Accuracy
1	0.1374	0.7543	0.1267	0.8038
2	0.0962	0.8529	0.1354	0.7790
3	0.0881	0.8636	0.1320	0.8297
4	0.0771	0.8803	0.1249	0.8341
5	0.0646	0.9013	0.1354	0.8365
6	0.0580	0.9120	0.1455	0.8403
7	0.0449	0.9319	0.1942	0.8337
8	0.0373	0.9443	0.2701	0.8404
9	0.0274	0.9596	0.2746	0.8430

2.4 CLASSIFICATION

The Mask R-CNN model was used to detect and classify multiple occurrences of the Rumex weed from single frames. The extension of Faster R-CNN was chosen as it would output an additional segmentation mask at low computational cost that could be compared with the semantic segmentation mask previously described and give insight and validation of both systems. Furthermore, the literature revealed that it would be possible to run at 10 fps and enable real-time object detection.

The model was built upon a ResNet101 backbone with a feature pyramid network (FPN) and used transfer learning from the MS COCO dataset.

The model applied 7x7 convolutional filters to the input images to obtain feature maps. These feature maps are passed forward as inputs into the ResNet101 backbone which apply 3x3 convolutions with residual 1x1 convolutions as per the official documentation of ResNet101 by He et al (He, Zhang, Ren, & Sun, 2016). The output is then passed into a top down FPN which up scales by a factor of 2 while laterally connecting to the ResNet backbone layers of the same ratio. The layers of the backbone, denoted by {C2, C3, C4, C5} in Figure 2.3, and the pyramid levels are coupled through element-wise addition following a 1x1 convolutional layer. C1 is not used as recommended by the original documentation due to its large memory footprint (Lin, et al., 2017). At each level of the FPN, a 3x3 convolutional filter is applied to create the final feature maps used by the RPN, {P2, P3, P4, P5} mapped from {C2, C3, C4, C5} respectively. Since C1 isn't used in the FPN, a 5^{th} level of pyramid is created by subsampling P5 with a 1x1 MaxPooling layer for 2 strides. This generates P6. The RPN proposes potential bounding boxes from {P2, P3, P4, P5, P6}. These then undergo non-maximum suppression (NMS) to remove proposed RoIs which majorly overlap others with an IoU threshold of 0.3 (Salscheider, 2021). The effect of the NMS is discussed later in discussion. Each proposed RoI undergoes RoI pooling which creates a feature map of fixed size. However, these RoI pooling layers create misalignment between the RoIs and the input image, so an additional RoI align layer is added. This aligns the extracted feature maps with the input image. Output of the RoI align is then passed through one branch to two fully connected layers with a dimension of 1024 and ReLU activation functions, and then to a softmax and linear regression activation function which return the class probability and bounding box respectively. The other branch of RoI output is passed to a fully convolutional network followed by a sigmoid activation function which returns the instanced segmentation mask. Figure 2.3 shows an overview of the model architecture.

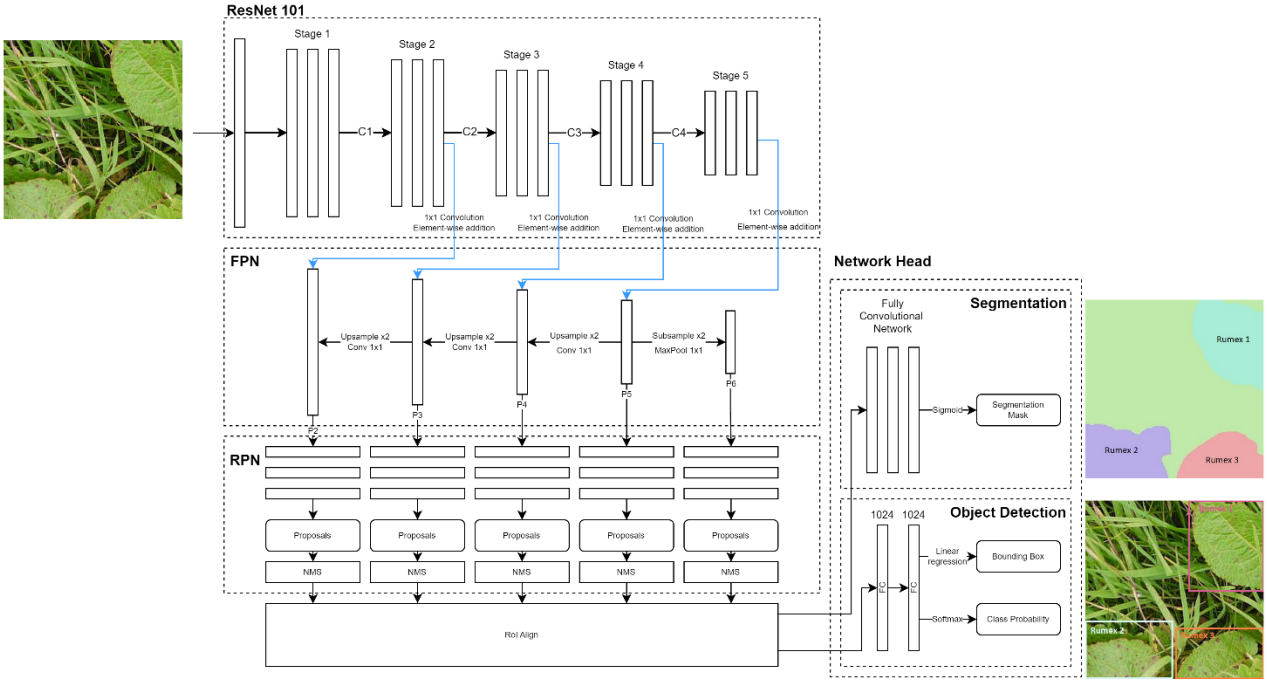


Figure 2.3 Object Detection Model Overview

The model was trained with the same ratio of 75% training to 25% testing used in the semantic segmentation. The stochastic gradient descent (SGD) optimiser was used with a loss function that accounts for class, bounding box and mask losses. This loss function follows the original documentation of Fast R-CNN and Mask R-CNN and is defined in equation (1) (Ren, He, Girshick, & Sun, 2015; He, Gkioxari, Dollár, & Girshick, 2017)

$$Loss = Loss_{class} + Loss_{bbox} + Loss_{mask} \quad (1)$$

Where $Loss_{class}$ is a normalised binary cross-entropy loss function of the of the prediction of p_i and the ground truth p_i^* for an index i of an anchor. This is defined in equation (2) where the function is normalised over the number of classes.

$$\frac{-1}{N_{class}} \sum_i (p_i^* \cdot \log(p_i) + (1 - p_i^*) \cdot \log(1 - p_i)) \quad (2)$$

Similarly, $Loss_{bbox}$ is a normalised robust loss function multiplied by the weight λ. The robust loss function acts as the linear L1 loss function when $|x|$ is greater than or equal to 1 and the 2nd order L2 loss function when below 1. Here, x is the difference of the ground truth (t_i^*) and prediction tuples (t_i) of bounding box coordinates.

$$x = (t_i^* - t_i) \quad (3)$$

$$L_{robust}(x) = \begin{cases} |x| - 0.5, & |x| \geq 1 \\ 0.5x^2, & |x| < 1 \end{cases} \quad (4)$$

$$Loss_{bbox} = \frac{\lambda}{N_{class}} \sum_i p_i^* \cdot L_{robust}(x) \quad (5)$$

The multiplication of p_i^* in (5) disables the loss functions for the background class ($p_i^* = 0$ for background and $p_i^* = 1$ for Rumex). The value λ is used for weighting loss between the three components of the total loss function. For this project, λ has been left as $\lambda = 1$.

Finally, $Loss_{mask}$ is a binary cross-entropy loss of a per-pixel sigmoid function, which is normalised over the number of pixels k.

$$\frac{-1}{k} \sum_i (k_i^* \cdot \log p(k_i) - (1 - k_i^*) \cdot \log(1 - p(k_i)) \quad (6)$$

k_i^* represents the ground truth label of the pixel k, and $p(k_i)$ represents the probability of the predicted class.

2.4.1 TRAINING

The model was trained with transfer learning on pretrained weights from the MS COCO dataset of over 300,000 images with 80 categories. This was done to both improve performance and reduce the training time. The model was run for 10 epochs with a learning rate of 0.001 and learning momentum of 0.9. Similar to the semantic model, weights were restored if the model validation loss increased over a set of epochs. This resulted in the model being trained for 5 epochs instead of the initial 15.

A full list of model training configurations can be found in the appendix. Configurations not discussed in this report are overlooked for the sake of brevity as they match the original documentations and are not fundamental for understanding this particular project (Girshick, Fast R-CNN, 2015; Ren, He, Girshick, & Sun, 2015; He, Gkioxari, Dollár, & Girshick, 2017).

Training was conducted on the same 11th Gen Intel Core i7-1165G7 @ 2.80GHz machine used in the semantic segmentation implementation. Training the model took 16 hours with these specifications. A summary of the training can be seen in Table 2.2.

Table 2.2 Training summary of Mask R-CNN

Epoch	Loss	RPN Class Loss	RPN Bbox Loss	Class Loss	Bbox Loss	Mask Loss
1	1.3505	0.0380	0.4725	0.1443	0.4040	0.2917
val	1.2007	0.0262	0.3675	0.1624	0.6259	0.7627
2	1.3109	0.0728	0.3983	0.1165	0.4018	0.3215
val	1.3818	0.0653	0.4050	0.1507	0.4258	0.3350
3	1.2178	0.0654	0.3765	0.1112	0.3509	0.3138
val	1.4204	0.0685	0.4432	0.1384	0.4478	0.3226
4	1.1448	0.0613	0.3618	0.1053	0.3123	0.3042
val	1.3062	0.0575	0.3926	0.1138	0.4163	0.3260
5	1.0450	0.0566	0.3165	0.0972	0.2805	0.2942
val	1.2661	0.0536	0.3760	0.1196	0.3975	0.3194

2.5 EVALUATION METRICS

The semantic segmentation model was assessed by precision, recall and IoU. The performance of the Mask R-CNN was evaluated on the metrics of mean average precision (mAP), mean average recall (mAR), and F1 score. The mAP was calculated on the MS COCO and PASCAL VOC equations used for their respective competitions. The terms used by the metric equations are extracted from the confusion matrix in Table 2.3

Table 2.3 Confusion Matrix of CNN. Adapted from (Kohavi & Provost, 1998)

		Predicted	
		Positive	Negative
Actual	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

These values of true and false are determined by the Intersect over Union (IoU) of the prediction and the ground truth bounding boxes. IoU is denoted by equation (7) which divides the amount of A_p that intersects A_{gt} by the total area of A_p and A_{gt} (Rezatofighi, et al., 2019). Here A_p is the bounding box prediction, and A_{gt} is the ground truth bounding box.

$$IoU = \frac{A_p \cap A_{gt}}{A_p \cup A_{gt}} \tag{7}$$

If the IoU was greater than or equal to a threshold, the prediction is classed a true positive. If the IoU was less than the threshold the prediction was a false positive. If no ground truth bounding box was detected but the model makes a prediction A, then this was classed as a false negative. True negatives are all areas of the image where an object was not detected. This is not useful for object detection and was ignored.

The precision of the model was calculated by the number of correct detections in an image divided by the total detections in the image seen in equation (8).

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

This precision was then summed over the testing dataset and divided by the number of images to

give the mAP seen in equation (9) where i denotes an image of the test dataset. This is also usually summed for each class and then divided by the number of classes, however given this project focused on binary classification and segmentation and only included Rumex in the dataset, this was not done.

$$mAP = \frac{1}{|tests|} \sum_{i \in tests} \frac{|TP_i|}{|FP_i| + |TP_i|} \quad (9)$$

An IoU threshold of 0.5 is used to evaluate the model on the PASCAL VOC metric (Everingham, Van Gool, Williams, Winn, & Zisserman, 2010). Additionally, if there are multiple detections of the same object, the highest IoU is taken. This mAP is denoted by mAP_{50} and is discussed in the results section. This is compared with a higher IoU = 0.75 denoted by mAP_{75} and the MS COCO metric, $mAP_{.5-.95}$. The MS COCO calculation of $mAP_{.5-.95}$ further averages the mAP from equation (9) by incrementally increasing the IoU threshold used for dictating the confusion matrix by 0.05. and then dividing by the total number of incrementations. This gives an AP across multiple detection confidences.

Recall of the model was calculated by the number of correct detections divided by the total number of ground truths seen in equation (10).

$$Recall = \frac{TP}{TP + FN} \quad (10)$$

This was again summed over the test set and divided by the number of images, giving the mAP seen in equation (11), where i is an image of the dataset.

$$mAR = \frac{1}{|tests|} \sum_{i \in tests} \frac{|TP_i|}{|TP_i| + |FN_i|} \quad (11)$$

F1 score tests a model's predictive accuracy relative to its sensitivity by relating precision and recall. This relation is show in equation (12) where precision is mAP_{50} and recall is mAR.

$$F1 = 2 \times \frac{precison \times recall}{precision + recall} \quad (12)$$

Both models were assessed for latency. The latency of the system was calculated by deducting the time at the start of the predictions from the time at the end of the predictions.

2.6 Risks

As this project was purely in software there were no physical risks to consider. However, there were other risks relating to deliverable dependencies, software dependencies and requirements, and computational resources.

The segmentation and object detection systems were developed in the Python programming language with the TensorFlow library and Keras API to interface. Furthermore, the Mask R-CNN library was used for the object detection system. The software was developed in Python as the student engineer had previously worked with it and was comfortable using it. Additionally, TensorFlow and Keras were chosen as they were well documented and had a large online support community of others that share their troubleshooting tips for issues that arose. This largely mitigated the risk of software not working or difficult troubleshooting problems. For the software to work as intended, the dependent Python libraries were required to import correctly and function as documented. Since the student completed segmentation software first, the most recent software versions were used for this system. However, when constructing the object detection system, the documentation supporting the Mask R-CNN implementation recommended using older versions of software and libraries. To mitigate risks of different libraries not working together or being of an old version, virtual environments withing the computer were used. Libraries and software with specific versions matching the documentation were installed in these virtual environments and ensured that software dependencies were met. Additionally, this allowed both software systems to run on the same computer at the same time if needed.

The overall completion of the software systems had a few dependencies on the completion of smaller software tasks. These tasks were given estimated completion times to maintain an appropriate completion trajectory and assess the overall state of the project. Additionally, weekly, or fortnightly meetings were conducted between the student engineer and the project supervisors to check in and allow for questions to be asked or feedback to be provided.

The risks pertaining to the dataset were limited. There was a small risk that the dataset provided would not contain enough information to complete the task. However, the dataset was analysed upon attainment and proved to be sufficient, especially given the potential to use image augmentation and expand the dataset. Additionally, since the dataset is public and cited by other published articles its risk factor was low.

Computational power was considered for both training and execution of the segmentation and object detection systems. Since training CNNs is often computationally expensive it is recommended to use a machine that has a powerful GPU. Unfortunately, this was not available to the student engineer. To mitigate the risk of the CNNs not training in time due to the lack of GPU, the software was developed well before project deliverable dates and given a long window of time to train. Additionally, training was run overnight when the no other processes were using CPU on the machine. Other services like Amazon Web Services E2C GPUs were considered to remotely run these executions on powerful computers, however they incurred a cost that was viable at the time.

2.7 ETHICS

Throughout this project, the four cores of the Engineers Australia (EA) Code of Ethics were consistently considered.

Integrity was demonstrated through the trustworthy and timely completion of this project and its interim deliverables. Respect for both project supervisors and potential future stakeholders of the project were held in high regard. The availability of the dataset provided to the student engineer was kept private to respect implied confidentiality obligations. Additionally, the project was conducted with a clear conscience knowing that the overall impact would be positive and solve real-world issues. This satisfied EA Codes 1.1, 1.2, 1.3.

To practice competently according to the EA Code of Ethics 2), the student engineer was constantly learning and developing new skills. The in-depth literature review meant that decisions made for this project were well-informed and made based on adequate knowledge. The literature review and development of the software system enabled continued learning as there was no single source that answered all questions. These showed competency in both 2.1 and 2.3.

Leadership was exercised through consistent meetings held between the student engineer and the project supervisors. These meetings were opportunities for the student engineer to effectively and honestly communicate with the stakeholders on progress on the project and request any further resources required of the stakeholders. This aligns with EA code of ethics 3.1 and 3.3.

Sustainability is the pillar of this project and reflects EA codes 4.1, 4.2, 4.3. The project considered public concerns of the high impact of invasive weed species and proposed a solution that is sustainable in economic, environmental, and social impact. The project aimed to provide proof of concept that a system could be implemented that would reduce the cost of weed management and boost profitability, reduce environmental harm due to over-use of herbicides, and reduce manual labour on large farms.

The results of these ethical considerations are discussed later in *3.3 Risks, Ethics and Sustainability*.

2.8 SUSTAINABILITY

Further from the sustainability ethics described above, the sustainability of the project

life cycle was also considered. The project aimed to provide valuable information for future implementations and work regardless of the system viability in regards to the research questions. This report aspired to act as a starting point for any future work such that future ventures could apply the results of this report to the considerations of theirs. To do this, the reported endeavoured to provide as much information regarding the theory, implementation, and results of the project.

3 RESULTS, ANALYSIS & DISCUSSION

The semantic segmentation and object detection systems were evaluated upon the metrics described *2.5 Evaluation Metrics*. Additionally, comparisons were made between the semantic segmentation and the masks extracted through the Mask R-CNN object detection. Both models are assessed for speed in completing their corresponding tasks.

3.1 SEMANTIC SEGMENTATION

The U-Net model was assessed under the criteria of accuracy, precision, recall, IoU and latency to assess its viability in real-time use. Section *2.2 Acquisition and Pre-Processing* discusses how input size could be scaled to reduce computational expense and improve speed. As such, the model was tested with three differently sized input images to compare these metrics. A summary of these results on the testing portion of the dataset is displayed in Table 3.1

Table 3.1 U-Net CNN evaluation results on testing dataset

Input image resolution	Accuracy (%)	Precision	Recall	IoU (%)	Time (ms)
64^2 (1:4)	82.62	82.58	82.69	70.37	**1952.970**
128^2 (1:2)	82.77	82.81	82.70	70.62	2708.358
256^2 (1:1)	**85.00**	**84.87**	**85.19**	**73.87**	6352.281

Table 3.1 shows that the resulting metrics are proportional to the image scale factor. As the image sizes were scaled down from 256^2 pixels, the model's accuracy, precision, recall and IoU also decreased. This was also true for the latency of the system. At the lowest input size of 64^2 pixels, the model operated at its highest speed and took 1.95 seconds to conduct predictions on 135 images, taking 14ms per image. At the highest resolution of input, the system took 6.35 seconds to make predictions on the same set of 135 images, averaging 47ms per image. This speed translates to 21 frames per second.

Preliminary testing of the model used only accuracy and latency, but upon further research, accuracy was not the optimal unit of measurement for segmentation systems. As such, IoU was calculated and is used as the primary assessor for the semantic system. Other metrics are displayed for the purpose of completeness.

IoU results showed slight change in performance across the three input image resolutions. Decreasing the image resolution by 4^2 (64^2 from 256^2) decreased latency by a factor of over 3, while only reducing the performance by 3.45%. Given the goal of this system was to work in real-time, this resolution was the most appropriate to use as it provided relatively accurate predictions and could theoretically operate at 69 fps.

The results of the semantic segmentation system were visualised by converting the output mask to a numerical array and then mapping each element in the array to a pixel of an RGB colour. An example of this can be seen in Figure 3.1 which shows the prediction mask overlayed over the original image. The red indicates a prediction of Rumex, and green a prediction of the background class. The opacity of the prediction has been lowered for viewing such that the image underneath is visible. Figure 1.1 also shows the un-altered image on the right.

Figure 3.1 CNN segmentation predictions of two images. Original images (left) & predictions overlayed with original image (right)

Observing these predictions, it was noticeable that predictions often extended passed the borders of the Rumex. This can be seen in Figure 3.1 where the red area is greater than the area of the Rumex leaf.

Figure 3.2 False positive prediction of Rumex

Furthermore, there were some images of grass with no Rumex that the model predicted to contain Rumex. This can be seen in Figure 3.2. This was likely due to the ground truth labels used to train the CNN. Since truth labels were rectangular in shape and as such were bigger than the Rumex, when the system made training predictions on areas of grass near a Rumex ground truth, they were considered a correct prediction. Optimising the system to reduce its loss function where these areas are considered good predictions trained the model to overestimate.

These ground truths also impact the evaluation metrics described in Table 2.3. These metrics make comparisons between predictions and ground truths, and as such when ground truth is not precise, it skews the evaluation metrics.

The model also struggled with extreme close-up photos of Rumex, evident in Figure 3.3.

Figure 3.3 Semantic Segmentation Prediction on close-up Rumex

Despite the imperfections described above, the overall results of this CNN were promising for this project. The system was able to operate at 69 fps and provided an IoU of 70.3%. However, the overall capability of semantic segmentation is limited. For an automated weed treatment system, more complex information like localisation would be beneficial to its performance.

3.2 Object Detection

The Mask R-CNN model was evaluated on mAP over three IoU conditions, as well as mAR, and F1 score. Model configuration parameters were then altered to investigate their impact on the performance of the system.

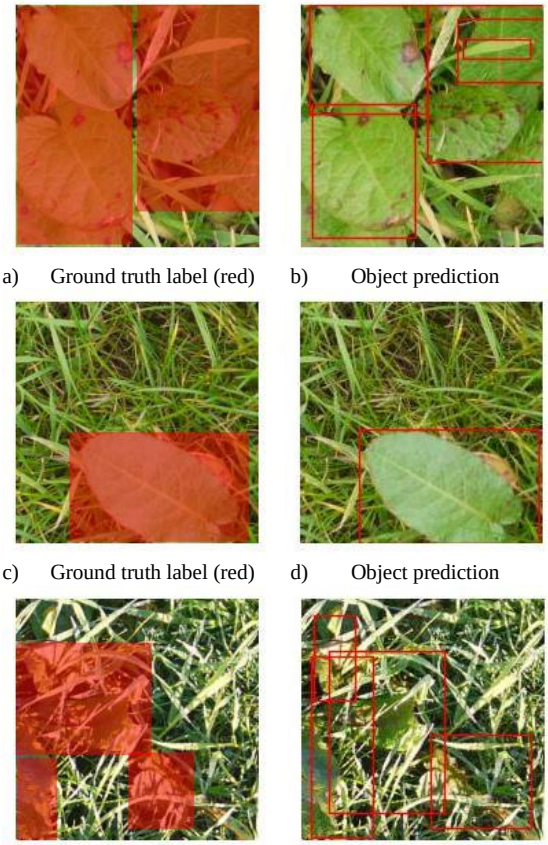
Table 3.2 shows the results of with the default parameters described by (Abdulla, 2018). Both mAR and F1 are both calculated based on IoU threshold of 0.5.

Table 3.2 Mask R-CNN evaluation results with default parameters. All values are percentages (%)

	mAP_{50}	mAP_{75}	$mAP_{.5:.95}$	mAR	F1
Train	54.90	24.09	27.79	62.22	58.33
Test	49.78	17.96	22.85	53.99	51.80

Numerically, this did not appear to show results on par with the results obtained from the semantic segmentation model. The semantic model showed precision of over 70%, contrasting with the object detection model resulting in 49.78% mAP on the testing dataset.

The results were visualised for further comparison before detection parameters were changed. A sample of images with their generated bounding boxes can be seen in Figure 3.4 where the proposed objects are displayed to the right of the original image with its ground truth on the left.



a) Ground truth label (red) b) Object prediction

c) Ground truth label (red) d) Object prediction

e) Ground truth label (red) f) Object prediction

Figure 3.4 Mask R-CNN Predictions on three images

Figure 3.4 d) shows that the model made accurate predictions on images where Rumex was visually isolated from surrounding elements. However, in Figure 3.4 b) and f), the model struggled with images where the Rumex was obstructed either by other Rumex weeds or heavy grass foliage. In Figure 3.4 b) the prediction boxes for two leaves on the left are accurate, however there are multiple predicted bounding boxes that overlap for the foliage on the right. This posed a problem to potential stakeholders in an automated weed treatment system as in these cases the system would detect additional instances of Rumex and as such spray more volume of herbicide to the area. The magnitude of this effect is largely dependent on the cost of herbicide and the environmental impact of applying unnecessary herbicide, which varies between stakeholders and the herbicides they have access to.

To address the flaws in the model, the parameters of detection confidence and detection NMS threshold were altered. The detection threshold was initially decreased from 0.7 to 0.6 with the aim of detecting more instances of Rumex. This resulted in the evaluations seen in Table 3.3. The table shows that decreasing the detection threshold increased the evaluation metrics across the entire system.

However, this change also presented more problems. With a decreased detection threshold, the system was making more predictions that overlapped each other, further perpetuating the problem of over spraying described previously. This can be seen in Figure 3.5 a) which shows an additional two counts of Rumex compared to the Figure 3.4 b). It also presented an issue of predicting areas of grass as Rumex, displayed in Figure 3.5 b).

Table 3.3 Mask R-CNN evaluation results with detection threshold = 0.6

	mAP_{50}	mAP_{75}	$mAP_{.5:.95}$	mAR	F1
Train	56.81	24.62	28.03	64.92	60.60
Test	**52.28**	**18.94**	**23.81**	**56.33**	**54.23**

a) Crowded predictions b) Misidentified grass

Figure 3.5 Poor Predictions of Rumex. Detection threshold = 0.6

During evaluation it was seen that these smaller bounding box predictions that overlapped with larger correct predictions (Figure 3.5 a), and the predictions made over clear grass (Figure 3.5 b) had a confidence of

roughly 0.7. As such, the detection confidence threshold was increased from 0.6 to 0.8.

This negatively affected the evaluation metrics as is evident when comparing Table 3.3 and Table 3.4. However, upon visual inspection, the prediction of objects appeared to be more aligned with correct boundaries. Figure 3.6 a) shows results similar to Figure 3.4 b) with the original parameters, and Figure 3.6 b) shows that the grass has been correctly identified.

Table 3.4 Mask R-CNN evaluation results with detection threshold = 0.8

	mAP_{50}	mAP_{75}	$mAP_{.5:.95}$	mAR	F1
Train	51.88	23.78	26.31	57.06	54.35
Test	48.20	17.65	22.34	51.95	50.00

a) Less crowded predictions b) No false positives

Figure 3.6 Improved Predictions of Rumex Detection Threshold = 0.8

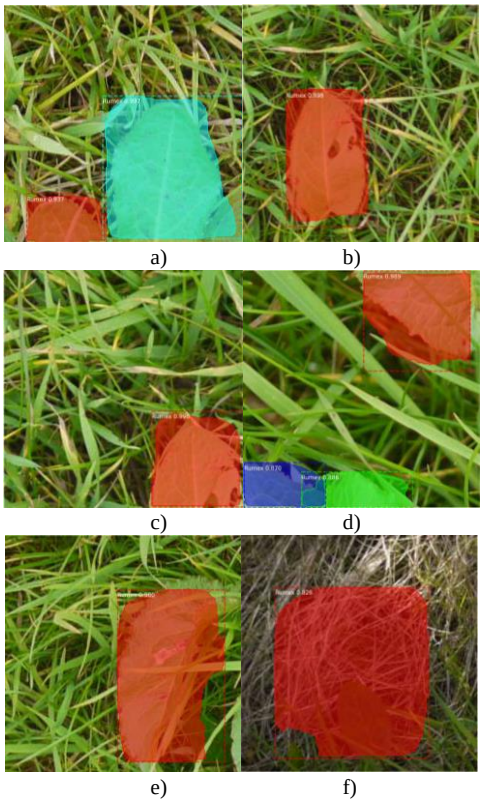
This detection confidence removed false positive detections but did not solve the issue of repeated detections of single objects discovered in the original parameters. To attenuate these, the detection NMS threshold was tuned from 0.3 to 0.2. Evaluation showed that the numerical performance decreased marginally, however visual performance was improved.

Table 3.5 Mask R-CNN evaluation results with detection threshold = 0.8, NMS threshold = 0.2

	mAP_{50}	mAP_{75}	$mAP_{.5:.95}$	mAR	F1
Train	51.93	23.72	26.32	56.63	54.18
Test	48.07	17.65	22.33	51.95	49.94

Further decreasing this threshold did omit more duplications of objects, however, it also removed correct predictions of the Rumex weed. In relation to the potential stakeholders of an automated weed management system, it was deemed more beneficial to overspray weeds at the cost of herbicide, than to fail to spray some weeds. As such this parameter was left as 0.2.

Using these parameters, the model was then used to extract instanced segmentation masks to compare to the results previously discussed in the semantic segmentation system. A sample of six images is displayed in Figure 3.7.



a) b)

c) d)

e) f)

Figure 3.7 Mask R-CNN Mask Predictions

The model appeared to make accurate mask predictions on most images, seen in Figure 3.7 a) – e). Similar to the semantic segmentation model, the predictions do extend slightly further from the bounds of the Rumex leaf. However, the degree of over estimation appears to be smaller on the Mask R-CNN predictions than that of the U-Net predictions.

The Mask R-CNN model did make some errors in predictions. The model failed to detect shadows in areas of dry, dead grass. This is evident in Figure 3.7 f) where the model has predicted that the shadow cast above the Rumex as a weed. Similar to the U-Net model, the Mask R-CNN model struggled to make accurate predictions on images that were extreme close-ups of the Rumex.

Despite these flaws, the parameters defined above appeared to return reasonable results of precision without over compromising recall. As such, the latency was measured with these parameters. The system took 114.29 seconds to make predictions on the testing dataset of 135 images. This speed averages 846 ms per image or 1.18 fps.

This was significantly lower than speeds reported in the literature review which ranged from 5 fps to 10 fps. This was likely due to the computational ability of the machine used to run the model. During initial meetings with the stakeholders, it was stated by the research supervisor that no additional hardware was required for the project. However, the literature review stated that an 8-GPU machine was used for the original implementation of Mask R-CNN (He, Gkioxari, Dollár, & Girshick, 2017). This was a stark contrast from the i7-1165G7 CPU used to run the model in this report and gave reason to the vast difference in performance speed.

This speed was also significantly slower than the speed obtained from the semantic segmentation system. This was due to the simplicity of semantic segmentation. Advancing the system to detect different instances of an object aggressively increased the number of processes needed for each image.

The instanced data produced by the model is however vital for an automated weed management system. Some form of localisation is essential for a system like an unmanned ground vehicle to be able to target weeds and spray accordingly. It is simply not viable to obtain this data from a semantic segmentation system due to the broadness of the data. Additionally, information like the frequency and size of detections enables a spraying system to spray herbicide volume according to these parameters, optimising this process in terms of both economic impact to the farmer and the environmental impact of the herbicide.

To this end, the two CNN models analysed in this report do not meet the criteria described in the research questions of this project. While fast, the semantic segmentation model does not extract enough information for an automated weed management system. Conversely, the object detection system appears to extract adequate information, though could be improved. It does however operate at a speed far too slow to implement into a real-time system. As such, the implementations of the two models discussed in this report cannot individually be used for a real-time weed management system given the hardware available.

3.3 Risks, Ethics and Sustainability

The consideration of risks, ethics and sustainability discussed in 2.6, 2.7 & 2.8 were largely responsible for the successful completion of the project.

Consideration for the programming language, libraries and APIs meant that there were no issues pertaining to the software implementation. The final software system operated as desired and successfully took input images and returned different segmentation techniques. The use of virtual environments to manage software and library versions enabled both CNN models to operate on their documented parameters in unison.

Despite this, the magnitude of importance of computational power was misjudged in the considerations. It was stated that GPUs are usually used to train CNN models and significantly impact the testing time. To mitigate this risk the CNNs were trained overnight when no other processes were running. Additionally, in preliminary implementations of the U-Net architecture, the model took 3 hours to train for 9 epochs. This information was used to make an assumption that while the Mask R-CNN was more complex and required more training time, it would take around 8 hours to train at maximum. This was a consequential misjudgement as the model took 16 hours to train in implementation. This large training time made it difficult to fine-tune training parameters. Instead, training was conducted only once for 5 epochs, and the detection parameters were fine-tuned.

The impact of GPUs also played a larger role in model runtime and performance than anticipated. Where literature showed potential for 5-10 fps for the Mask R-CNN model, implementation in this project reached just over 1 fps. While significantly faster than methods like R-CNN discussed in the literature review, this speed is not high enough to implement in real-time. During the literature review, other CNN implementations such as YOLO were investigated for their speed. However, they were not pursued due to their loss of accuracy in exchange for increased speed. Future work may investigate YOLO or other lightweight models that are designed to run in real-time and assess whether they still perform accurately enough for this problem space.

Project trajectory was well managed through the considerations made in *2.6.* Setting realistic completion dates for intermediate tasks meant that each task was completed on time so that the successive tasks that depended on them were also completable with enough leeway if something were to take longer. Regular meetings with project supervisors also supported the project timeline as the student was able to get timely and useful feedback as the project progressed.

Feedback from the most recent meeting with project supervisors revealed that the project findings were good and although the model's results make them inappropriate for the prescribed task given the project scope, the findings are useful for future works. This further proves the project life cycle as this report can act as the foundation for future work in this space.

4 CONCLUSION

To summarise, this project created two software systems for detecting the Rumex obtusifolius weed. One model was tasked with semantic segmentation, and the other with object detection and instanced segmentation. This was motivated by the huge cost weeds pose on the agriculture industry, not just economically, but in terms of environmental impact and socially through human labour.

Both detection models were created using convolutional neural networks that extracted features from image and made decisions based upon unsupervised learning. The semantic segmentation CNN was built upon the U-Net architecture as it was shown to maintain contextual awareness after encoding by concatenating feature maps with ones from before the max pooling process. The object detection software was built upon the Mask R-CNN model as literature showed it to have the real-time object detection capabilities of Faster R-CNN while also providing instanced segmentations.

The detection systems aimed to fill a gap in the literature and prove the viability of detecting and treating Rumex weeds in real-time. However, the testing results with the specifications available disproved this viability on these two particular models. Results from the U-Net model achieved IoU scores above 70% on all parameters and achieved accuracies of up to 85%. The model achieved higher accuracy metrics when trained on un-scaled and original sized images. This increase in performance came at the cost of process speed, with the system taking 6.35 seconds to make predictions on 135 images from the testing dataset. Speed was increased by factoring down the input image size when training the model. Reducing the image size to 64^2 pixels from 256^2 increased the speed to 1.95 seconds to make predictions on the same 135 images. However, this decreased the accuracy performance. Though these speeds are highly promising for real-time applications, the data value obtained from semantic

segmentation is limited. The information semantic masks contain is too broad to be useful for an automated weed treatment system without applying an additional localisation process.

Conversely, the Mask R-CNN model extracted useful information such as bounding box locations of different occurrences of Rumex in the same frame. This information is useful to stakeholders such as farmers as they are able to precision spray weeds and vary the amount of herbicide used based upon the size and frequency of detections. The system produced mAP results up to 52.28% and F1 scores of 54.23%. The model achieved greater numerical results by lowering detection thresholds. However, this led to duplicate predictions of the same weed and was deemed unfavourable for stakeholders as it would lead to higher volume of herbicide being used than necessary. The biggest downfall of the Mask R-CNN was that it only operated at 1.18 fps on the hardware available. While this theoretically could be used, subsequent work needs to be made on this. This could include mounting a processor with the software on a UGV and reporting the detection results.

Future works could also use the implementations described in this report on more powerful devices with greater processing power. Alternatively, since the rectangular ground truth labels for the dataset caused both models to make predictions that extended past the bounds of the weeds, the output masks of the semantic segmentation could be used as ground truths to train a more lightweight, speed-based model. This would theoretically solve issues of both overestimation and performance speed.

Overall, the project successfully delivered on the segmentation and classification systems described in the scope, as well as the other intermediate deliverables outlined in the project proposal. Some changes were made to the scope throughout the semester after meetings with project supervisors and stakeholders. The most notable change was to use the Faster R-CNN/Mask R-CNN model instead of using a novel RPN and classification network separately. Despite the change, the project still delivered two CNN models and answered the research questions posed for this project.

REFERENCES

Abdulla, W. (2018). Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. Github. Retrieved from https://github.com/matterport/Mask_RCNN

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon, 4*(11).

Adil, T., Ahmed, G., Mohammed, B., & Soufiane, B. (2022). Weeds detection efficiency through different convolutional neural networks technology. *International Journal of Electrical and Computer Engineering, 12*(1), 1048-1055.

Alam, M. S., Alam, M., Tufail, M., Khan, M. U., Güneş, A., Salah, B., . . . Khan, M. T. (2022). TobSet: A New Tobacco Crop and Weeds Image Dataset and Its Utilization for Vision-Based Spraying by Agricultural Robots. *Applied Sciences, 12*(3).

Cai, L., Long, T., & Dai, Y. (2020). Mask R-CNN-Based Detection and Segmentation for Pulmonary Nodule 3D Visualization Diagnosis. *IEEE Access*.

Dave, V. S., & Dutta, K. (2014). Neural network based models for software effort estimation: a review. *Artificial Intelligence Review*, 295-307.

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*, 303-338.

Girshick, R. (2015). Fast R-CNN. *IEEE International Conference on Computer Vision (ICCV)*, 1440-1448.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 580-587.

Grace, R. K., Anitha, J., Sivaramakrishnan, R., & Sivakumari, M. S. (2021). Crop and Weed Classification Using Deep Learning. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 935-938.

Hafiz, A. M., & Bhat, G. M. (2020). A Survey on Instance Segmentation: State of the art. *arXiv*.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. *IEEE International Conference on Computer Vision (ICCV)*, 2980-2988.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition., (pp. 770-778).

Hlaing, S. H., & Khaing, A. S. (2014). Weed and Crop Segmentation and Classification Using Area Thresholding. *International Journal of Research in Engineering and Technology, 3*, 375-382.

Kamath, R., Balachandra, M., Vardhan, A., & Maheshwari, U. (2022). Classification of paddy crop and weeds using semantic segmentaion. *Cogent Engineering, 9*, 3-15.

Kohavi, R., & Provost, F. (1998). Glossary of terms. Special issue of applications of machine learning and the knowledge discovery process. *Machine Learning 30*.

Kounalakis, T., Malinowski, M. J., Chelini, L., Triantafyllidis, G. A., & Lazaros, N. (2018). A Robotic System Employing Deep Learning for Visual Recognition and Detection of Weeds in Grasslands. *IEEE International Conference on Imaging Systems and Techniques (IST)*, 1-6.

Lam, O. H., Dogotari, M., Prüm, M., Vithlani, H. N., Roers, C., Melville, B., . . . Becker, R. (2021). An open source workflow for weed mapping in native

grassland using unmanned aerial vehicle: using Rumex obtusifolius as a case study. *European Journal of Remote Sensing*, 71-88.

Li, Z., Yang, W., Peng, S., & Zhou, J. (2021). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 1-21.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature Pyramid Networks for Object Detection. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 936-944.

Liu, L., Wang, Y., & Zhao, H. (2019). An Image Segmentation Method for the blind sidewalks recognition by using the convolutional neural network U-net. *IEEE International Conference on Signal, Information and Data Processing (ICSIDP)*.

Liu, X., Deng, Z., & Yang, Y. (2018). Recent progress in semantic image segmentation. *Artificial Intelligence Review*, 1089-1106.

McLeod, R. (2018). *Annual Costs of Weeds in Australia.* Canberra: Centre for Invasive Species Solutions.

Narasinga Rao, M., Venkatesh Prasad, V., Sai Teja, P., Zindavali, M., & Phanindra Reddy, O. (2018). A survey on prevention of overfitting in convolution neural networks using machine learning techniques. *International Journal of Engineering and Technology (UAE)*, 177-180.

O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G. V., Krpalkova, L., . . . Walsh, J. (2019). Deep Learning vs. Traditional Computer Vision. *Computer Vision Conference (CVC)*, 128-144.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., & Savarese, S. (2019). Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression. *arXiv*.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 234-241.

Rowley, H. A., Baluja, S., & Kanade, T. (1998). Neural Network-Based Face Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23-38.

Salscheider, N. O. (2021). FeatureNMS: Non-Maximum Suppression by Learning Feature Embeddings. *International Conference on Pattern Recognition (ICPR)*.

Sheikh, R., Milioto, A., Lottes, P., Stachniss, C., Bennewitz, M., & Schultz, T. (2020). Gradient and Log-based Active Learning for Semantic Segmentation of Crop and Weed for Agricultural Robots. *IEEE International Conference on Robotics and Automation (ICRA)*, 1350-1356.

Shiruru, K. (2016). An Introduction to Artificial Neural Network. *International Journal Of Advance Research And Innovative Ideas In Education*, 27-30.

Sonntag, D., Stauden, S., Barz, M., Rahmani, V., Lörincz, A., Zacharias, J., & Fóthi, Á. (2017). Fine-tuning deep CNN models on specific MS COCO categories. *ArXiv*.

Sugar Research Australia. (2022). *Reducing herbicide usage on sugarcane farms in reef catchment areas with precise robotic weed control*. Retrieved 4 15,

2022, from https://sugarresearch.com.au/current-research-projects/

Talaviya, T., Shah, D., Patel, N., Yagnik, H., & Shah, M. (2020). Implementation of artificial intelligence in agriculture for optimisation of irrigation and application of pesticides and herbicides. *Artificial Intelligence in Agriculture, 4*, 58-73.

Tao, T., & Wei, X. (2022). A hybrid CNN–SVM classifer for weed recognition in winter rape field. *Plant Methods, 18*(29).

TensorFlow. (2021, 4 22). *TensorFlow Core v2.9.1*. Retrieved from TensorFlow: https://www.tensorflow.org/api_docs/python/tf/keras/metrics/Accuracy

van Evert, F. K., Polder, G., van der Heijden, G., & Kempenaar, C. (2009). Real-time vision-based detection of Rumex obtusifolius in grassland. *Weed Research, 49*(2), 164-174.

van Evert, F. K., Samsom, J., Polder, G., Vijn, M., van Doreen, H.-J., Lamaker, A., . . . Lotz, L. A. (2010). A robot to detect and control broad-leaved dock (Rumex obtusifolius L.) in grassland. *Journal of Field Robotics*.

Vasavi, P., Punitha, A., & Rao, T. V. (2022). Crop leaf disease detection and classification using machine learning and deep learning algorithms by visual symptoms: a review. *International Journal of Electrical and Computer Engineering (IJECE)*, 2079-2086.

Vayssade, J.-A., Jones, G., Gée, C., & Paoli, J.-N. (2022). Pixelwise instance segmentation of leaves in dense foliage. *Computers and Electronics in Agriculture*.

Veeragandham, S., & Santhi, H. (2021). A Detailed Review on Challenges and Imperatives of Various CNN Algorithms in Weed Detection. *International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, 1068-1073.

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A Survey of Transfer Learning. *Journal of Big Data*.

Wen, J., Yihui, R., Ying, L., & Jiaxu, L. (2022). Artificial Neural Networks and Deep Learning Techniques Applied to Radar Target Detection: A Review. *Electronics (Switzerland), 11*(1).

Yamashita, R., Nishio, M., Do, R. K., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights Imaging* , 611-629.

Yang, S., Zheng, L., Chen, X., Zabawa, L., Zhang, M., & Wang, M. (2022). Fine-tuning deep CNN models on specific MS COCO categories. *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1665-1674.

Yu, R., Wang, Y., Zou, Z., & Wang, L. (2020). Convolutional neural networks with refined loss functions for the real-time crash risk analysis. *Transportation Research Part C*.

Zisserman, A., & Simonyan, K. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ICLR*.

5 APPENDIX

```
-   annotation
    -   folder
    -   filename
    -   path
    -   source
        -   database
    -   size
        -   width
        -   height
        -   depth
    -   segmented
    -   object
        -   name
        -   pose
        -   truncated
        -   difficult
        -   bndbox
            -   xmin
            -   ymin
            -   xmax
            -   ymax
```

Figure 5.1 Annotation xml file structure

Figure 5.2 Random selection of Mask R-CNN bounding box outputs on the test dataset. Ground truth labels are displayed to the left for comparison

Table 5.1 Project Deliverables

#	Focus	Deliverable	Dependant	Milestone
1	Literature Review	Literature Review Report		1/5/2022
2	Conceptual design	Outline of methodology for segmentation and	1	1/5/2022

		classification		
3	Ground truth label software	Python program	1,2	22/5/2022
4	Semantic segmentation software	Python program	1,2,3	5/6/2022
5	Test & validate	Accuracy, latency, visualisation of predictions	4	
6	Interim Report	Progress on software system, results, changes to scope	4,5	14/6/2022
7	Interim Presentation	Oral presentation of project so far and interim results	6	29/6/2022
8	Stakeholder consultation	Project trajectory analysis		21/6/2022
9	Further Literature	Object detection literature review	1,8	8/9/2022
10	Classification software	Image segmentation and classification software	4,9	29/9/2022
11	Test and evaluate	Visual predictions, mAP, mAR, F1 score, latency	10	13/10/2022
12	Final Report	Final Report of project	11	26/10/2022
13	Final Oral Defence	Oral presentation of project	11	16/11/2022

Table 5.2 Mask R-CNN Default Configurations

Configurations:	**Value:**
BACKBONE	resnet101
BACKBONE_STRIDES	[4, 8, 16, 32, 64]
BATCH_SIZE	2
BBOX_STD_DEV	[0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE	None
DETECTION_MAX_INSTANCES	100
DETECTION_MIN_CONFIDENCE	0.7
DETECTION_NMS_THRESHOLD	0.3
FPN_CLASSIF_FC_LAYERS_SIZE	1024
GPU_COUNT	1
GRADIENT_CLIP_NORM	5.0
IMAGES_PER_GPU	2
IMAGE_CHANNEL_COUNT	3
IMAGE_MAX_DIM	256
IMAGE_META_SIZE	14
IMAGE_MIN_DIM	256
IMAGE_MIN_SCALE	0

IMAGE_RESIZE_MODE	square
IMAGE_SHAPE	[1024 1024 3]
LEARNING_MOMENTUM	0.9
LEARNING_RATE	0.001
LOSS_WEIGHTS	{'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mrcnn_class_loss': 1.0, 'mrcnn_bbox_loss': 1.0, 'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE	14
MASK_SHAPE	[28, 28]
MAX_GT_INSTANCES	100
MEAN_PIXEL	[123.7 116.8 103.9]
MINI_MASK_SHAPE	(56, 56)
NAME	rumex_cfg
NUM_CLASSES	2
POOL_SIZE	7
POST_NMS_ROIS_INFERENCE	1000
POST_NMS_ROIS_TRAINING	2000
PRE_NMS_LIMIT	6000
ROI_POSITIVE_RATIO	0.33
RPN_ANCHOR_RATIOS	[0.5, 1, 2]
RPN_ANCHOR_SCALES	(32, 64, 128, 256, 512)
RPN_ANCHOR_STRIDE	1
RPN_BBOX_STD_DEV	[0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD	0.7
RPN_TRAIN_ANCHORS_PER_IMAGE	256
STEPS_PER_EPOCH	405
TOP_DOWN_PYRAMID_SIZE	256
TRAIN_BN	False
TRAIN_ROIS_PER_IMAGE	200
USE_MINI_MASK	True
USE_RPN_ROIS	True
VALIDATION_STEPS	50
WEIGHT_DECAY	0.0001