**Edith Cowan University**
School of Science

# CSI6209 Artificial Intelligence

## Assessment 3 Report

# Weed detection utilising artificial intelligence and semi-autonomous systems: A practical review of YOLO algorithms.

Written by:

Nathan Tran – Student no. 10563314 – School of Science

Stephen Goosen – Student no. 10592099 – School of Science

Tom Donaldson – Student no. 10600875 – School of Science

## Abstract

Agriculture's ongoing challenges with weeds – invasive plants that deplete nutrients, light, and space intended for crops – necessitate more efficient control methods than traditional manual techniques. These techniques, while established, often prove labour-intensive, costly, and environmentally harmful. This study delves into leveraging machine and deep learning algorithms to craft a precise and adept weed detection system. With object detection as its cornerstone, especially within an agricultural purview, this research exploits the YOLO (You Only Look Once) algorithm's prowess, specifically the YOLOv8 version. Trained extensively on the COCO and ImageNet datasets, YOLOv8 presents an auspicious approach to classifying and localizing weeds in imagery. This report details the research's approach, from data acquisition and preparation to the training of the YOLOv8 model. The training was conducted in the cloud using Google Colab, with the model achieving satisfactory results during validation. Additionally, we explored Roboflow's auto-ML toolkit, which employs various YOLO versions for custom dataset training. Our evaluation aims to compare processing time, loss metrics, mAP, recall, and other performance indicators between YOLO versions, underscoring the significance of precision and recall in assessing the model's efficacy in weed species differentiation.

## 1. Introduction / Background

Weeds are unwanted plants that compete with crops for nutrients, light, and space, leading to reduced crop yields and quality. The traditional weed control methods are often labour-intensive, costly, or environmentally unfriendly. Hence, there is an emerging need for automated, efficient, and precise weed detection systems to optimize weed management and control processes. This study revolves around implementing machine learning and deep learning algorithms to develop a robust weed detection system that is accurate and efficient.

In agriculture, object detection is a fundamental component of weed detection, especially when using machine learning or deep learning techniques. Object detection not only classifies objects (or, in this case, weeds) within images but also localizes them by placing bounding boxes around the detected object (Sivakumar et al., 2020). Renowned for its object detection characteristic, YOLO is among the most famous and widely used algorithms (Jiang et al., 2022). The original algorithm, developed by Redmon et al. (2016), powered by a single convolutional

network, performs the simultaneous prediction of multiple bounding boxes and their respective class probabilities, all while directly optimizing detection performance on complete images. YOLO holds several attractive qualities, including speed, detection accuracy, strong generalization, and its open-source nature (Keita, 2023). Since its introduction, the YOLO framework has seen multiple iterations and versions. With time, each new release brought advanced features and refined methodologies. These advancements have addressed various challenges and limitations of the original model, enhancing accuracy, speed, and overall performance. The continuous evolution of YOLO underscores the rapid pace of development in the field of object detection. For the purposes of this report, we have selected YOLOv8 as the algorithm of choice, owing to its advanced capabilities and recent improvements. In contrast to its predecessors, YOLOv8 excels not only in speed and accuracy but also in efficiency, demanding fewer parameters for its remarkable performance (Dorfer, 2023). Additionally, it offers an intuitive and user-friendly command-line interface (CLI) along with a Python package, ensuring a smoother experience for both users and developers (Dorfer, 2023). For comparative analysis, we selected YOLOv5 due to its similar object detection features.

The remainder of this report is structured as follows: Section 2 outlines our research approach; Section 3 evaluates the performance of YOLOv8 and contrasts it with YOLOv5; and Section 4 concludes the paper, highlighting challenges faced and potential improvements.

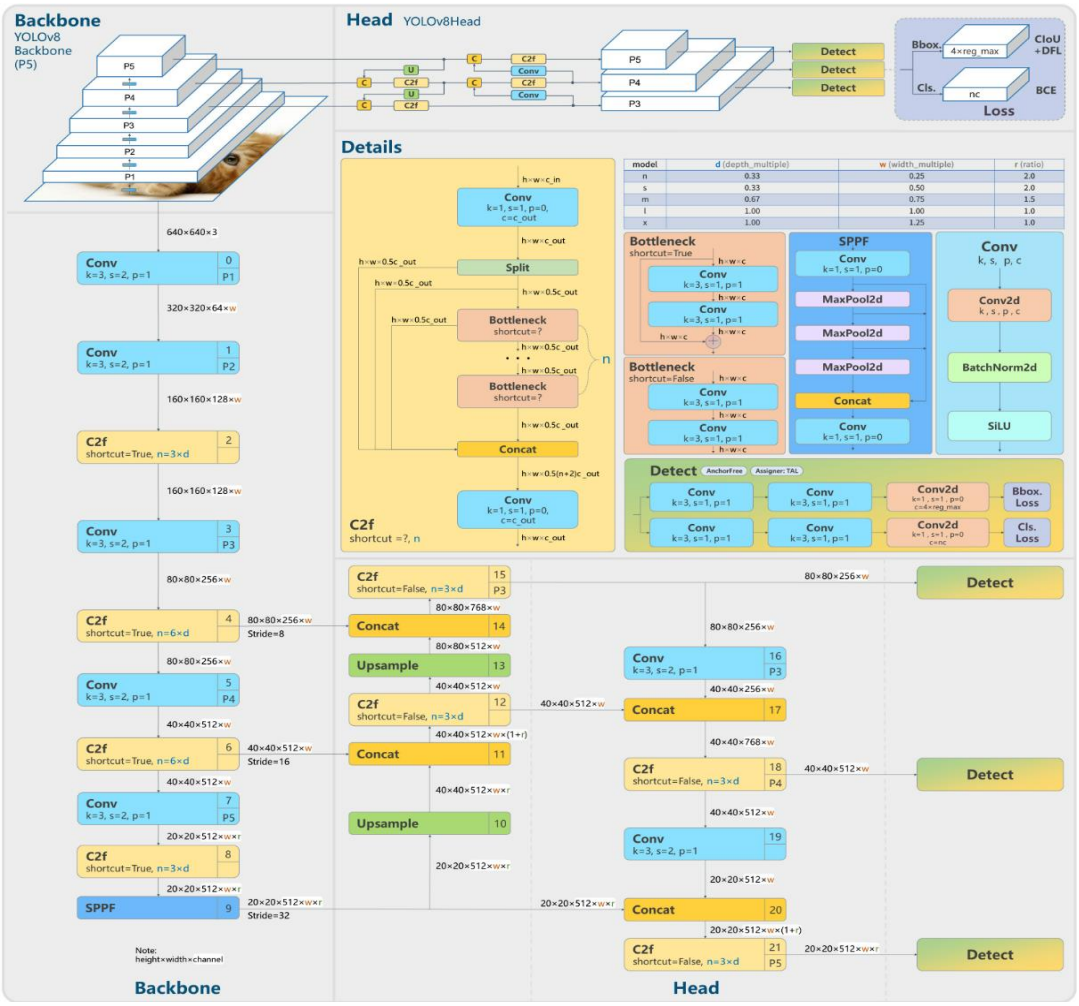
## 2. Research Approach

Over the past decade, there has been a significant amount of growth in data production and consumption, with a forecast of approximately 320 exabytes of data being created, captured, copied and consumed per day in 2023 (Statista, 2023). As a result, we constantly witness new and exciting technologies capable of handling larger amounts of data for various purposes. In our previous work, we noted that several researchers used a model called YOLOv3 (You Only Look Once) for computer vision projects. However, over the space of approximately five years, the YOLO model has evolved five times with the release of YOLOv8 in January 2023, which is the model we chose for the practical aspect of this report.

YOLOv8

YOLOv8 is an open-source, state-of-the-art (SOTA) object detection model developed by Ultralytics. Like the previous YOLO variants, YOLOv8 was trained on the COCO and ImageNet datasets, which contain millions of images, making it highly suitable for image classification, object detection and instance segmentation.

**Figure 1 - Structure of YOLOv8 detection model**




Source: RangeKing GitHub repository, https://github.com/ultralytics/ultralytics/issues/189

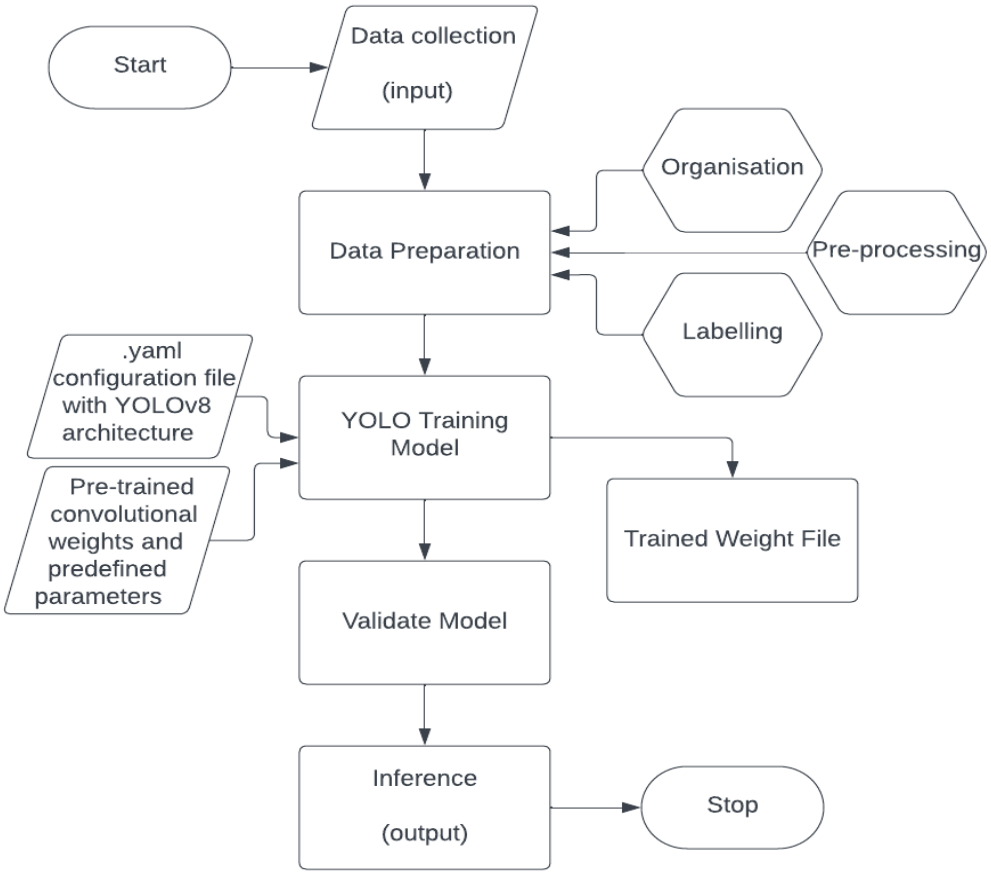
The backbone of YOLOv8, as shown in Figure 1 (above), is a Feature Pyramid Network (FPN) inspired, multi-layered convolution neural network. The P1 and P2 layers use a fast version of spatial pyramid pooling (SPPF), which provides a multi-scaled representation of the input feature maps. By pooling at different scales, SPPF allows the model to capture features at various levels of abstraction before upsampling and passing them into the model's head (layers 3-5). The head of the model - which has been decupled into two segments (one for classification and the other for detection) - employs the remaining fully connected layers, which incorporate numerous weights and biases to detect features using a loss function based on the bounding box and class loss (Ultralytics, 2023). Dandekar et al. (2022) mention that the fully connected layer is always located immediately before the output within a CNN architecture.

The current version of YOLOv8 offers five different-sized models, all trained and tested on the COCO dataset. On the one hand, the nano model, which contains 3.2 million parameters, is the fastest but also the least accurate, with an mAP value of 37.3%. On the other hand, the extra-large model contains 68.2 million parameters and is the most accurate, achieving an mAP of 53.9% - but also the most resource-intensive. To achieve a reasonable level of accuracy and avoid any significant computational delays, we opted for the small-sized model, which contains 11.2 million parameters and achieved a mAP value of 44.9% on the COCO dataset (Ultralytics, 2023).

When we began the development of our weed detection algorithm, we decided that a cloud development environment would be the most viable option for our project as our group is geographically dispersed. The platform we chose was Google Colaboratory, commonly called “Colab,” which offers users free access to their web-based Jupyter Notebook service and computing resources. As the underlying framework of YOLOv8 is PyTorch - which makes it relatively easy to use either CPU or GPU for machine learning - we could use Colab’s GPU via CUDA v11.8. CUDA, a model created by Nvidia for parallel computing and API integration, enables users to significantly increase their computing power by leveraging the power of high-end GPUs. Thus, for our project, we used CUDA to leverage Google Colab’s Tesla T4 GPU, which is typically faster than an average CPU, making it (and GPUs in general) a preferred option by many for machine learning projects (Partel, Kakarla & Ampatzidis, 2018).

The method that we used to build our weed detection algorithm consisted of the following:

**Figure 2 – Weed Detection Model Build Process**




- Data collection. As mentioned in our previous work, the data set chosen for this project was the weedCOCO dataset from the University of Sydney. The dataset comprises 5648 images captured using various sensors, from iPhones to DSLR cameras. The image resolution and file size also vary, with images ranging from 4032x3024 pixels to 6000x4000 pixels. The median image ratio of the dataset was 3024x4032. The average image size was 12.9 MP. The file size of the dataset was approximately 28 GB.

- Data Preparation. The preparation for our data was performed using a platform called RoboFlow, which is an online computer vision platform powered by TensorFlow JS. By using RoboFlow's toolkit, we were able to do the following effectively:
    - Organise the dataset. Our previous research showed that an 80/10/10 split - 80% train, 10% validate and 10% test – would be optimal for our weed detection algorithm. Our train set contained 4518 images, and the validation and test set each contained 565 images.
    - Pre-process all images within the dataset. Due to the image size disparity within our dataset, we decided to preprocess and reduce the image size of all images to make them one size. The size we chose was 640x640, as that was the image size used by the Ultralytics team to train the YOLOv8 model (Ultralytics, 2023). By reducing the image size to 640x640, we cut the overall dataset volume of 28 GB to approximately 5 GB, making it much lighter and user-friendly for our cloud-based architecture.
    - Re-label the data. The weedCOCO dataset had already been well annotated, which included the ground truth bounding box coordinates, type of weed, type of soil, weather description, location and camera metadata in JSON format. Unfortunately, JSON is incompatible with YOLO, so we had to convert the labels to the YOLO format, which requires one label per image containing class, ground truth bounding box coordinates and the image size (X, Y) in pixels.

It was also noted during the data preparation phase that the class balance of the dataset was skewed, with the 'amaranthus tuberculatus' class being significantly overrepresented and 'physalis angulata' being underrepresented, as shown in Figure 3 below. Although some researchers, such as Fessel (2019), suggest that class imbalance negatively impacts classification accuracy, it only applies to models that do not have focal or class loss functions – typically two-stage target detection algorithms such as R-CNN. For models with these functions, such as YOLO, the impact is marginal. As such, we decided to proceed with the WeedCOCO dataset in its current state.

**Figure 3 – Class Balance within the WeedCOCO dataset**

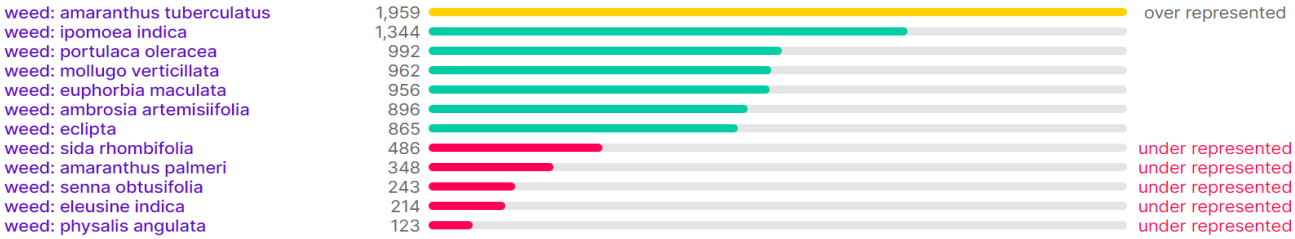



**Figure 4 – Code Structure flow within Google Colab**


- YOLOv8 Training Process:
    - After creating our working directory within Google Colab, we imported VOLOv8.0.20 into our notebook.
    - Next, we imported the pre-trained YOLOv8, trained on the MS COCO dataset.

- We then imported our dataset, which, as mentioned above, had been pre-processed and stored within RoboFlow's online open-source archive.
- Once we had the YOLO architecture and our dataset, we were able to commence training. It took approximately 2 hours to train the small (pre-weighted) YOLO model on our dataset for 200 epochs.
- The Adam optimizer was used at a learning rate of 001, with a batch norm size of 32.

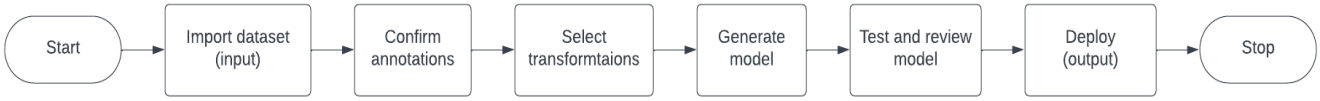
- Validating the Model. After successfully training the YOLOv8 model on our 'train' dataset, we validated the model on our 'valid' data set using the best-weighted model from the training iteration.
- Inference on test dataset. To test the model, we changed the mode parameter from validate to predict and set the confidence threshold to 0.25. Setting the confidence threshold to 25% was to maximise the number of accurate predictions, otherwise, the model will ignore the prediction entirely.

YOLOv5

Our second object detection model was compiled by an auto-ML toolkit called Roboflow. Roboflow uses various algorithms, particularly YOLO versions 3, 5 and 8, to train new models on custom datasets. The process of creating a model using Roboflow's platform involves collecting, organising, labelling processing and training before being able to test and ultimately deploy the model for inference on new sets of data.

As shown in Figure 5 below, the process for creating a working model within Roboflow was very straightforward. First, we had to upload the weedCOCO, which had been transformed into 640 x 640 images with YOLO labelling. Then, we had to verify that the labels were accurate, which was performed automatically. Once we were happy with the dataset, we commenced training. The default number of epochs for training was 200, which resulted in a 2-hour training window. After training, we were able to view the training results, as well as the mAP of the validation and test datasets.

**Figure 5 – Roboflow Auto-ML process to building a new model**




## 3. Performance Evaluation

This section aims to provide an in-depth analysis of the comparative performance of the YoloV5 and YoloV8 models on the weedCOCO dataset. The overall performance of the object detection, as well as the individual performance results of each weed class, will be analysed. We also consider the possibility of overfitting and the effects it may have on individual weed classes. This analysis provides a comparison of each object detection algorithm's strengths and weaknesses as well as insight into the value of their applications in weed management in agriculture and environmental monitoring.

In comparing the training and validation loss results between the YoloV5 and YoloV8 models, we can gain insight into the training dynamics, convergence behaviour, and potential overfitting tendencies of these two object detection models. Training models that are overly complex and are trained for too many epochs can cause high variance and low bias, which leads to overfitting of the training data. However, looking at the results of the YoloV5

model in Figure 6 and the YoloV8 model in Figure 7, this is likely not the case. For the YoloV5 model, the loss curves for the box, class, and distribution focal graphs have not yet plateaued, but the equivalent validation measures have continued to steadily decrease. For the YoloV8 model, there was a substantial drop in box, class, and distribution focal loss at the 191st epoch. This rapid convergence is a likely indication of overfitting. However, we can see that this has not affected the loss values for the validation set, suggesting that the model has not been invalidated and still generalizes well to unseen data. Overall, the loss, recall, and precision curves for both models are very similar and more detailed metrics need to be used to distinguish which model is superior.

**Figure 6 - Train/Validation Loss, Recall, and Precision Curves for YoloV5**

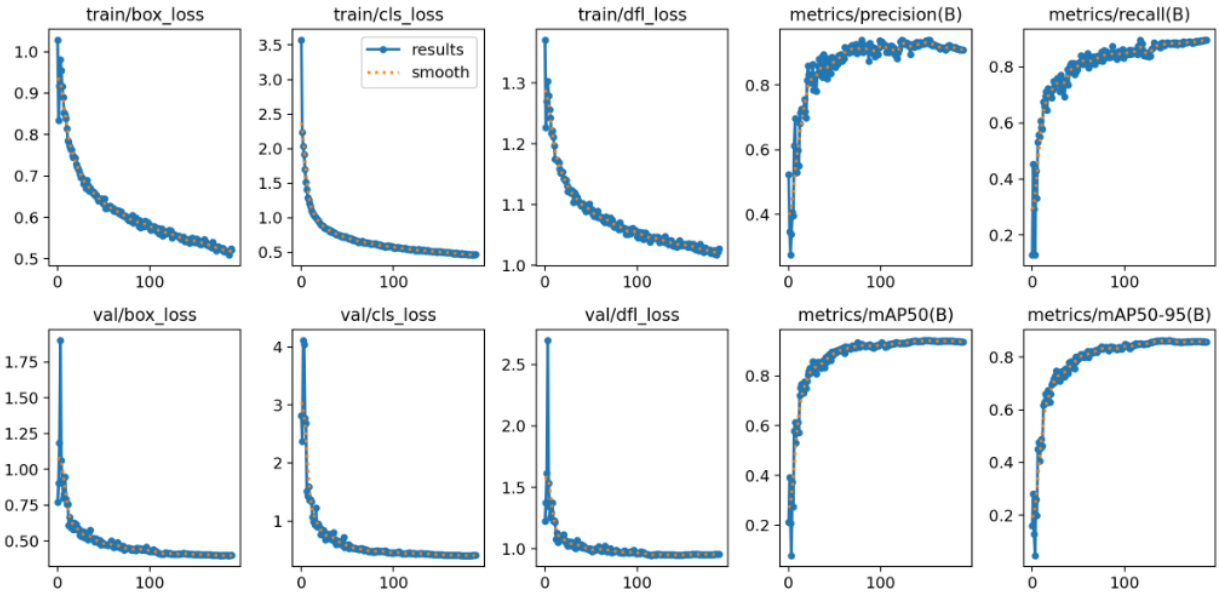
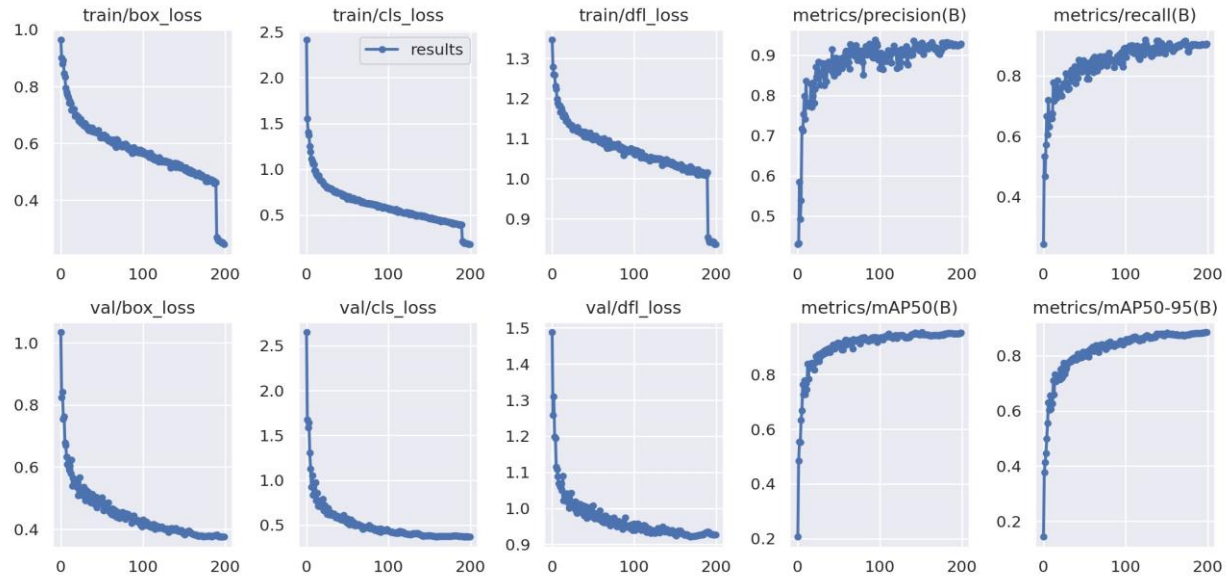



**Figure 7 – Train/Validation Loss, Recall, and Precision Curves for YoloV8**




When researchers analyse the effectiveness of Yolo object detection models, they typically consider common metrics like sensitivity (precision), specificity (recall), accuracy, and F1 scores related to classification tasks

(Salazar-Gomez et al. 2022; Dandekar et al., 2022). Additionally, they also assess measurements that help determine the accuracy of the predicted bounding boxes around the detected objects.

It is typical for machine learning tasks to have a hierarchy of importance when it comes to performance metrics. Although the datasets do not contain instances of non-weed labelled data with which to compare rates of false positives (though they do have background non-weed objects), the precision and recall measures are important indicators of the model's ability to clearly distinguish between different weed species. The effectiveness of the precision measure is amplified by its use in the evaluation of bounding box accuracy through Intersection-over-Union (IoU) and mean average precision (mAP). IoU measures the spatial overlap between the ground truth bounding box and the predicted bounding box, while the mAP takes both the precision and IoU threshold rank into account. Salazar-Gomez et al. (2022) argue that mAP results are relative to the task being performed, and a relatively low score may still be practically useful. Since we are comparing two models on the same dataset, the mAP score will be the most critical performance measure. Below are the equations to calculate average precision (AP) and mAP, where P is the Precision, R is the Recall, n is the IoU score, and N is the number of iterations.

$AP_i = \sum_n (R_n - R_{n-1}) \cdot P_n$

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i$$

The validation set performance results for the object detection models are shown in Table 1. The mAP-50 is calculated at the 50% confidence threshold. The mAP and Recall results slightly favour the YoloV8 model, whereas the Precision score slightly favours the YoloV5 model. Although the mAP is the critical measurement, we also need to consider the effects of each model's predictive performance on the classes themselves.

**Table 1 -** *Comparison of Models' mAP-50, Precision, and Recall*

| | *mAP*-50 | Precision | Recall |
|---|---|---|---|
| YoloV5 | 0.942 | 0.938 | 0.882 |
| YoloV8 | 0.953 | 0.928 | 0.906 |

**Table 2 -** *Comparison of YoloV8 and YoloV5 Precision by Class*

| | YoloV5 | YoloV8 |
|---|---|---|
| Amaranthus Palmeri | 0.96 | 0.91 |
| Amaranthus Tuberculatus | 0.95 | 0.92 |
| Ambrosia Artemisiifolia | 0.95 | 0.94 |
| Eclipta | 0.98 | 0.89 |
| Eleusine Indica | 0.95 | 0.95 |
| Euphorbia Maculata | 0.92 | 0.90 |
| Ipomoea Indica | 0.98 | 0.88 |
| Mulloga Verticillata | 0.90 | 0.99 |
| Physalis Angulata | 0.88 | 0.78 |
| Portulaca Oleracea | 0.94 | 0.94 |
| Senna Obtusifolia | 0.98 | 0.96 |

| | | |
|---|---|---|
| Sida Rhombifolia | 0.92 | 0.97 |

Table 2 shows the individual class precision scores for each model. The lowest precision scores for both YoloV5 (88%) and YoloV8 (78%) were for the Physalis Angulata weed species. If we look at the confusion matrix for the YoloV8 model training set in Figure 8, we can see that the model performed worse on the training set for Physalis Angulata precision (61%), and the majority of the false positive Physalis Angulata predictions are being predicted as Sida Rhombifolia (22%). We can see the two weeds side-by-side in Figure 9 for comparison, and it is easy to see where instances of each weed may be confused by both AI and human classification. However, it is important to remember that the Physalis Angulata weed has the lowest representation in the dataset, and its affected predictive power within each of the trained models could be mitigated through up-sampling/bootstrapping or more fine-tuning of the hyperparameters.

**Figure 8 – YoloV8 Confusion Matrix for the Training Set Performance**

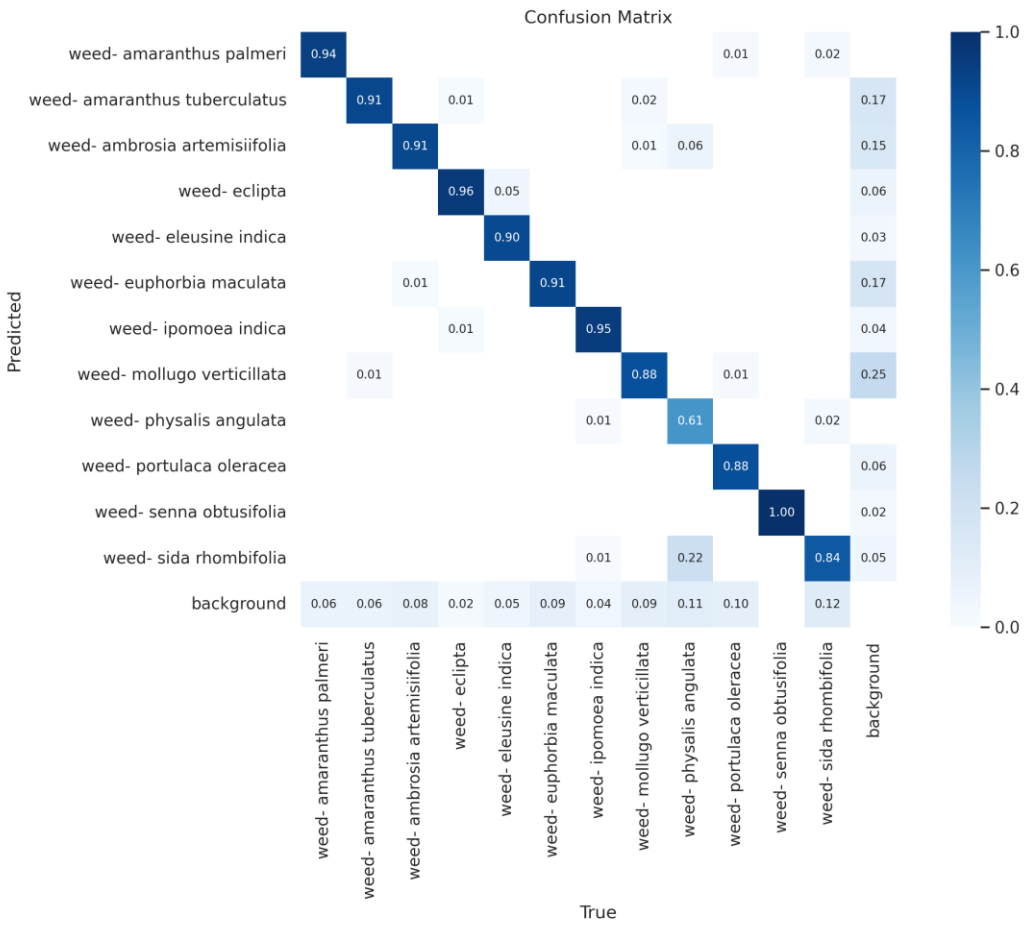
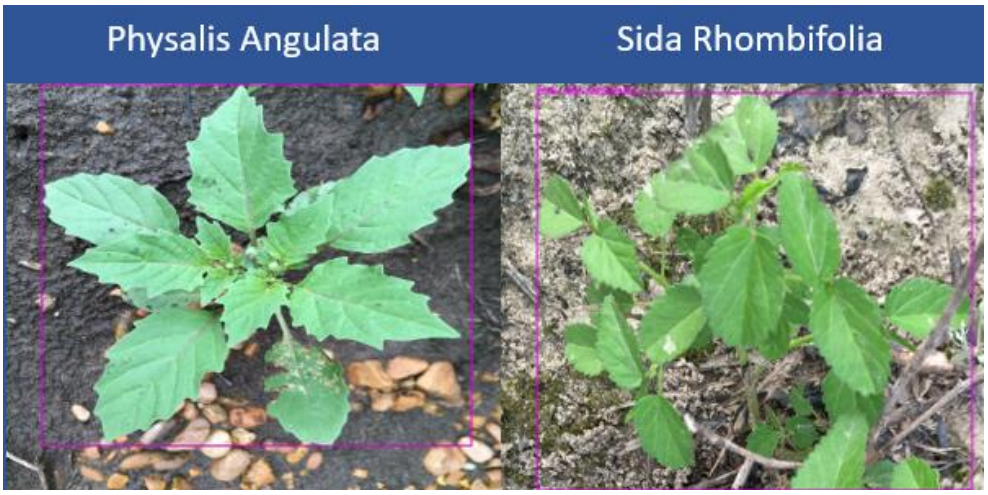
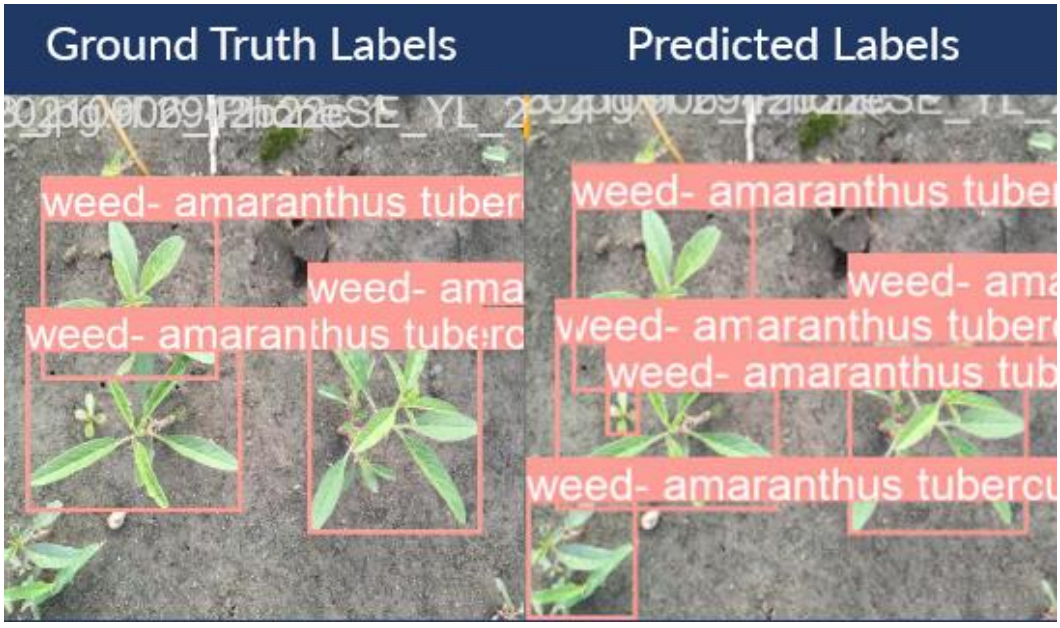



**Figure 9 – Commonly Misclassified Weeds**




Looking again at the confusion matrix in Figure 8, most instances false positives and false negatives involve non-weed background objects. This could be due to the abundance of background objects in the data set and errors with the ground truth labels themselves. As the dataset creators only considered weeds in full view of the cameras, and not those partially in view, this led to some inconsistent labelling. Figure 10 is an excellent example of this occurrence, where the partially visible Amaranthus Tuberculatus on the bottom left was not hand labelled but was correctly classified by the YoloV8 algorithm. The effects size of this mislabelling error is unknown but needs to be considered when being used in agricultural and environmental management applications.

**Figure 10 – Image Depicting Possible Ground Truth Label Errors**




While both object detection models have their merits, based on the comparative analysis of the precision, recall, and mAP-50 for both models, we have determined the trained YoloV8 model to be the most optimal in the context of weed management. This is due to the slightly greater recall and mAP-50 metrics, which we regarded as the critical measure of performance. The two caveats are the individual performance on the Physalis Angulata weed, which can be mitigated through re-tuning hyperparameters and up-sampling, and the rapid convergence anomaly at the 191st epoch, which may be an indication of overfitting.

## 4. Conclusion

The research demonstrated the capabilities of object detection algorithms, notably YOLO, in weed control operations. The YOLOv8 version was specifically chosen for this study due to its state-of-the-art performance and recent advancements. For our weed detection system, data was sourced from the weedCOCO dataset, which was then pre-processed and organized. Issues such as data imbalance were observed but did not significantly impact the model's performance. The model was trained using Google Colab, a cloud platform, leveraging Google's advanced GPU capabilities to speed up the process. Training was conducted over 200 epochs, taking roughly 2 hours. Validation and testing followed. Additionally, a secondary model, YOLOv5, was also developed using Roboflow's auto-ML toolkit. Both models' performances were assessed on metrics like precision, recall, and mAP to determine their efficacy in differentiating weed species.

Throughout the project, it has been demonstrated that machine and deep learning hold transformative potential for agriculture. However, there are also hurdles that need to be addressed. Data quality, especially amidst environmental challenges like varying lighting or plant orientation, directly influences model accuracy (Coleman, et al., 2022). Furthermore, the computational demands of sophisticated models, like CNNs, are considerable, requiring substantial processing power and memory resources. The variability of crops and weeds due to environmental factors introduces another layer of complexity, underscoring the need for continuously updated datasets. The selected algorithms' complexity and the practical integration of machine learning models into agricultural equipment further complicate the landscape.

Looking ahead, future research in weed detection algorithms can benefit from advanced data augmentation, enhancing model resilience to environmental changes. Exploring hybrid models that combine machine learning and deep learning can optimize their strengths and minimize weaknesses. Additionally, developing real-time detection capabilities is key for efficient weed management in agricultural operations.

In conclusion, while the implementation of machine and deep learning in weed detection presents promising prospects, addressing the inherent challenges requires concerted efforts, innovative approaches, and multidisciplinary collaboration. The future landscape of automated weed detection and management is ripe with opportunities for exploration, innovation, and advancement, potentially revolutionizing agricultural efficiency, productivity, and sustainability.

## Appendix A – Code Snippets and Project Links

As both of our projects are cloud-based, we are unable to save and upload the projects in a .zip file. Instead, we have copied the links to our projects that can be viewed and tested online.

Link to Google Colaboratory project:
https://colab.research.google.com/drive/13dtXHWxCvBWeoLpiNGwAaygLu7bBG7zs?usp=sharing

Link to RoboFlow project :
https://universe.roboflow.com/td/g3_weeds/browse?queryText=&pageSize=50&startingIndex=0&browseQuery=true

**Below is the summary of the code that we used for our main weed detection project within Google Colab.**

Install Ultralytics YOLOv8

```
[ ]  # Pip install method (recommended)

     !pip install ultralytics==8.0.20

     from IPython import display
     display.clear_output()

     import ultralytics
     ultralytics.checks()
```

Import YOLOv8 model

```
[ ]  from ultralytics import YOLO

     from IPython.display import display, Image
```

Install our dataset from an online archive

```
[ ]  !pip install roboflow

     from roboflow import Roboflow
     rf = Roboflow(api_key="HOjJyzOb8W0c7ml2xyjl")
     project = rf.workspace("td").project("g3_weeds")
     dataset = project.version(1).download("yolov8")
```

Train the model (Set for 200 epochs).

```
%cd {HOME}

!yolo task=detect mode=train model=yolov8s.pt data={data_yaml_path} epochs=200 imgsz=640 plots=True batch=32 optimizer=Adam lr0=0.001
```

Validate the model using the best training weight.

```
[ ]  %cd {HOME}

     !yolo task=detect mode=val model={HOME}/runs/detect/train3/weights/best.pt data={dataset.location}/data.yaml
```

Test the model

```
[ ]  %cd {HOME}
     !yolo task=detect mode=predict model={HOME}/runs/detect/train3/weights/best.pt conf=0.25 source={dataset.location}/test/images save=True
```

## References


Coleman, G. R., Bender, A., Hu, K., Sharpe, S. M., Schumann, A. W., Wang, Z., . . . Walsh, M. J. (2022). Weed detection to weed recognition: reviewing 50 years of research to identify constraints and opportunities for large-scale cropping systems. *Weed Technology*, 741-757. doi:10.1017/wet.2022.84

Dandekar, Y., Shinde, K., Gangan, J., & Firdausi, S. (2022). Weed plant detection from agricultural field images using YOLOv3 algorithm. *2022 6th International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, Pune, India: IEEE. doi:10.1109/ICCUBEA54992.2022.10011010

Dorfer, T. A. (2023, November 02). Enhanced Object Detection: How To Effectively Implement YOLOv8. Retrieved from Medium: https://towardsdatascience.com/enhanced-object-detection-how-to-effectively-implement-yolov8-afd1bf6132ae

Jiang, P., Ergu, D., Liu, F., Cai, Y., & Ma, B. (2022). A Review of Yolo Algorithm Developments. *Procedia Computer Science*, 1066-1073. doi: 10.1016/j.procs.2022.01.135

Keita, Z. (2023, November 02). *YOLO Object Detection Explained*. Retrieved from Datacamp: https://www.datacamp.com/blog/yolo-object-detection-explained

Partel, V., Kakarla, S. C., & Ampatzidis, Y. (2019). Development and evaluation of a low-cost and smart technology for precision weed management utilising artificial intelligence. *Computers and Electronics in Agriculture, 157*, 339–350. doi:10.1016/j.compag.2018.12.048

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 779-788. Las Vegas, NV, USA: IEEE. doi:10.1109/CVPR.2016.91

Salazar-Gomez, A., Darbyshire, M., Gao, J., I Sklar, E., & Parsons, S. (2022). Beyond mAP: Towards practical object detection for weed spraying in precision agriculture. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 9232-9238). Kyoto, Japan: IEEE. doi:10.1109/IROS47612.2022.9982139

Sivakumar, A. N., Li, J., Scott, S., Psota, E., Jhala, A. J., Luck, J. D., & Shi, Y. (2020). Comparison of Object Detection and Patch-Based Classification Deep Learning Models on Mid- to Late-Season Weed Detection in UAV Imagery. *Remote Sensing*, 12(13), 2136. doi:10.3390/rs12132136

Statista. (2023). The volume of data/information created, captured, copied and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025. https://www.statista.com/statistics/871513/worldwide-data-created/

Ultralytics. (2023). Ultralytics: New – YOLOv8 in PyTorch. GitHub. https://github.com/ultralytics