

Project Report

M.Sc Data Science

Submitted by

Aditya Ankush Avhad

(Roll.No: 11102)

Submitted For:

Data Science Practical – VI (Project)

(CSD5308)



Department of Computer Science
Fergusson College (Autonomous), Pune



**Deccan Education Society's
Fergusson College (Autonomous), Pune
Department Of Computer Science**

CERTIFICATE

This is to certify that the project entitled
Crop vs Weed – Object Detection submitted by

1. Aditya Avhad (11102)
2. Swaroop Hadke (11116)
3. Siddhant Khobragade (11129)

in partial fulfillment of the requirement of the completion of M.Sc.(Data Science)-II [Semester-III], has been carried out by them under our guidance satisfactorily during the academic year 2020-2021.

Place: Pune

Date: 29/05/2021

Head of Department
Department Of Computer Science
Fergusson College, Pune

Project Guide:

1. _____

Examiners Name

Sign

1. _____

2. _____

INDEX

1	PROJECT DESCRIPTION	1
1.1	ABSTRACT	1
1.2	PROBLEM STATEMENT	1
2	LITERATURE SURVEY	2
3	METHODOLOGY	3
4	DATA COLLECTION	4
4.1	DATA SOURCE	4
4.2	DATA DESCRIPTION	4
5	TECHNOLOGY USED	5
6	PROJECT DETAILS AND IMPLEMENTATION	6
6.1	OBJECTIVE	6
6.2	DATA PREPARATION	6
6.3	ALGORITHM SELECTION	7
6.4	INITIAL STEPS	10
6.5	TRAINING ON CUSTOM DATA	10
6.6	RESULTS	13
6.7	DEPLOYMENT	16
6.8	MAPPING OBJECT/BOXES TO REAL LIFE	18
7	FUTURE ENHANCEMENT	21
8	CONCLUSION	22
9	REFERENCES	23

ACKNOWLEDGEMENT

With immense pleasure, I would like to present this report on my Project on Weed-Crop Object detection for Fergusson College.

I would like to use this opportunity to express my deepest gratitude **to Mrs. Swati Satpute**, who guided, and encouraged me through the project.

I would also like to convey my heartiest thanks to **Mr. Venkata Prashanth K, Data Scientist, Infosys** who in spite of being busy with his duties, took time out to hear, guide and keep me on the correct path.

Finally, I thank all the other teaching staff for their wholehearted support at every stage, of this work.

- Aditya Avhad

1 PROJECT DESCRIPTION

1.1 ABSTRACT

Weed management is one of the most important aspects of crop productivity; knowing the amount and the locations of weeds has been a problem that experts have faced for several decades. Weeds are plants that are misplaced or undesirable to the purpose of the crop plants in the field. Crop plants are being cultivated because of the economic value to the producer. Thus, any plant can be a weed, like volunteer corn growing in a soybean field, if it is not serving the purposes of the producer's management scheme. Thus plants are weeds based on their location and competition relative to the crop plants. So robotic weed control is an ill-posed problem until the agricultural producer's intentions for a field are made known to the robot, which will then identify and make decisions about which plants are the weeds that need to be controlled. Another challenge is that while crop plants are mechanically planted in a structured manner that is compatible with agricultural machinery, weed plants emerge and grow in patterns that are consistent with their ecology. Thus weed plants exist in random patterns in a field. The project, in its elementary stage, focuses on detecting crop and weed simultaneously based on deep learning image processing in sesame crop. The YOLOv5 (**Y**ou **O**nly **L**ook **O**nce **V**5) algorithm, takes advantage of its robust architecture for object detection and is capable of detecting weed amongst crops.

1.2 PROBLEM STATEMENT

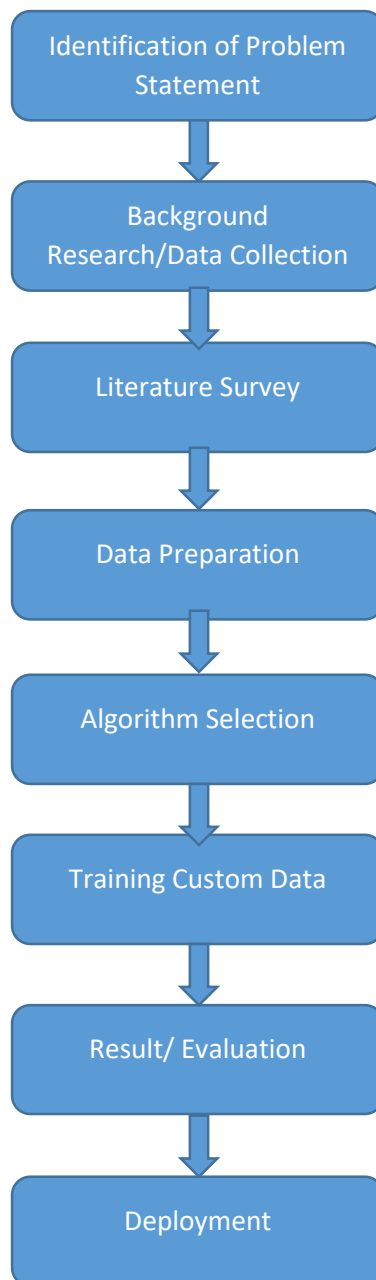
Automated Robots currently utilize various sensors like ultrasonic, infrared imaging. These methods are costly and not feasible in developing countries. Having a system which utilizes a simple handheld device attached to the robot like a mobile device or a microcontroller can be a cheaper alternative.

2 LITERATURE SURVEY

Agricultural robots have great potential to deliver weed control technologies that are much more adaptable even down to the plant scale. They potentially could direct chemical or cultivation tools to directly target weed plants. Tripathi et.al [1] in the survey paper, explored the usage of computer vision for in agriculture. Recognition and Classification models were greatly explored. The methods generally used for the approaches were Neural Networks, Support Vector Machines (SVM) and modifications of SVMs. Santos et.al [2] in their research paper looked specifically at deep learning approaches. Semantic Segmentation, Object detection and instance segmentation methods were evaluated. Mask R-CNN results were evaluated for all the three methods whilst the YOLO based results were evaluated for object detection. Structure-from-Motion (SfM), a 3-D spatial registration method was also evaluated. The Mask R-CNN network presented superior results as compared to the YOLO networks (YOLOv2 and YOLOv3) and concluded that Combining structure-from-motion (or its real-time version: SLAM) and convolutional neural networks, advanced monitoring and robotics applications can be developed for agriculture and livestock. In their review paper, Shanmugam et.al [3] looked at automated weed detection systems. The aspects broadly looked at were Localisation and Navigation, Detection and Classification of Plants and Control Methods. Some systems used infrared or combination of infrared and visible light to be able to detect green vegetation. However, it was soon discovered that good detection systems are those which can separate the weed and crop plants at the single plant or leaf level. Manual weed detection systems were not feasible and 65-85% of weeds were left unnoticed. Simulations of automated systems had probability of maximum yield was 80% and approximately 93% for crop detection.

Lottes et.al [4] and Milioto et.al [5] in their respective papers implemented Semantic Segmentation approaches based on CNNs for the Crop-Weed Classification. The results achieved were very accurate in terms of segmentation and classification. The approaches however were resource intensive.

3 METHODOLOGY



4 DATA COLLECTION

4.1 DATA SOURCE

The crop & weed data from [Kaggle](#) was used.

4.2 DATA DESCRIPTION

The dataset contained 1300 colour images of each class.

Each image is of 512 x 512 size.

The two classes were: Crop and Weed.

5 TECHNOLOGY USED

Python: Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation.

Google Colaboratory: Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.

YOLO: It a real-time object detection framework and stands for You Only Look Once. Meaning the image is only passed once through the FCNN or fully convolutional neural network.

LabelImg is a graphical image annotation tool. It is written in Python and uses Qt for its graphical interface. Annotations are saved as XML files in PASCAL VOC format, the format used by ImageNet. Besides, it also supports YOLO and CreateML formats.

Android Studio: Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.

6 PROJECT DETAILS AND IMPLEMENTATION

6.1 OBJECTIVE

With the use of Deep Learning and computer vision, object detection can be done to help detect weed and help guide the robot to remove the weed and not touch the crop.

6.2 DATA PREPARATION

The Labelling toolkit was used to label the images of each class in the YOLO format.

We choose the Bounding box method to annotate each class in the given image. Bounding boxes are usually represented by either two coordinates $(x1, y1)$ and $(x2, y2)$ or by one coordinate $(x1, y1)$ and width (w) and height (h) of the bounding box. Eg.:

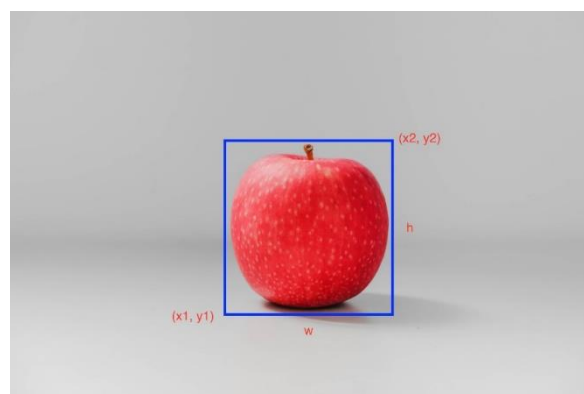


Fig 1 : The labelling of the object, accessed 28 February 2021, <<https://towardsdatascience.com/image-data-labelling-and-annotation-everything-you-need-to-know-86ede6c684b1>>

For each image, the labels were in the text format as:



0 0.478516 0.560547 0.847656 0.625000

Fig 2 : The crop image and the corresponding coordinates of the bounding box in YOLO format.

The format is as such: <object-class> <x> <y> <width> <height>

In this case the class 0 corresponds to the object being a crop. A class 1 would have corresponded to the object being weed.

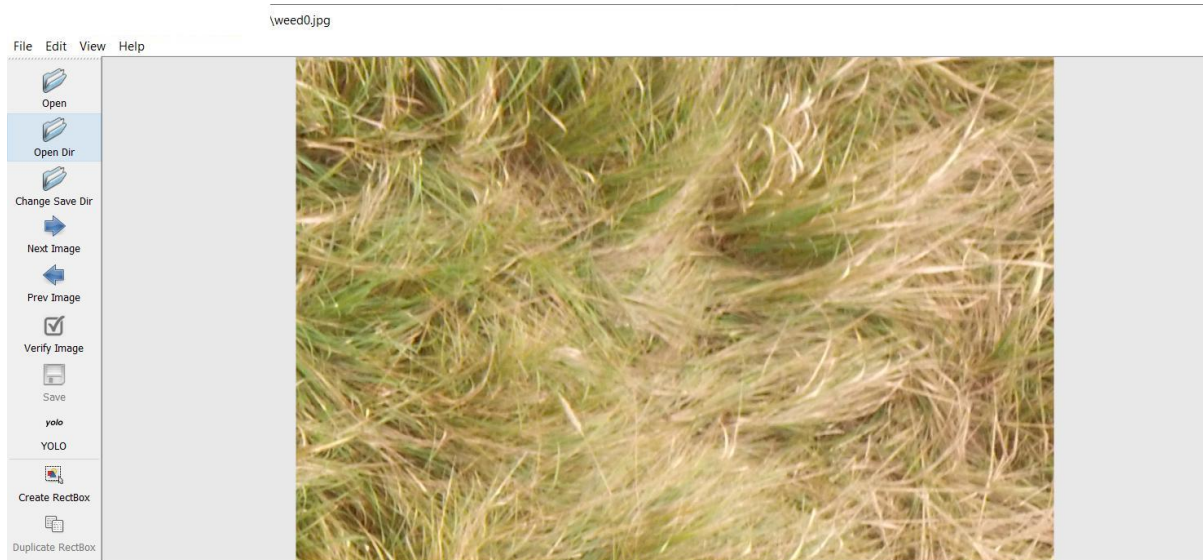


Fig 3: LabelImg for labelling images.

6.3 ALGORITHM SELECTION

For the purpose of Object Detection, multiple Object Detection Frameworks and algorithms were looked into. Comparisons of the Object Detection models were based on the MS-COCO dataset. COCO is a large-scale object detection, segmentation, and captioning dataset. The COCO dataset is used as a benchmark for comparison.

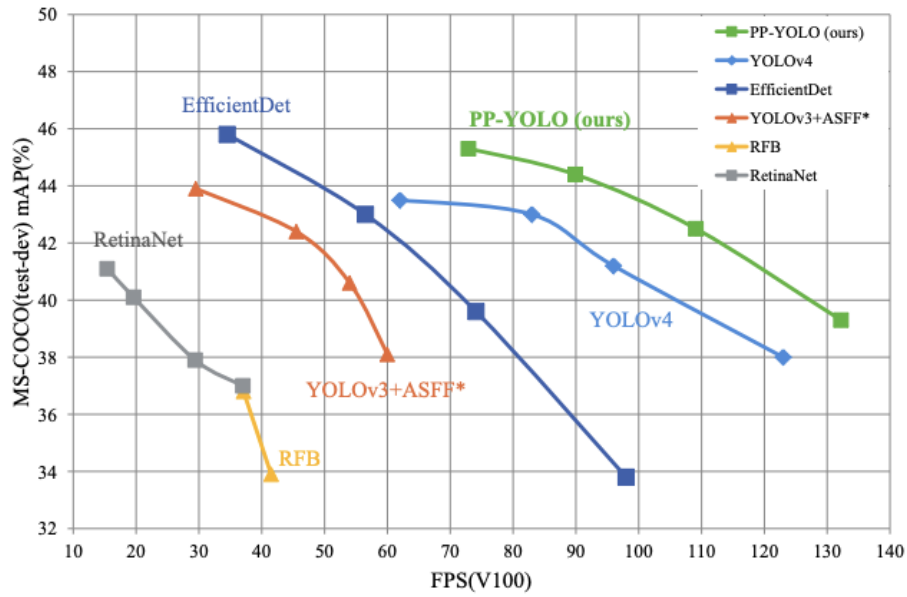


Fig 4 : Comparison amongst various Object Detection models, accessed 5 August 2020,
<https://blog.roboflow.com/yolov4-versus-yolov5/>

As evident from the above image, the YOLO based models seemed to be performing better in terms of having higher FPS as well as a richer mAP (%) relative to other models like RetinaNet, EfficientDet etc. The project commenced with understanding of the YOLO algorithm used in the YOLO model. Midway to the project the YOLOv5 model was explored. According to Roboflow's research, YOLO V5 trains very quickly, far surpassing YOLO V4 in training speed. For Roboflow's custom data set, it took 14 hours for YOLO V4 to reach the maximum verification evaluation, while YOLO V5 only took 3.5 hours.

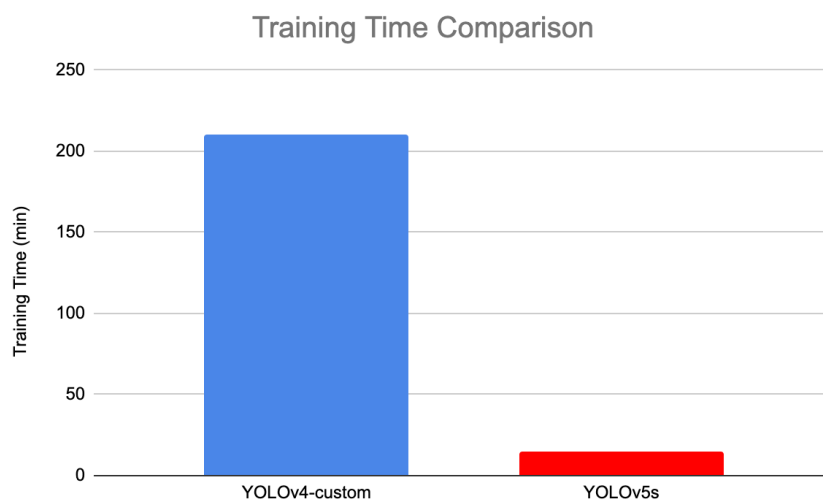


Fig 5: Comparison of YOLOv5 and YOLOv4 for training time, accessed 28 March 2021,
<https://blog.roboflow.com/yolov4-versus-yolov5/>

The two models were compared on their AP values on the benchmark COCO dataset.

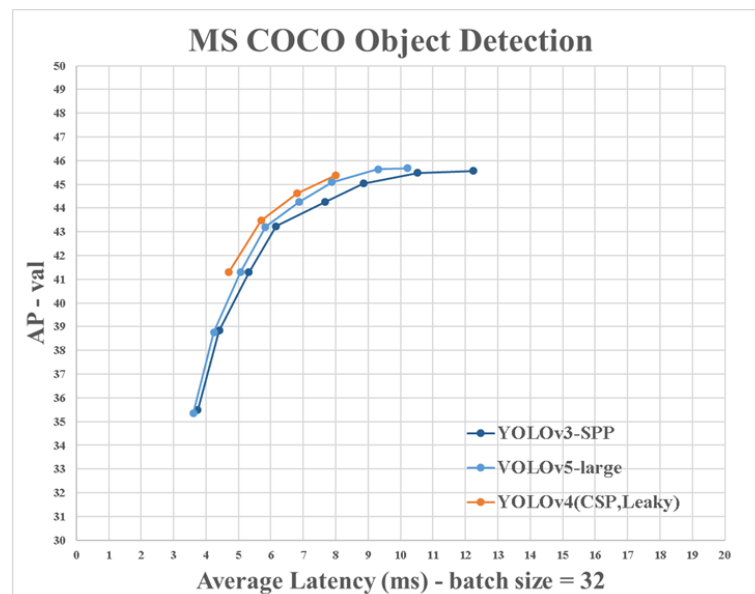


Fig 6: YOLOv4 tops YOLOv5 in mAP on the COCO benchmark, accessed 28 March 2021, <https://blog.roboflow.com/yolov4-versus-yolov5/>

The YOLOv5 and YOLOv4 were also compared for the model storage size.

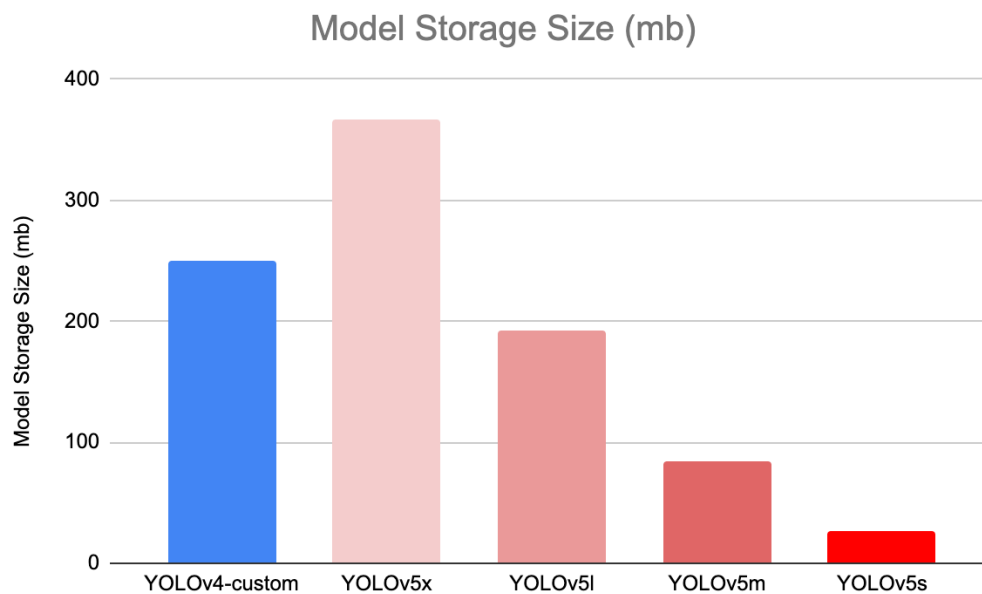


Fig 7: Model storage size comparison, , accessed 28 March 2021, <https://blog.roboflow.com/yolov4-versus-yolov5/>

Evidently, on many fronts, the YOLOv5 models seemed to be a better and logical choice over the YOLOv4. The project, hence, is based on the YOLOv5 model.

6.4 INITIAL STEPS

The project commenced with understanding of the YOLO algorithm used in the YOLO model. The YOLOv4 was initially implemented for the COCO dataset. The model, trained on COCO dataset, was used for videos. The objects were detected very well and even distant objects, which seemed pixelated to the human-eye were picked up by the algorithm accurately.

6.5 TRAINING ON CUSTOM DATA

Amongst all the YOLOv5 models available, the YOLOv5s was chosen initially since it was the smallest and the fastest of the models.

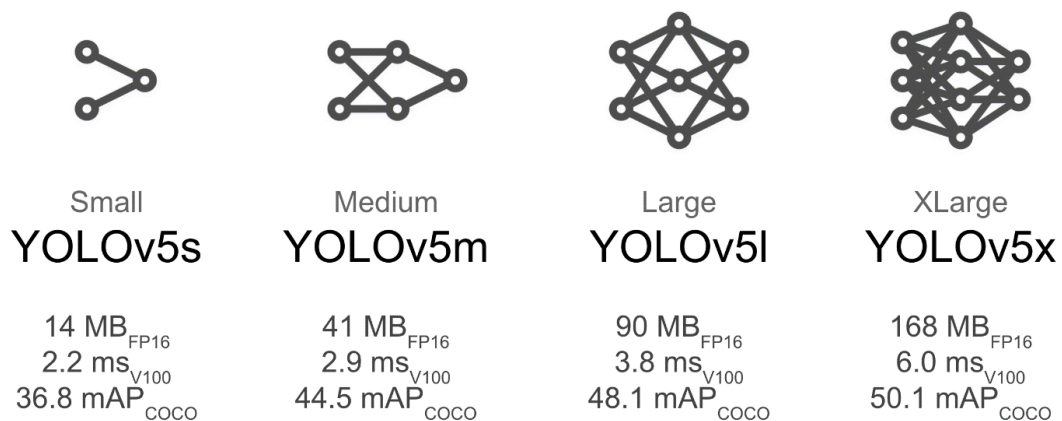


Fig 8: Brief comparison amongst the available YOLOv5 models, accessed 2 April 2021,
https://pytorch.org/hub/ultralytics_yolov5/

Model	size (pixels)	mAPval 0.5:0.95	mAPtest 0.5:0.95	mAPval 0.5	Speed V100 (ms)	params (M)	FLOPS 640 (B)
YOLOv5s6	1280	43.3	43.3	61.9	4.3	12.7	17.4
YOLOv5m6	1280	50.5	50.5	68.7	8.4	35.9	52.4
YOLOv5l6	1280	53.4	53.4	71.1	12.3	77.2	117.7
YOLOv5x6	1280	54.4	54.4	72.0	22.4	141.8	222.9
YOLOv5x6 TTA	1280	55.0	55.0	72.0	70.8	-	-

Table 1: Detailed comparison amongst the available models (based on COCO dataset) , accessed 2 April 2021,
https://pytorch.org/hub/ultralytics_yolov5/

For the training on custom dataset, the yaml file is used as an input which contains the summary information of the data set. The data.yaml file used in YOLO model is as such:

```
1. train: 'training set directory path'
2. val: 'validation set directory path'
3.
4. nc: 'number of classes'
5. names: 'name of objects'
```

Fig 9: yaml file for YOLO

The default architecture provided was utilized.

```
[43] %cat /content/yolov5/models/yolov5s.yaml

# parameters
nc: 80 # number of classes
depth_multiple: 0.33 # model depth multiple
width_multiple: 0.50 # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23] # P3/8
  - [30,61, 62,45, 59,119] # P4/16
  - [116,90, 156,198, 373,326] # P5/32

# YOLOv5 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Focus, [64, 3]], # 0-P1/2
  [-1, 1, Conv, [128, 3, 2]], # 1-P2/4
  [-1, 3, C3, [128]],
  [-1, 1, Conv, [256, 3, 2]], # 3-P3/8
  [-1, 9, C3, [256]],
  [-1, 1, Conv, [512, 3, 2]], # 5-P4/16
  [-1, 9, C3, [512]],
  [-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
  [-1, 1, SPP, [1024, [5, 9, 13]]],
  [-1, 3, C3, [1024, False]], # 9
  ]

# YOLOv5 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
  [-1, 1, nn.Upsample, [None, 2, 'nearest']],
  [[-1, 6], 1, Concat, [1]], # cat backbone P4
  [-1, 3, C3, [512, False]], # 13

  [-1, 1, Conv, [256, 1, 1]],
  [-1, 1, nn.Upsample, [None, 2, 'nearest']],
  [[-1, 4], 1, Concat, [1]], # cat backbone P3
  [-1, 3, C3, [256, False]], # 17 (P3/8-small)

  [-1, 1, Conv, [256, 3, 2]],
  [[-1, 14], 1, Concat, [1]], # cat head P4
  [-1, 3, C3, [512, False]], # 20 (P4/16-medium)

  [-1, 1, Conv, [512, 3, 2]],
  [[-1, 10], 1, Concat, [1]], # cat head P5
  [-1, 3, C3, [1024, False]], # 23 (P5/32-large)

  [[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)
  ]
```

Fig 10: The architecture of YOLOv5s

To train the model, the following command was used:

```
!python train.py --img 640 --batch 15 --epochs 10 --data wc.yaml --weights /content/yolov5/best.pt --nosave --cache
```

Fig 11: The command to train on custom dataset

Here,

- `img`: define input image size. The original image size is 512 x 512 and is upsampled to 640 x 640.
- `batch`: determine the batch size. The forwarding of thousand images into the neural network at the same time makes the number of weights that the model learns in one time (one epoch) to increase a lot. Thus, the dataset is usually divided into multiple batches of n images and training batch by batch. The results of each batch are then saved to RAM and aggregated after the training for all batches is completed. Because the weights learned from the batches are stored in RAM, so the larger the number of batches, the more memory consumption will be consumed.
- `epochs`: define the number of training epochs. An epoch is responsible for learning all input images, in other words, training all input. Since the dataset is split into multiple batches, one epoch will be responsible for training all the batches. The number of epochs represents the number of times the model trains all the inputs and updates the weights to get closer to the ground truth labels. Often chosen based on experience and intuition.
- `data`: the path to `data.yaml` file containing the summary of the dataset. The model evaluation process is executed immediately after each epoch, so the model will also access the validation directory via the path in `data.yaml` file and use its contents for evaluation at that moment.
- `weights`: specify a path to weights. A pretrained weight can be used for saving training time. If it is left blank, the model will automatically initialize random weights for training.
- `cache`: cache images for training faster

For the training the concept of Transfer Learning was utilized. The intuition behind transfer learning for image classification is that if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world. You can then take advantage of these learned feature maps without having to start from scratch by training a large model on a large dataset.

6.6 RESULTS

The YOLO runs gave an output of some evaluation plots.

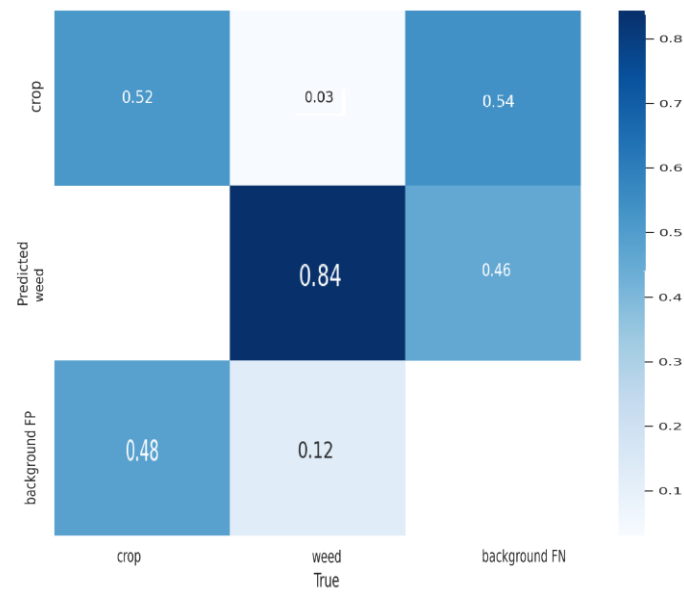


Fig 12: Confusion Matrix for the two classes and any incorrect predictions done on the background.

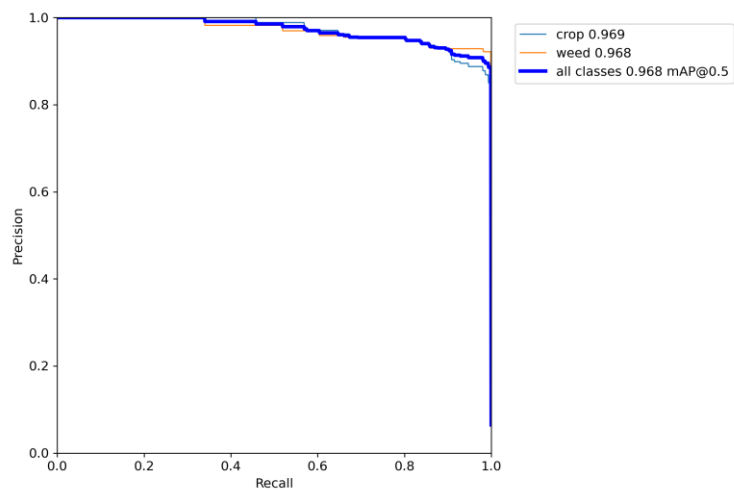


Fig 13: Precision-Recall Curve

TensorBoard was also used to visualize the entire training process.



Fig 14: mAP for when IoU is greater than 0.5 over various iterations.

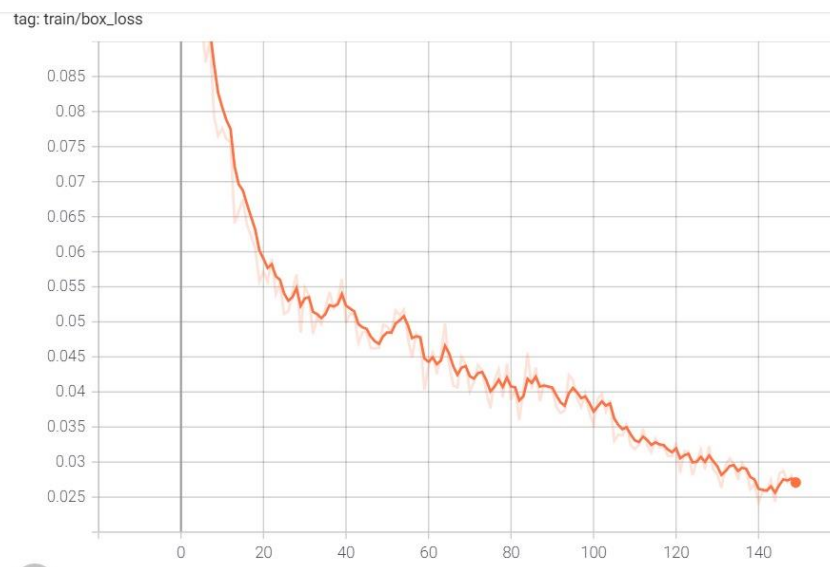


Fig 15: The Loss function against iterations

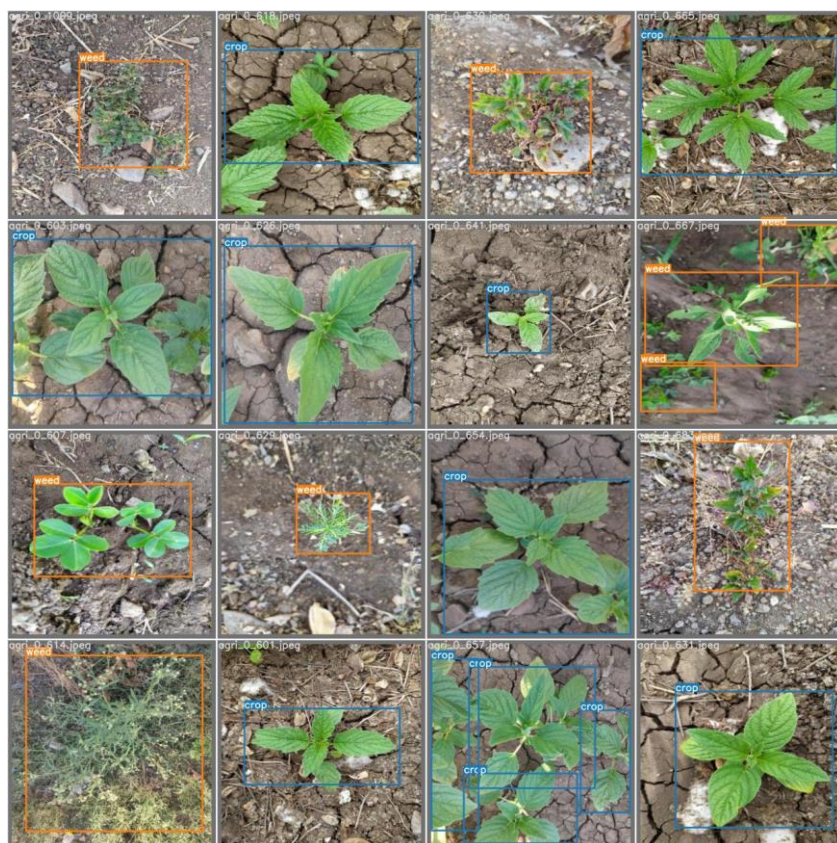


Fig 16: The labelled images

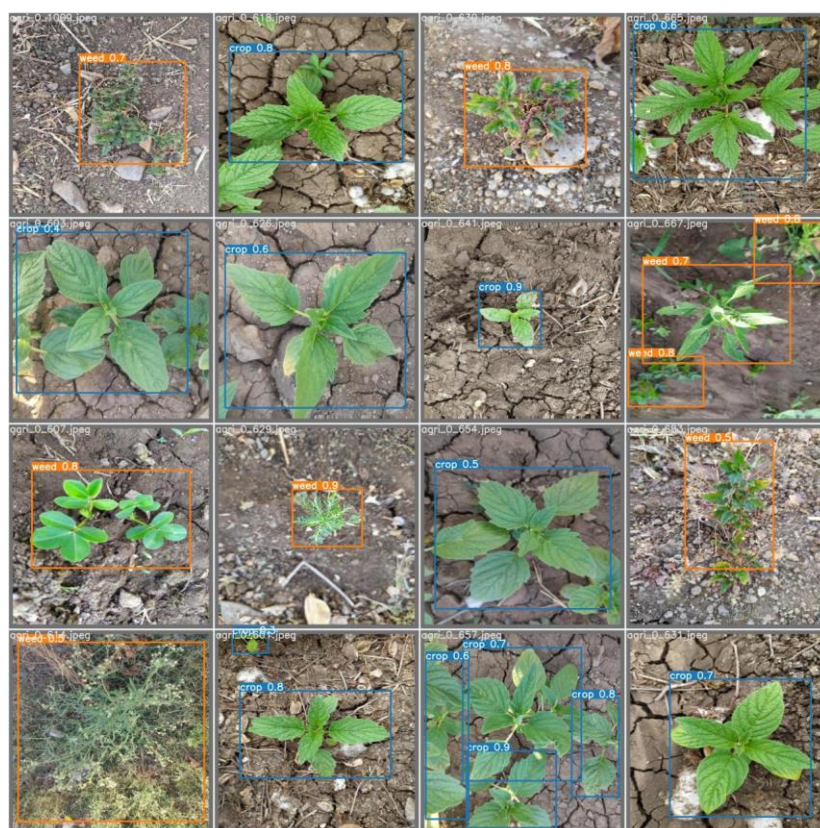


Fig 17: The images as predicted by model.

The models seems to be classifying very accurately and the detection is also very accurate.

6.7 DEPLOYMENT

The YOLOv5 models once trained on the custom data was ready to be deployed on a mobile device. The PyTorch github provides example of Android Projects to build applications for Image Segmentation, Speech Recognition, Object Detection, and more. The Android Project for the Object Detection was utilized. The custom trained model however was not fit to be used for the application and would have been heavy to run on a mobile device. The model had to be converted into an mobile compatible format to be used.

The following command was used to convert the model.pt file into model.torchscript.pt file:

```
python models/export.py --weights abcd.pt --img 640 --batch 1
```

The command runs the export.py script file from the Ultralytics Github and essentially converts the model from a .pt file to a .torchscript.pt file. The size of the model is seen to decrease by around 50%.



 weed.pt	27-05-2021 16:06	PT File	14,071 KB
 weed.torchscript.pt	27-05-2021 16:08	PT File	27,908 KB

Fig 18: The size of the model files after conversion.

The model is now ready to be integrated with the android project.

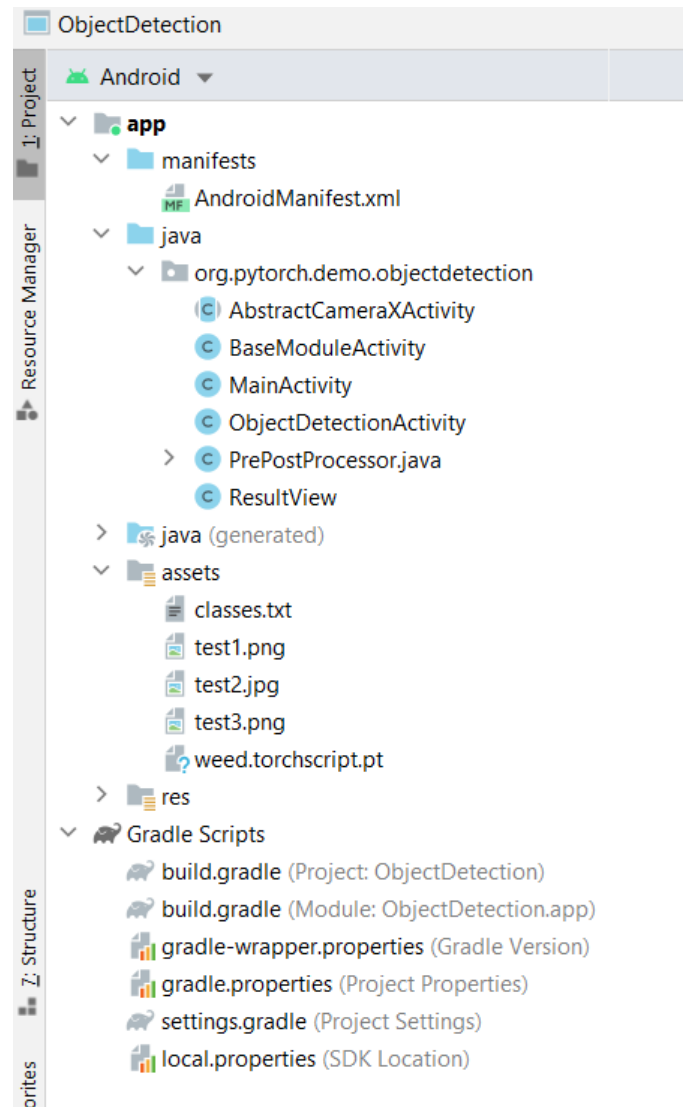


Fig 19: The project files of the Android Project.

The model.torchscript.pt and the classes.txt file are placed in the assets folder. Relevant changes to the other java files were done to properly run the model and the application.

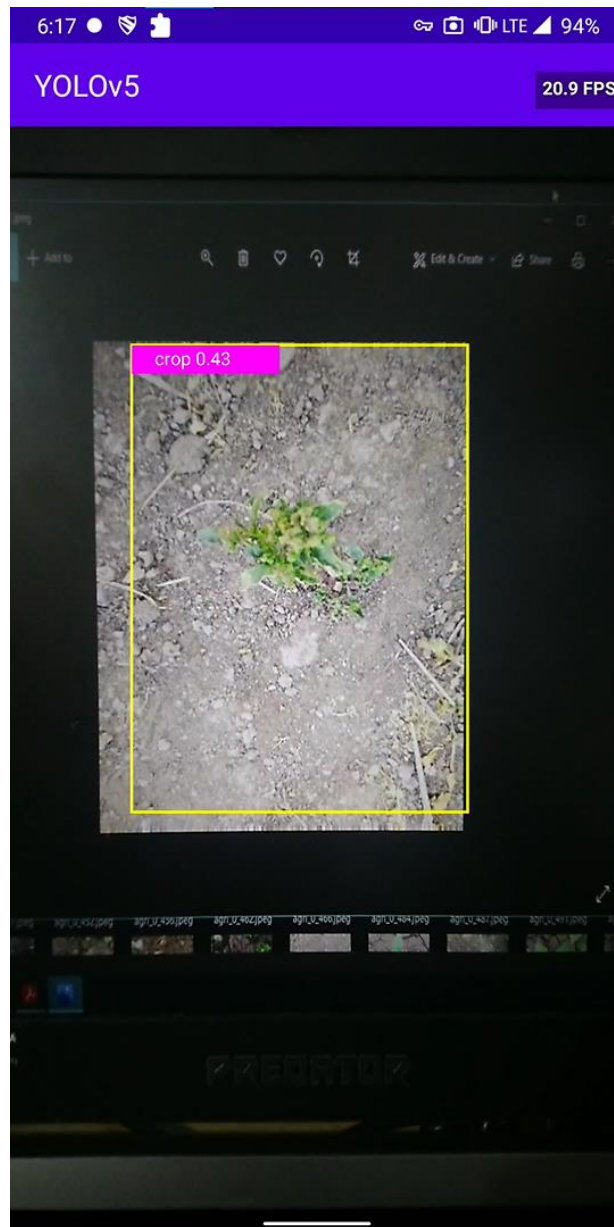


Fig 20: The Android application detecting the crop in the image.

6.8 MAPPING OBJECT/BOXES TO REAL LIFE

For the purpose of automation, the robot should be able to map the location of the weed to be removed with respect to that displayed on the screen. For that a simple experiment setup was devised.

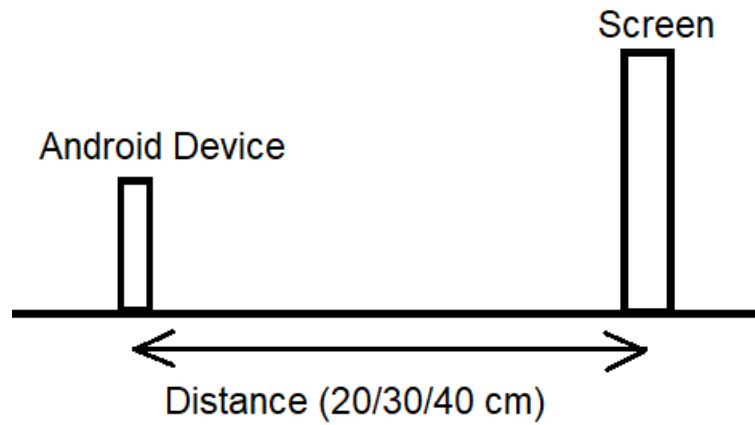


Fig 21: Experimental Setup

The experiment is such that the size of the image was measured on the android device and the screen as well. The distance between the two was maintained at 20 cm, 30 cm and 40 cm. The size of the image on the screen remained constant (11.5 cm) whilst the android device was at varying distances from the screen, had varying sizes of the image being shown.

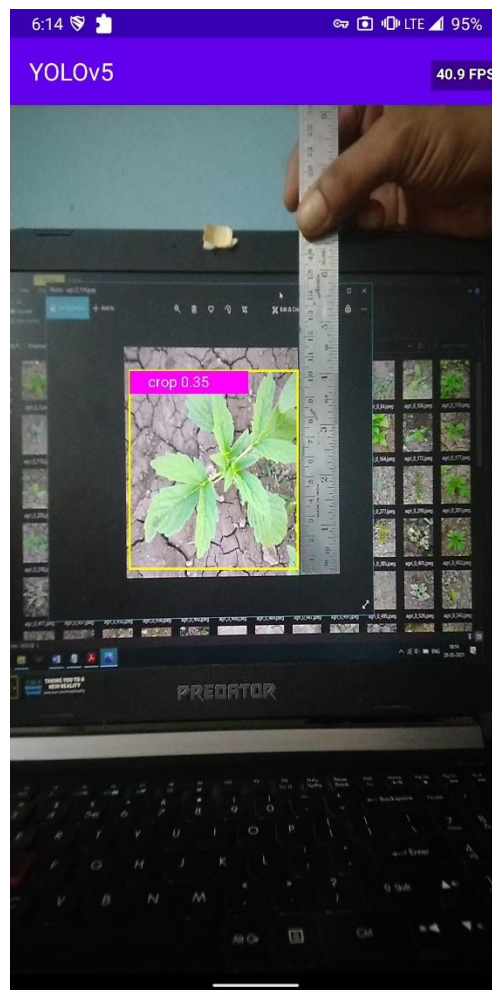


Fig 22: The measurement of image on the screen

Distance (cm)	Height of Image in Screen (cm)	Height of Image in Android Device (cm)
20	11.5	5.4
30	11.5	3.5
40	11.5	2.1

Table 2: The measurements for the experiment.

Plotting the above data on Microsoft Excel helped in concluding that the size of the image in the Android Device is inversely proportional to the distance it is from the Screen.

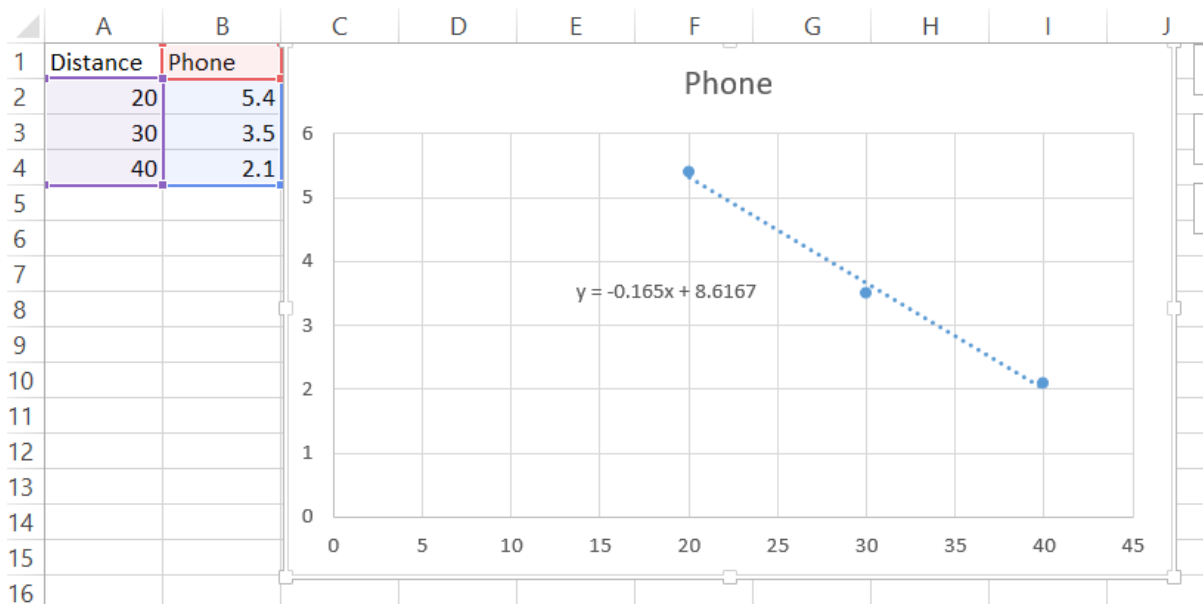


Fig 23: Plot of the experiment.

The plot and the equation can hypothetically help in mapping the location of the object and the bounding boxes from the Android Device to real life. This can help the robot in accurately cutting unwanted plants like weed.

The above obtained result is applicable to the particular model of the Android Device used and may vary for other devices due to the camera sensor being different.

7 FUTURE ENHANCEMENT

The following points can be considered in further enhancing the Project:

- Implementing the model on a microcontroller based system like Arduino or Raspberry Pi.
- Use of image segmentation to more accurately detect the object.
- Detection of the robot cutter tool to map it with respect to the object.
- Train model to detect different types of weed.
- Train model to detect different types of vegetables/fruits/crops.

8 CONCLUSION

The YOLOv5 object detection model was used to train on custom Weed-Crop data. The model is returning mAP far exceeding general values. The custom trained model was deployed on to an Android App.

9 REFERENCES

- [1].M. K. Tripathi and D. D. Maktedar, A role of computer vision in fruits and vegetables among various horticulture products of agriculture fields: A survey, *Information Processing in Agriculture*, <https://doi.org/10.1016/j.inpa.2019.07.003>
- [2]. Thiago T. Santos, Leonardo L. de Souza, Andreza A. dos Santos, Sandra Avila, Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association, *Computers and Electronics in Agriculture*, Volume 170, 2020, 105247, ISSN 0168-1699, <https://doi.org/10.1016/j.compag.2020.105247>.
- [3].Saraswathi Shanmugam, Eduardo Assunção, Ricardo Mesquita, André Veiros, and Pedro D. Gaspar, (2020), “Automated Weed Detection Systems: A Review” in *International Congress on Engineering — Engineering for Evolution*, KnE Engineering, pages 271–284.
- [4].Philipp Lottes, Jens Behley, Nived Chebrolu, Andres Milioto, Cyrill Stachniss, Joint Stem Detection and Crop-Weed Classification for Plant-specific Treatment in Precision Farming, *International Conference on Intelligent Robots and Systems (IROS)*, 2018
- [5].Milioto, Andres & Lottes, Philipp & Stachniss, Cyrill. (2017). Real-time Semantic Segmentation of Crop and Weed for Precision Agriculture Robots Leveraging Background Knowledge in CNNs. <https://arxiv.org/abs/1709.06764>.
- [6].<https://github.com/pytorch/android-demo-app>
- [7].<https://github.com/ultralytics/yolov5>
- [8].<https://blog.roboflow.com/yolov4-versus-yolov5/>
- [9].<https://arxiv.org/pdf/1506.02640.pdf>
- [10]. Steward, Brian, Jingyao Gai, and Lie Tang. “The use of agricultural robots in weed management and control.” In *Robotics and automation for improving agriculture*, edited by John Billingsley. Volume 44 of *Burleigh Dodds Series in Agricultural Science Series*. Cambridge, UK: Burleigh Dodds Science Publishing, 2019. ISBN: 9781786762726. DOI: 10.19103/AS.2019.0056.13.