Industrial Internship Report on

”Crop and Weed Detection”

Prepared by

Md Shahrukh

Executive Summary
This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT). This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks’ time. My project was ***Crop and Weed Detection*** where with the goal of developing a system that targets weeds for pesticide application, minimizing pesticide waste and preventing contamination of crops. This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship.

TABLE OF CONTENTS

1 Preface

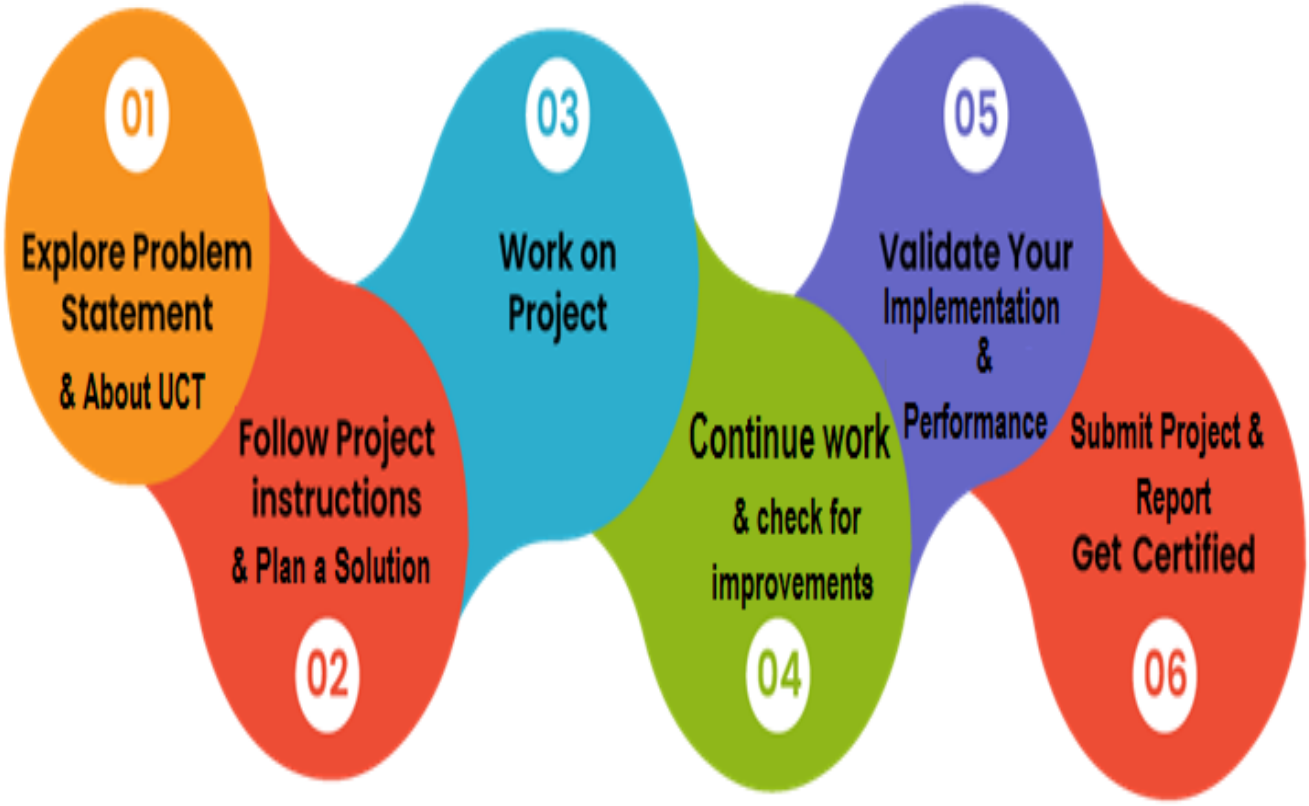
An internship plays a crucial role in career development, offering practical experience and bridging the gap between academic knowledge and real-world applications. It provides exposure to industry practices, professional networking opportunities, and hands-on skills that are often essential for job readiness.

Weeds are a major problem in agriculture, as they compete with crops for essential resources like nutrients, water, and space, leading to reduced crop yields. Farmers typically use pesticides to control weeds, but this method has its drawbacks. Pesticides can adhere to crops, posing potential health risks to consumers and contributing to environmental pollution. Additionally, indiscriminate spraying leads to pesticide waste and higher costs.

Our project focuses on developing a targeted weed detection and pesticide spraying system. By using advanced technology to differentiate between crops and weeds, the system sprays pesticides only on weeds, minimizing pesticide usage and preventing contamination of crops. This selective approach not only ensures that crops remain free of harmful chemicals but also reduces pesticide waste, making farming more sustainable and cost-effective. Ultimately, this solution aims to optimize resource usage, improve crop yields, and promote safer food production.

I received an exciting internship opportunity from USC/UCT. This opportunity offers a chance to gain valuable hands-on experience, develop new skills, and work alongside industry professionals. It's a great step forward in my career, providing both academic and practical growth in my field of interest.

How Program was planned



.

2 Introduction

2.1 About UniConverge Technologies Pvt Ltd

Established in 2013, the company specializes in digital transformation, providing industrial solutions with a strong emphasis on sustainability and return on investment (RoI). It focuses on delivering innovative and efficient solutions that drive value for businesses while promoting environmentally conscious practices. To develop its products and services, the company leverages a wide range of cutting-edge technologies. These include the Internet of Things (IoT) for smart, connected devices, robust cybersecurity measures, and cloud computing platforms like AWS and Azure for scalable infrastructure. Additionally, the company incorporates machine learning to enhance decision-making and automation, as well as advanced communication technologies such as 4G, 5G, and LoRaWAN for seamless connectivity. Development tools such as Java Full Stack, Python, and front-end technologies further support the creation of dynamic, user-friendly solutions. This comprehensive approach ensures the delivery of top-tier, sustainable solutions that enhance operational efficiency and deliver measurable RoI for clients.

i. UCT IoT Platform (uct Insight)

UCT Insight is a versatile IoT platform designed for rapid deployment of IoT applications while providing critical insights into business processes. Built with a Java backend and ReactJS front end, it supports MySQL and various NoSQL databases for flexible data management. The platform enables seamless device connectivity using industry-standard IoT protocols such as MQTT, CoAP, HTTP, Modbus TCP, and OPC UA. It offers the flexibility of both cloud and on-premises deployments.

Key features of UCT Insight include:

- Customizable dashboards for real-time monitoring

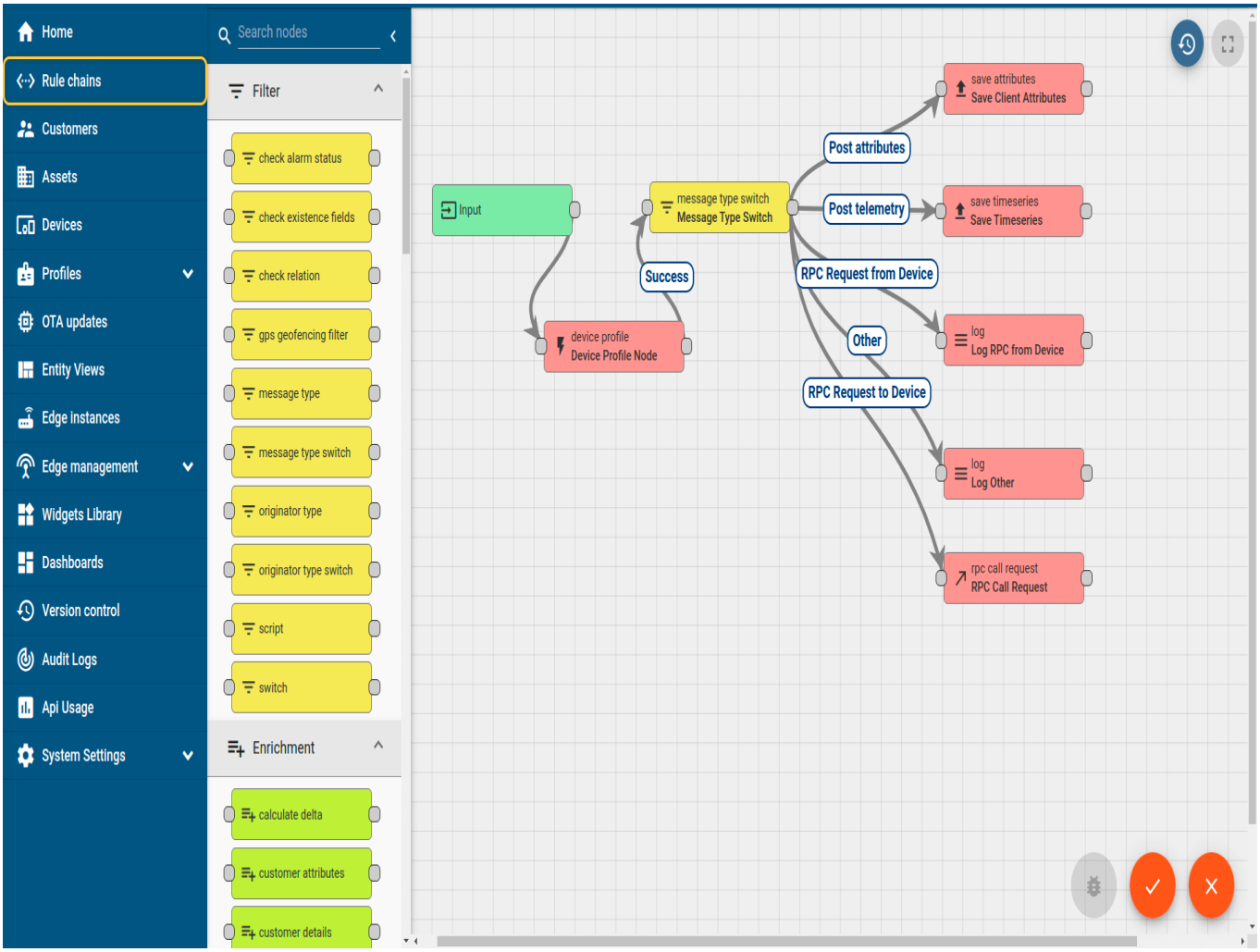
- Advanced analytics and reporting capabilities

- Alerts and notifications to stay informed

- Integration with third-party applications like Power BI, SAP, and ERP systems

- A powerful rule engine to automate processes

UCT Insight is designed to streamline IoT operations and deliver valuable insights, improving decision-making and operational efficiency.


Home
Rule chains
Customers
Assets
Devices
Profiles
OTA updates
Entity Views
Edge instances
Edge management
Widgets Library
Dashboards
Version control
Audit Logs
Api Usage
System Settings
Search nodes
Filter
check alarm status
check existence fields
check relation
gps geofencing filter
message type
message type switch
originator type
originator type switch
script
switch
Enrichment
calculate delta
customer attributes
customer details
Input
message type switch
Message Type Switch
device profile
Device Profile Node
Success
Post attributes
Post telemetry
RPC Request from Device
Other
RPC Request to Device
save attributes
Save Client Attributes
save timeseries
Save Timeseries
log
Log RPC from Device
log
Log Other
rpc call request
RPC Call Request

ii. Smart Factory Platform (FACTORY WATCH)

Factory Watch is a comprehensive platform designed to meet the needs of smart factories, offering scalable solutions for production and asset monitoring. It provides tools for Overall Equipment Effectiveness (OEE) and predictive maintenance, with the ability to scale up to digital twins for factory assets. Factory Watch helps users unlock the full potential of the data generated by their machines, enabling them to identify and enhance key performance indicators (KPIs).

With its modular architecture, users can select specific services to start with and scale up to more complex solutions as needed. The platform's unique SaaS model saves time, costs, and resources, making it a cost-effective solution for factory optimization and improved operational efficiency.

Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i

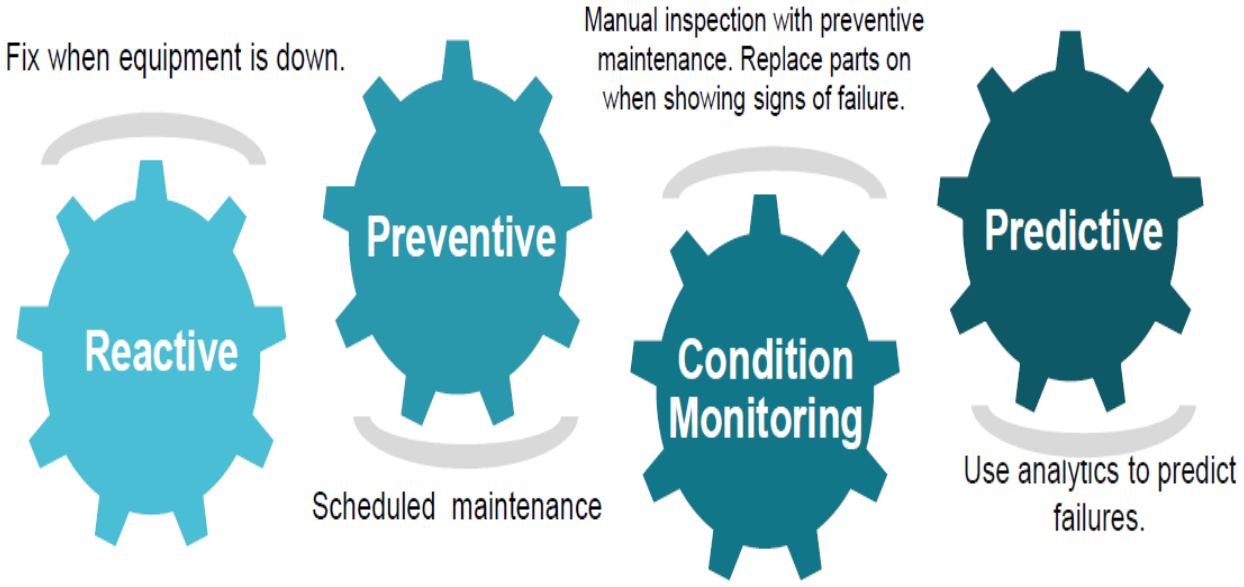
iii. LoRaWAN based Solution

UCT is an early adopter of LoRaWAN technology, offering innovative solutions across various sectors such as Agritech, smart cities, and industrial monitoring. Their expertise extends to smart street lighting, as well as smart water, gas, and electricity metering solutions. UCT's cutting-edge LoRaWAN-based solutions enhance connectivity, efficiency, and sustainability in these diverse applications.

iv. Predictive Maintenance

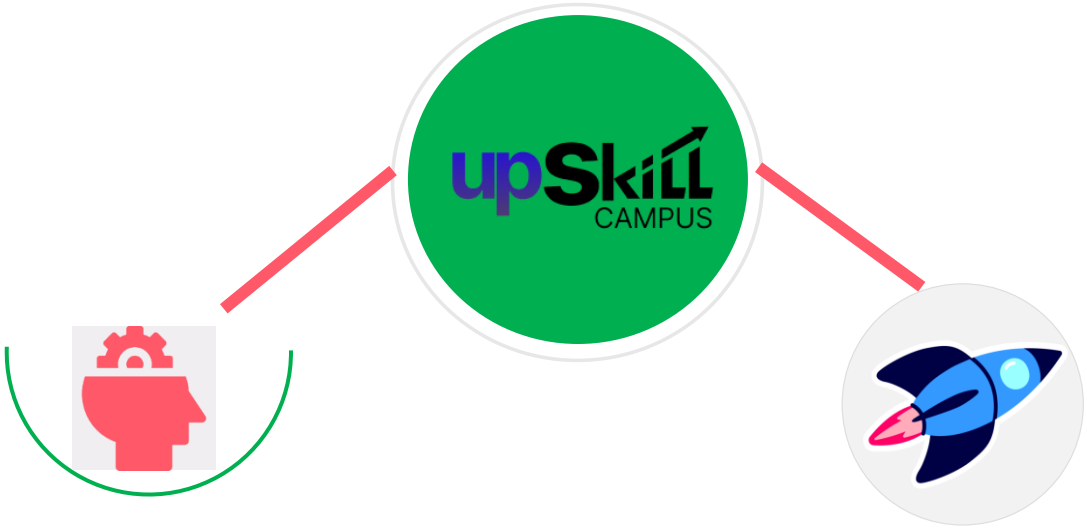
UCT offers industrial machine health monitoring and predictive maintenance solutions by utilizing embedded systems, Industrial IoT, and machine learning technologies. Their solutions focus on determining the remaining useful lifetime (RUL) of various machines used in production processes. By providing insights into machine performance and potential failures, UCT helps industries optimize maintenance schedules, reduce downtime, and improve overall efficiency.

.



2.2 About upskill Campus (USC)

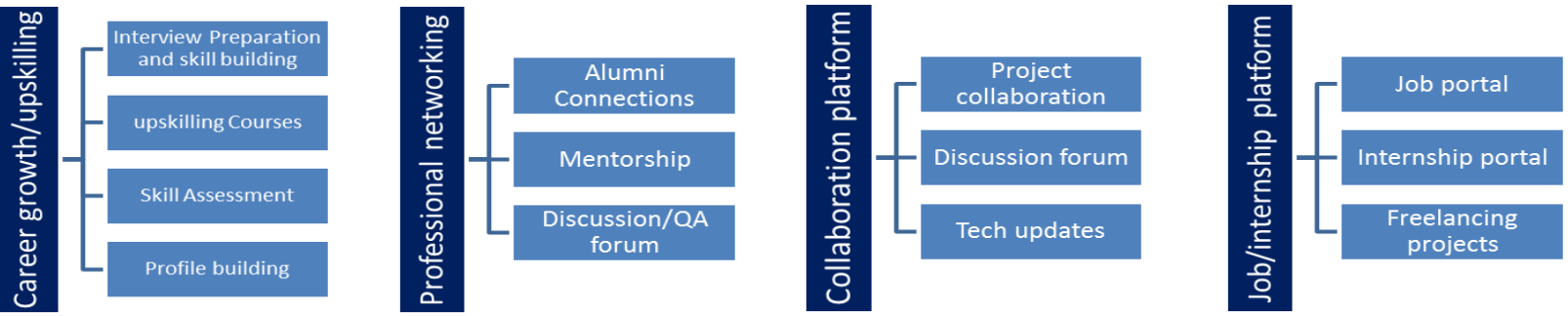
Upskill Campus, in collaboration with The IoT Academy and Uniconverge Technologies, has successfully facilitated the entire internship process. USC serves as a career development platform, offering personalized executive coaching that is more affordable, scalable, and measurable, ensuring effective support for individuals seeking to advance their careers.



Seeing need of upskilling in self-paced manner along-with additional support services e.g. Internship, projects, interaction with Industry experts, Career growth Services

Upskill Campus aiming to upskill 1 million learners in next 5 year

https://www.upskillcampus.com/



2.3 Objectives of this Internship program

The objective for this internship program was to

- ☛ get practical experience of working in the industry.
- ☛ to solve real world problems.
- ☛ to have improved job prospects.
- ☛ to have Improved understanding of our field and its applications.
- ☛ to have Personal growth like better communication and problem solving.

2.4 Reference

[1] What is Data Science | Introduction to Data Science | Data Science Training Video | TheIoTAcademy

[2] Introduction To Machine Learning | Machine Learning Course for Beginners

[3] Introducing Data Science by Devi Cielin, Arno D. B. Meysman, Mohamed Ali

[4] An Introduction to Probability and Statistics by Walter A. Shewhart and Samuel S. Wilks

2.5 Glossary

Terms	Acronym
Yolo	YOLO (You Only Look Once) is a popular object detection algorithm known for its speed and accuracy.
OpenCV	OpenCV (Open Source Computer Vision Library) is a powerful and widely-used open-source library for computer vision, machine learning, and image processing..
Pandas	**Pandas** is a popular Python library used for data manipulation and analysis. It provides data structures and functions to efficiently handle structured data, such as time series and tabular data, making it essential for data science, statistics, and machine learning tasks.
Numpy	Numpy is a fundamental library for numerical and scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a vast collection of mathematical functions to operate on these arrays efficiently.
tqdm	tqdm is a Python library used to display progress bars for loops and tasks that take time to execute.
YAML	YAML is a human-readable data serialization language that is often used for writing configuration files.
Documentation	The process of creating and maintaining written or digital materials that provide information about a software project, system, or process. Documentation serves as a reference for users, developers, and other stakeholders to understand and use the software effectively.

3 Problem Statement

Weeds are an ongoing challenge in agriculture, as they compete with crops for essential resources like nutrients, water, and land, which can lead to significantly reduced crop yields. Farmers typically rely on pesticides to control weed populations, and while this method can be effective, it has several drawbacks. Many pesticides can adhere to crops during application, posing potential health risks for consumers and contributing to environmental pollution. Furthermore, indiscriminate spraying often results in pesticide waste, leading to higher costs and a negative impact on sustainability.

To tackle this pressing issue, we aim to develop an advanced system that utilizes technology to selectively spray pesticides only on weeds, leaving crops untouched. This targeted approach not only minimizes the risk of pesticide contamination but also significantly reduces waste, ensuring that pesticides are used more efficiently. By employing techniques such as machine learning and advanced imaging, the system can accurately differentiate between crops and weeds, allowing for precise application.

Our solution is designed to enhance food safety by preventing pesticide residues on crops, thereby safeguarding consumer health. Additionally, by optimizing pesticide usage, we aim to promote more sustainable agricultural practices, reducing the overall environmental impact of farming. This innovative system will empower farmers to manage weeds more effectively, ultimately improving crop production efficiency while ensuring that their practices are both economically viable and environmentally responsible. Through this initiative, we hope to revolutionize weed management in agriculture, creating a safer and more sustainable future for the industry.

4 Existing and Proposed solution

Provide summary of existing solutions provided by others, what are their limitations?

First we load the data then resize the images 512X512X3 size, using yolo algorithm we also label the images and divide in train and test part for further process.

we create a DataFrame to read annotations from text files, capturing essential details like class (crop or weed), center coordinates, width, and height. We then generate a new DataFrame in the Pascal VOC format, calculating bounding box coordinates and appropriately labeling each class.

To aid analysis, we incorporate visualization using OpenCV, drawing bounding boxes on images and overlaying class labels. This helps evaluate the detection model's performance. The Pascal VOC DataFrame is saved as a CSV file for future use, while images and labels are organized into structured train and test folders.

Subsequently, we detail the training and exporting of a YOLOv5 model using the Ultralytics implementation. Key steps include setting up the environment, installing dependencies, training the model with specific configurations, exporting it to various formats, and running inference on new data. This comprehensive approach effectively prepares the data and model for optimized pesticide application in agricultural practices.

Then we implemented a YOLOv5-based object detection system to identify crops and weeds in agricultural images. The process begins with loading necessary libraries and configuration data from a YAML file, which defines the class labels. Subsequently, the ONNX model is loaded using OpenCV's DNN module, allowing for CPU-based inference.

The code reads an input image, resizing it into a square format suitable for YOLO. Predictions are made by processing the image through the model, yielding an output tensor that contains detection results. Non-Maximum Suppression (NMS) is applied to filter out redundant boxes based on confidence scores, retaining only the most reliable detections.

Bounding boxes are drawn on the image to visualize detected objects, accompanied by class labels and confidence scores. The resulting images are displayed for evaluation, showcasing the model's ability to accurately identify and differentiate between crops and weeds.

4.1 Code submission (Github link):

https://github.com/Pathaan/upskillcampus/blob/master/CropandWeedDetection_MdShahrukh_USC_UCT.python.zip

4.2 Report submission (Github link) :

https://github.com/Pathaan/upskillcampus/blob/master/CropandWeedDetection_Report_USC_UCT.pdf

5 YoloV5

What is YOLOv5?

YOLOv5 is a cutting-edge object detection model that belongs to the "You Only Look Once" (YOLO) family of computer vision models. Released by Ultralytics on June 25, 2020, YOLOv5 builds on the success of previous YOLO versions but introduces several enhancements, particularly in terms of training and inference speed. YOLOv5 has become one of the most popular and widely-used object detection models due to its ease of use, high performance, and compatibility with the PyTorch framework.

Object detection is a crucial task in computer vision, where the goal is to identify and locate objects within images or videos. YOLO models are designed to perform this task in real-time, making them ideal for applications where speed is critical, such as autonomous driving, surveillance, and robotics. YOLOv5 improves on the previous versions by offering faster training and inference times, while maintaining competitive accuracy.

YOLOv5 Variants

YOLOv5 comes in four main sizes: small (s), medium (m), large (l), and extra-large (x). Each variant offers a trade-off between accuracy and speed:

1. **YOLOv5s (Small)**: This version is the fastest but least accurate among the four. It is ideal for applications where inference speed is the priority, such as mobile or embedded systems, but where some loss in accuracy is acceptable.
2. **YOLOv5m (Medium)**: Offers a balance between speed and accuracy, making it suitable for many real-time applications that require a moderate level of precision.
3. **YOLOv5l (Large)**: This version offers better accuracy than YOLOv5m but takes more time to train and run inference. It is suitable for tasks where higher precision is necessary, and there is more computational power available.
4. **YOLOv5x (Extra-Large)**: This is the most accurate version of YOLOv5, but it comes with a trade-off of slower inference speed. YOLOv5x is ideal for applications where accuracy is paramount, and the model can run on powerful hardware, such as GPUs.

Each version of YOLOv5 scales the model's depth and width, allowing it to perform well on different computational resources. This modularity is one of the reasons YOLOv5 is so flexible and can be adapted to a variety of use cases.

How YOLOv5 Differs from YOLOv4

One of the most frequently asked questions about YOLOv5 is how it compares to YOLOv4, its predecessor. Although both models share similar architectural foundations, there are several key differences between them.

1. **Framework**: One of the most significant differences is that YOLOv5 is implemented in PyTorch, while YOLOv4 was implemented in the Darknet framework. PyTorch is a widely used deep learning library that offers several advantages, including easier debugging, a more extensive ecosystem, and a more user-friendly interface. PyTorch also enables quicker development, deployment, and experimentation, making YOLOv5 more accessible to a broader range of users.
2. **Training Time**: YOLOv5 is faster to train than YOLOv4. Thanks to optimizations in PyTorch, YOLOv5 significantly reduces the time it takes to train on custom datasets compared to YOLOv4, which is still heavily reliant on Darknet's framework.
3. **Model Size and Speed**: All variants of YOLOv5 (s, m, l, x) generally perform faster than YOLOv4 when it comes to inference speed, especially in real-time applications. YOLOv5's architecture is optimized for better utilization of memory and computational resources, allowing for quicker inference times without sacrificing much in terms of accuracy. In contrast, YOLOv4, while very accurate, takes longer to run inference.
4. **Ease of Use**: YOLOv5 is easier to use than YOLOv4, especially for newcomers to the field of computer vision. The YOLOv5 repository, developed by Ultralytics, offers clear documentation, tutorials, and examples, making it more accessible to a broader audience. This ease of use extends to deploying YOLOv5 on custom datasets, where the model's PyTorch base streamlines the process.
5. **Model Accuracy**: YOLOv5, particularly in its larger variants, maintains accuracy levels comparable to YOLOv4. The extra-large (x) variant of YOLOv5 is known to achieve results on par with YOLOv4 while being faster in both training and inference times. This makes YOLOv5 a strong contender for anyone looking to implement high-performance object detection models.

YOLOv5 Performance vs. Other Models

When comparing YOLOv5 to other object detection models, such as EfficientDet, YOLOv5 demonstrates several performance advantages. YOLOv5 can process images faster than EfficientDet models while maintaining a similar level of accuracy. For example, the most accurate variant, YOLOv5x, can process images multiple times faster than EfficientDet D4, making it more suitable for real-time applications. These improvements stem primarily from PyTorch-based optimizations rather than significant changes to the model architecture.

While EfficientDet focuses on optimizing model size and resource usage, YOLOv5 prioritizes speed, making it a better choice for applications where both performance and real-time processing are crucial.

Custom Training with YOLOv5

One of the major selling points of YOLOv5 is its ability to be easily adapted to custom datasets. YOLOv5's repository includes detailed guides on how to train the model on new data, which involves minimal modifications. This process is streamlined through PyTorch's intuitive interface, allowing developers to train object detectors tailored to their specific needs.

Whether you're working in agriculture, healthcare, surveillance, or any other industry, YOLOv5 can be fine-tuned to detect virtually any object with high accuracy. The model's flexibility and adaptability make it particularly appealing for developers looking to create custom object detectors without needing to write extensive amounts of code.

Evolution of YOLOv5

Since its release in 2020, YOLOv5 has undergone continuous improvements and updates. Ultralytics has been actively developing the model, adding new features and optimizations that make YOLOv5 more efficient and easier to use. The model’s performance improvements, especially in terms of speed and resource efficiency, have positioned it as a leading solution in the object detection domain.

One significant milestone in the evolution of the YOLO family is the release of YOLOv8 in January 2023, which has been described as a new state-of-the-art in object detection. YOLOv8 builds on the successes of YOLOv5 but introduces even more enhancements, particularly in terms of accuracy and robustness. Despite this, YOLOv5 remains highly relevant due to its balance of speed and accuracy and is still widely used in the industry.

Conclusion

YOLOv5 represents a major advancement in object detection technology. Its combination of speed, accuracy, and ease of use makes it an attractive choice for developers across various industries. With its PyTorch implementation, YOLOv5 simplifies training and deployment, allowing for faster development cycles and quicker results. The availability of multiple model sizes ensures that YOLOv5 can be tailored to meet specific performance and resource constraints, making it a versatile solution for both real-time and batch processing applications.

As the model continues to evolve, YOLOv5 remains one of the most reliable and efficient tools for object detection, offering users the ability to train custom models with minimal effort while maintaining competitive performance levels.

6 Model

1. Setup and Directory Navigation:

- I changed the working directory to '/content/drive/MyDrive/Yolo Training '.
- I cloned the YOLOv5 repository from GitHub into this directory, which includes all the necessary files for training and running YOLOv5 models, such as train.py, detect.py, val.py, configuration files, and more.

2. YOLOv5 Training Command:

- I navigated to the yolov5 directory and started training YOLOv5 using:

  "python train.py --data data.yaml --cfg yolov5s.yaml --batch-size 8 --name Model --epochs 50"

- This command specifies:
  - --data data.yaml: Specifies the dataset configuration (path, labels, etc.).
  - --cfg yolov5s.yaml: Specifies the YOLOv5 model architecture (YOLOv5s, in this case).
  - --batch-size 8: Batch size for training.
  - --name Model: The name for the training run (saved under runs/train/Model/).
  - --epochs 50: Training for 50 epochs.
- **Training Output**:
  - I receive training output showing details like loss values (box_loss, obj_loss, cls_loss), memory usage, and performance metrics (precision, recall, mAP) per epoch.

3. Training Results:

- The output shows training progress for each epoch with metrics like:
  - **Box Loss**: Measures how well the model predicts bounding boxes.
  - **Objectness Loss**: Measures how well the model identifies whether there is an object in a predicted bounding box.
  - **Class Loss**: Measures how well the model classifies the object (weed, crop, background).
  - **Precision, Recall, mAP (Mean Average Precision)**: Performance metrics indicating how well the model is detecting and classifying objects.
- At the end of training, I get a summary with values for:
  - Precision: 0.85
  - Recall: 0.863
  - mAP50 (mean average precision at 0.5 IoU): 0.889

- mAP50-95: 0.632
- This means my model is performing quite well in detecting and classifying objects (weeds/crops) after training.

4. Model Export:

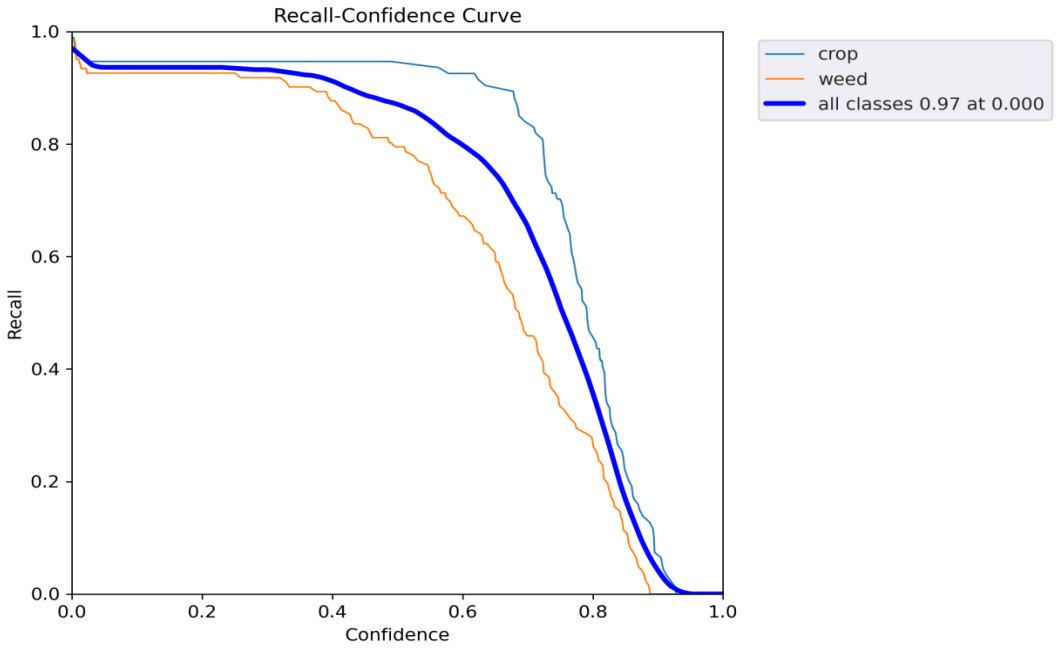
- After training, I ran the following command to export the trained model:

  "python export.py --weights runs/train/Model2/weights/best.pt --include torchscript onnx"

- This command exports the trained YOLOv5 model (best.pt) into multiple formats:
  - **TorchScript**: A format for running PyTorch models in production environments.
  - **ONNX**: Open Neural Network Exchange format, which allows interoperability between different frameworks.
- **Export Output**:
  - The TorchScript model is saved as best.torchscript (27.2 MB).
  - The ONNX model is saved as best.onnx (27.2 MB).
  - During the export process, there was a warning that the required onnx>=1.12.0 was not installed, but the script automatically updated and installed onnx 1.17.0.

7 Performance Test

1. **Recall-Confidence Curve:**



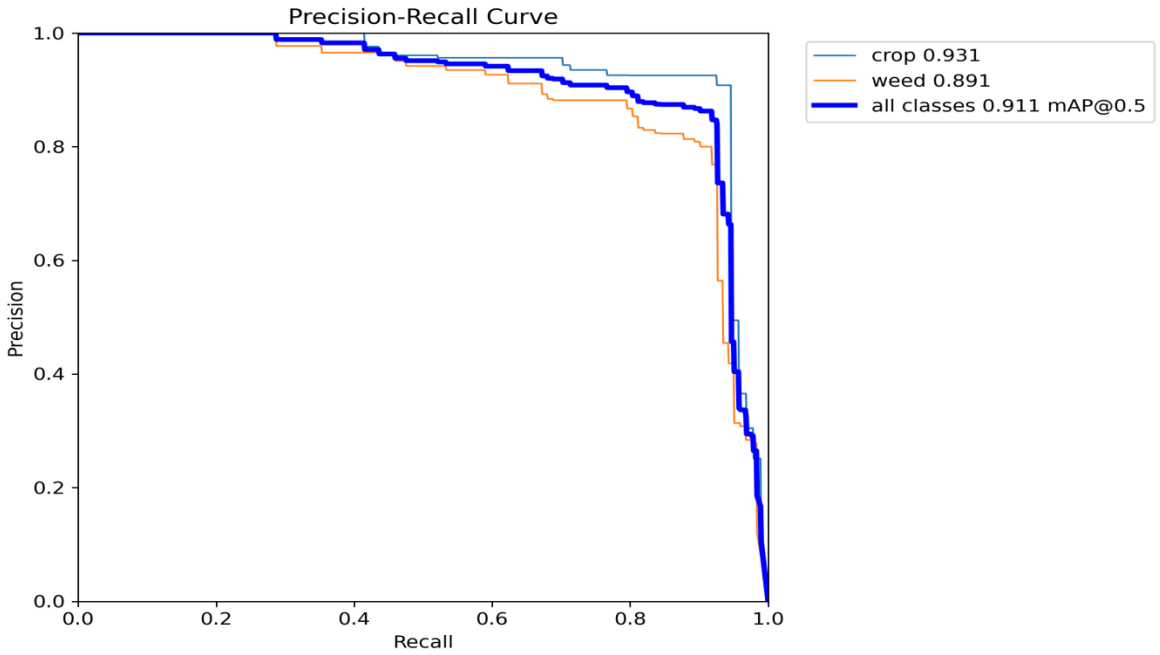
High recall at low confidence: At the lowest confidence threshold (close to 0), the model captures nearly all instances of crops and weeds, with recall close to 1. However, this high recall may come at the cost of **precision**, meaning the model makes a lot of predictions, some of which may be incorrect (false positives).

Lower recall at higher confidence: As the confidence threshold increases, recall decreases. This is because the model becomes more conservative, only making predictions when it is very sure. While this reduces false positives, it also leads to **missed true positives** (fewer correct detections).

Crop vs. Weed performance: The weed recall drops off more steeply than the crop recall. This means the model is more likely to miss weeds as the confidence level rises compared to crops.

This graph shows that while the model achieves high recall at lower confidence thresholds, as the confidence increases, it sacrifices recall, particularly for weeds. If recall is critical, then using a lower confidence threshold might be preferable. However, this must be balanced against precision, as seen in the previous Precision-Confidence Curve.

2. Precision-Recall Curve



Light blue curve (Crop): Represents the PR curve for detecting **crop**, with an average precision of **0.931**. This suggests the model performs well in identifying crops, with a good balance between precision and recall.

Orange curve (Weed): Represents the PR curve for detecting **weed**, with an average precision of **0.891**. This is slightly lower than for crops, indicating the model is somewhat less precise or has lower recall when identifying weeds.

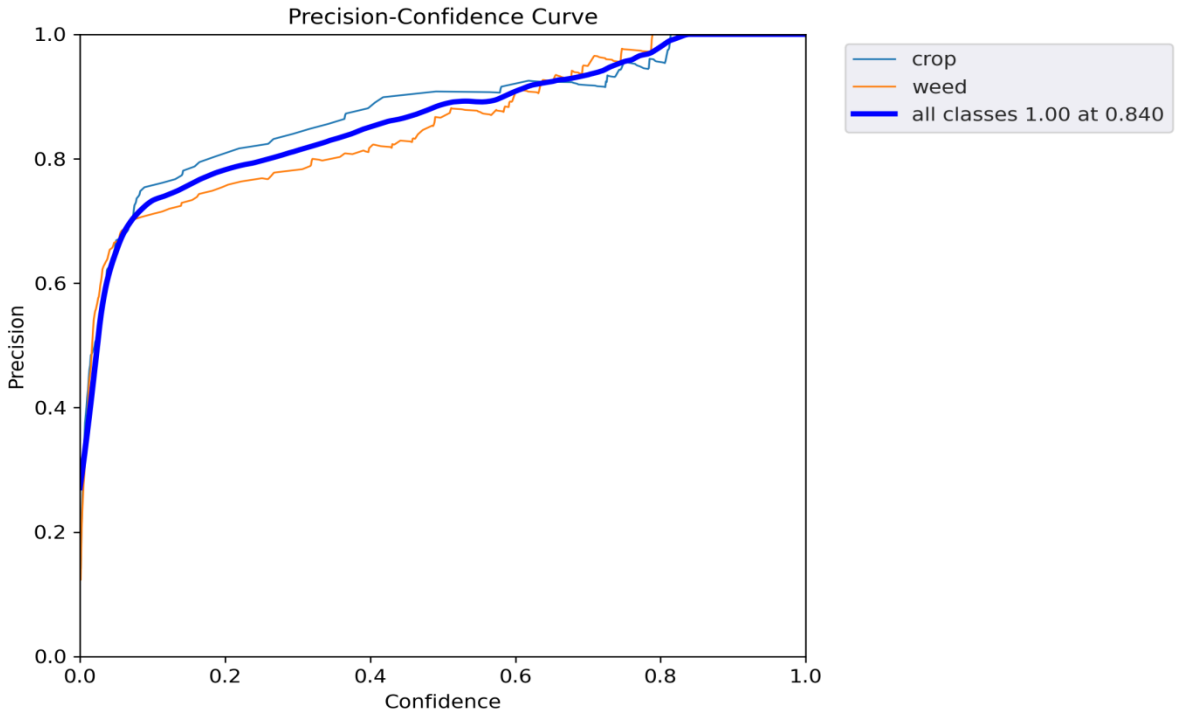
Dark blue curve (All classes): This thicker line shows the **mean Average Precision (mAP) for all classes** (crop and weed) at a confidence threshold of 0.5. The mAP of **0.911** summarizes the model's overall performance across both classes. It shows how well the model balances precision and recall across all classes.

The **Crop** class has a higher precision-recall performance compared to the **Weed** class. This could indicate that the model is more confident in distinguishing crops than weeds.

The **All classes mAP of 0.911** at a threshold of 0.5 indicates a strong overall model performance, though there is still room for improvement, especially in detecting weeds.

The curves are quite high overall, meaning your model generally does well in maintaining a high level of precision while recalling most of the positive instances across both categories (crops and weeds).

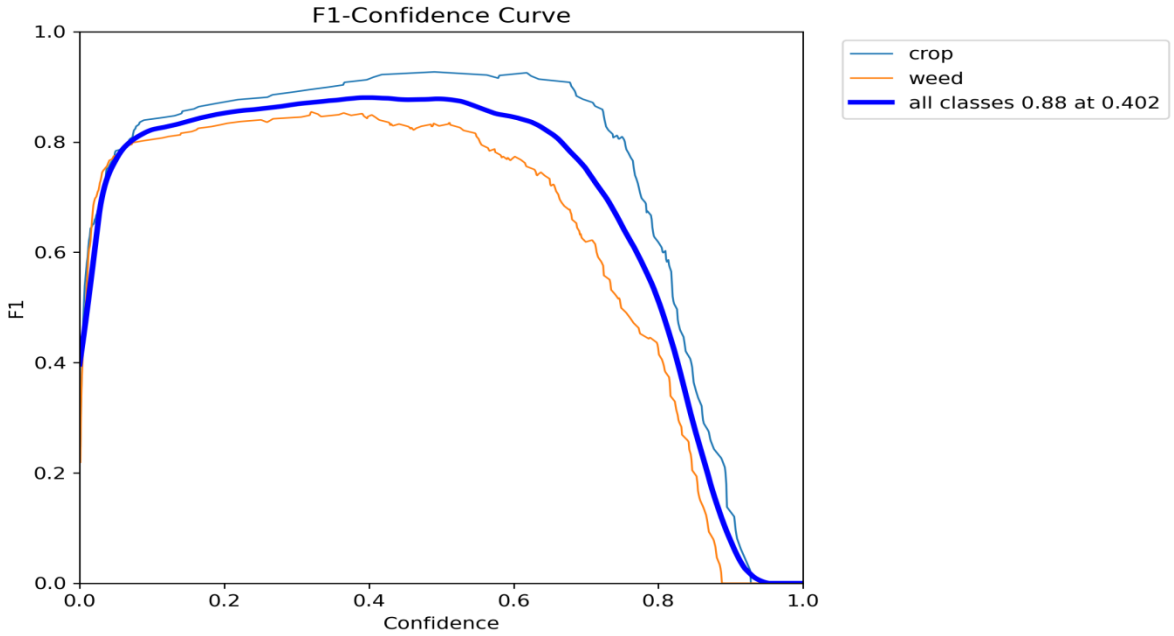
3. Precision-Confidence Curve



The graph indicates that as the confidence threshold increases, the model's precision improves for both crops and weeds. At a confidence level of **0.840**, the model achieves **100% precision** across all classes, meaning that all the predictions it makes at this confidence level are correct. However, a very high confidence threshold could lead to fewer predictions being made (since the model only predicts when it's very sure), potentially reducing recall.

This curve helps in understanding how confident the model should be to maximize precision in the classification task, which is critical in applications like crop and weed detection where incorrect classifications could be costly (e.g., spraying herbicide on crops instead of weeds).

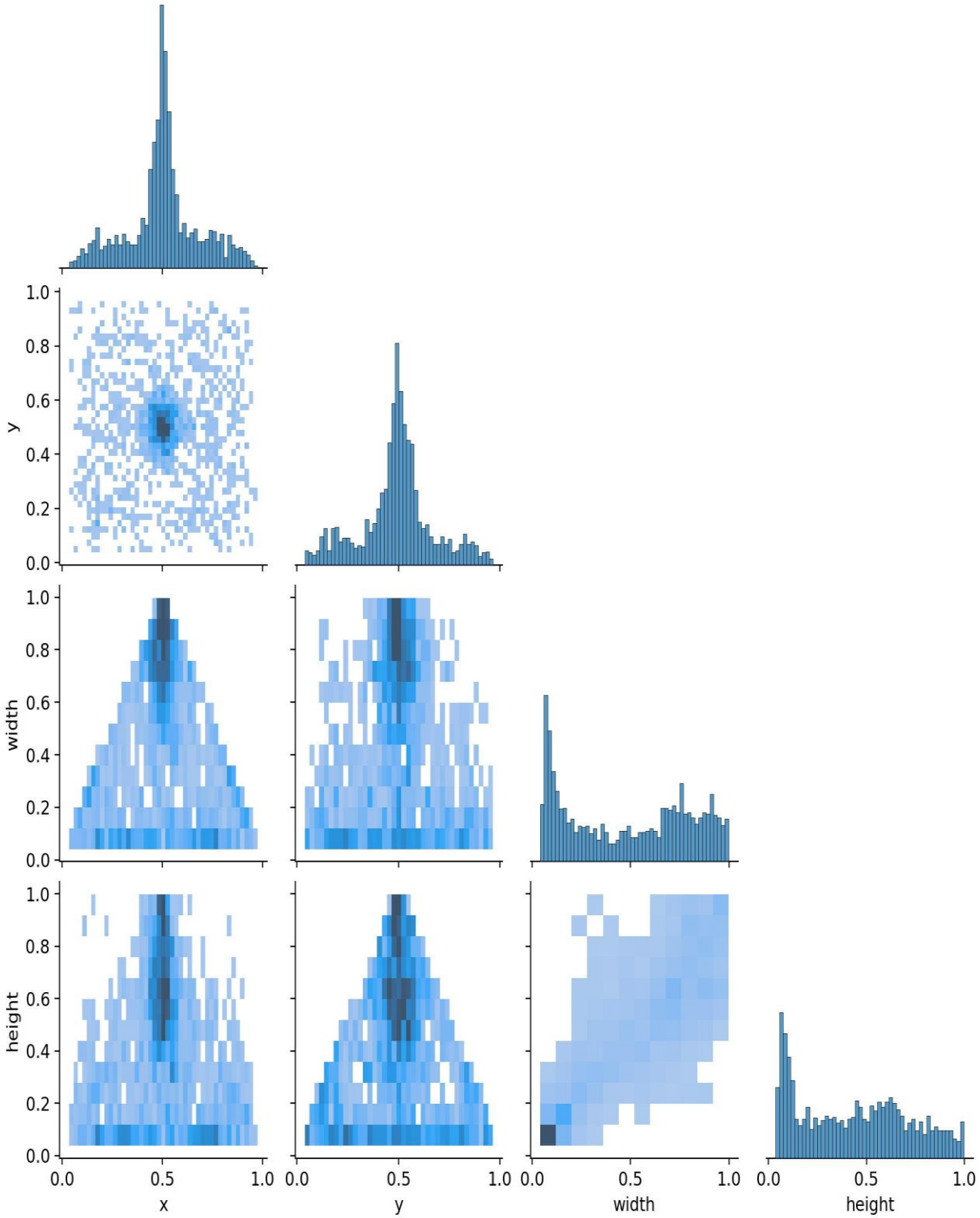
4. F1-Confidence Curve:



1. **X-axis (Confidence):** This represents the confidence level thresholds. As the confidence increases from 0 to 1, the model becomes more certain about its predictions.
2. **Y-axis (F1 Score):** The F1 score, ranging from 0 to 1, combines both precision and recall to give a balanced evaluation of the model's performance. Higher F1 scores indicate better model performance.

Crop class (light blue curve): As the confidence threshold increases from 0 to around 0.6, the F1 score also improves, meaning the model gets better at detecting crops. After a certain point (around confidence 0.7), the F1 score starts dropping. This indicates that as the model becomes more selective (choosing only very confident predictions), it might miss some crops, reducing the overall performance.
Weed class (orange curve): The weed curve shows a similar trend but peaks at a lower F1 score compared to crops. This suggests the model struggles more with detecting weeds than crops, particularly as the confidence threshold rises. The F1 score decreases faster for weeds, meaning the model sacrifices performance with higher confidence.
Both classes (thick blue curve): This is the combined performance for both crops and weeds. The best F1 score across both classes is **0.88**, achieved at a confidence level of **0.402**. This means that at this threshold, the model strikes the best balance between precision and recall for both crop and weed detection.

5. Labels Correlogram



Variables on Axes:

The plot displays four variables:

x: Likely representing the x-coordinate of an object, perhaps in an image.
y: Likely representing the y-coordinate.
width: Likely representing the width of the object (bounding box).
height: Likely representing the height of the object (bounding box).

Components of the Plot:

- **Histograms along the diagonal**: These show the **distribution** of each variable.
  - The **x-coordinate** is somewhat centralized, peaking around 0.5.
  - The **y-coordinate** also centers around 0.5 but with more variability.
  - The **width** and **height** have broader distributions, with noticeable peaks around 0.2-0.3.
- **Scatter plots or 2D density plots** on the off-diagonal: These show pairwise relationships between variables.
  - **x vs. y**: Appears to be a scatter plot with most points concentrated around the center of the image (around 0.5 for both x and y), indicating many objects are detected near the center.
  - **x vs. width** and **x vs. height**: The plots form a triangular shape, suggesting some relationship between the x-coordinate and the width/height of the bounding boxes. Objects near the center tend to have a variety of sizes, but larger boxes are less common as you move away from the center.
  - **y vs. width** and **y vs. height**: Similar trends as x vs. width/height, but with slightly different density distributions.
  - **width vs. height**: which showing that larger width values often correspond to larger height values, with a concentration around smaller widths and heights.

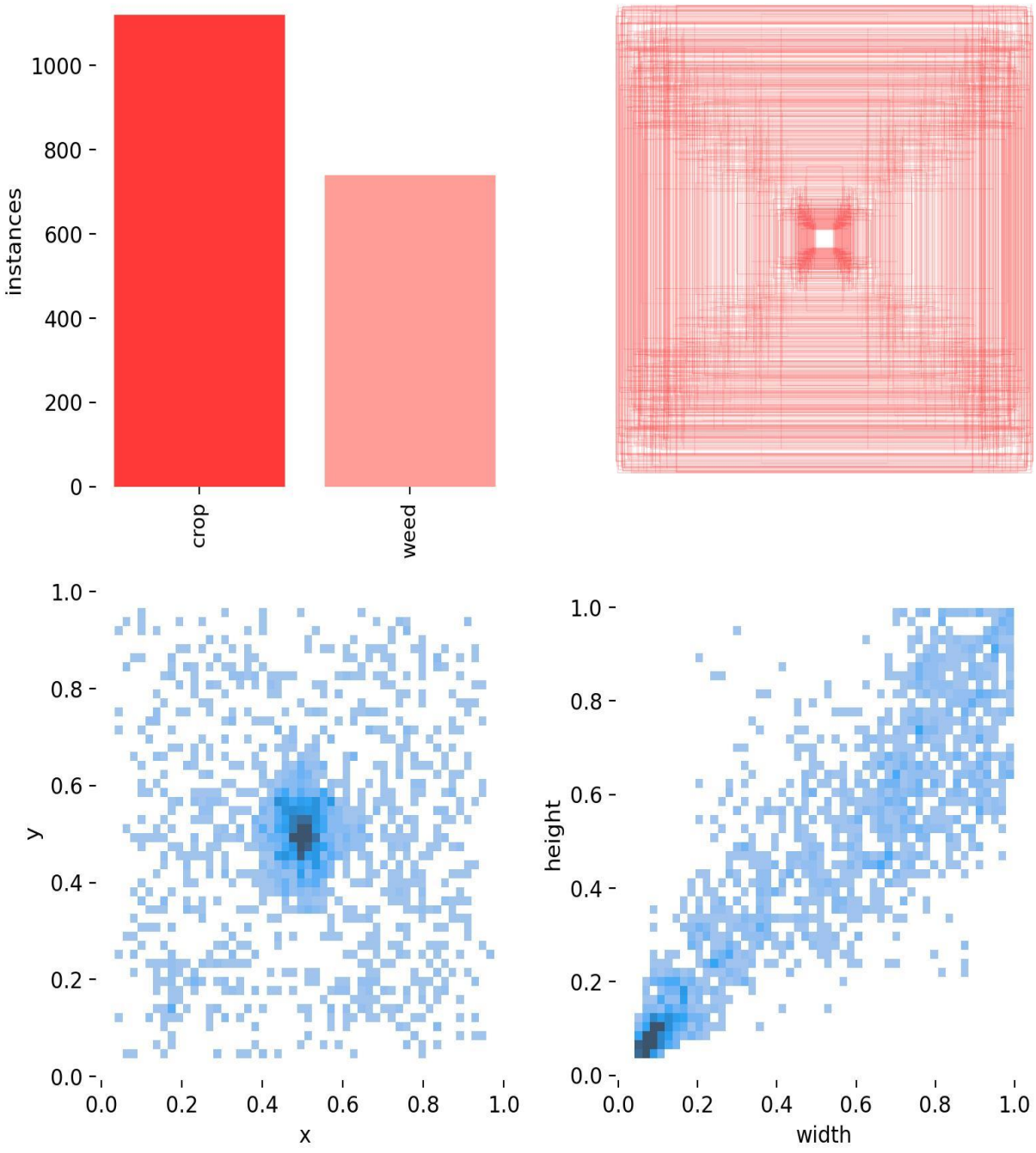
Interpretation:

The **central concentration** of x and y values suggests that the objects (likely crops or weeds) are most commonly found near the center of the image, or that the bounding boxes are centered there.

The **width and height** distributions indicate that most bounding boxes are relatively small (peaking around 0.2-0.3) but can vary significantly in size.

The pairwise relationships, especially between width and height, suggest that there may be a correlation between the size of detected objects.

6. Labels


instances
1000
800
600
400
200
0
crop
weed
y
x
height
width
0.0
0.2
0.4
0.6
0.8
1.0

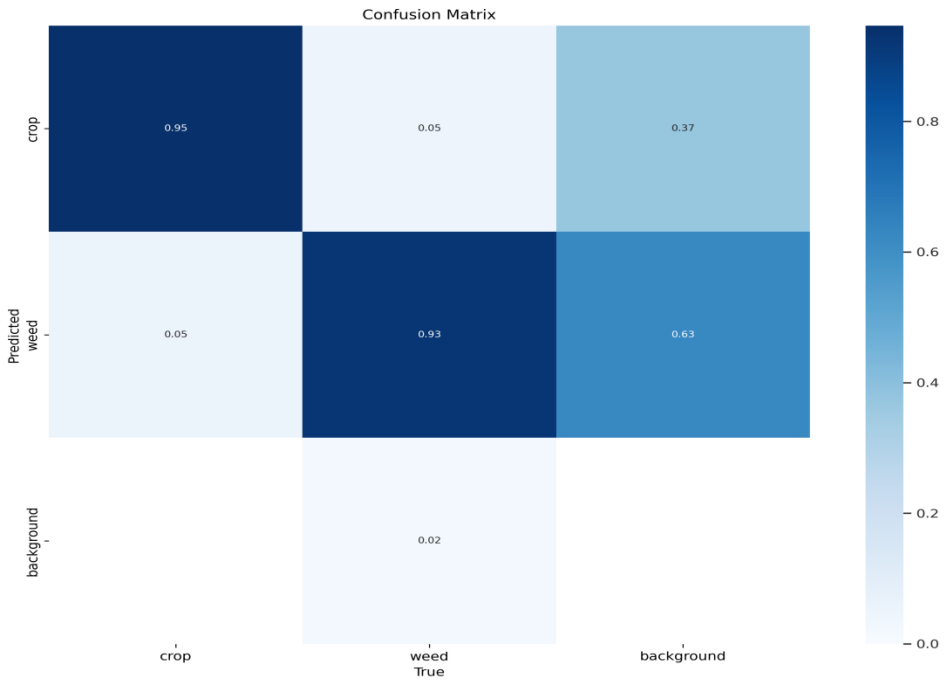
1. **Class Distribution (Top-left)**: The bar chart compares the number of instances between crops and weeds, showing a clear imbalance. The dataset contains more crop instances (over 1000) compared to weed instances (around 700-800). This imbalance may affect model performance, as models tend to favor classes with more examples. To improve the detection of weeds, techniques like oversampling or class-weighted loss functions might be required.

2. **Bounding Box Distribution (Top-right)**: The red overlay of bounding boxes illustrates where objects (crops and weeds) are located within the images. Most bounding boxes are centered around the middle of the image, as indicated by the dense pattern near the center. This could mean that the camera capturing the data focuses on objects near the center, or that the crops and weeds themselves are spatially concentrated in the middle of the images.

3. **Object Location (Bottom-left)**: The scatter plot showing the relationship between x and y coordinates confirms the findings from the bounding box distribution. There's a notable clustering of objects around the image center ($x \approx 0.5$, $y \approx 0.5$). However, the scatter plot also shows objects spread throughout the entire image, though in lower densities.

4. **Size of Objects (Bottom-right)**: This scatter plot displays the relationship between bounding box width and height. A positive correlation is observed, indicating that larger objects tend to maintain proportional width and height. Most bounding boxes have smaller dimensions, concentrated around width and height values of 0.2 to 0.3, likely reflecting the typical size of crops and weeds in the dataset.

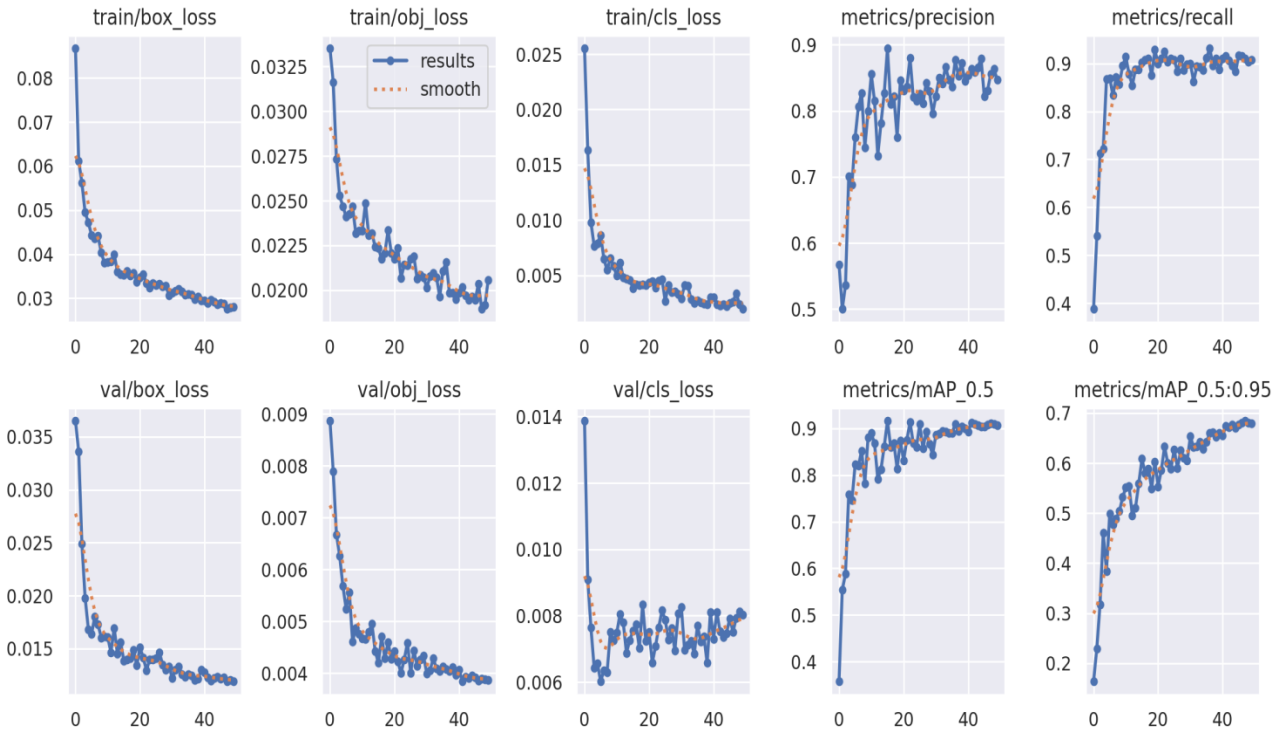
7. Confusion Matrix



Key Observations:

1. **Crop Classification:**
   - **True Positives:** The model correctly classified **95%** of the crop images (true crop predicted as crop).
   - **False Negatives:** Only **5%** of crops were incorrectly predicted as weeds.
   - **Misclassification with Background:** Notably, **37%** of the crop images were misclassified as background, suggesting that distinguishing crops from the background poses a challenge.
2. **Weed Classification:**
   - **True Positives:** The model identified **93%** of the weed images correctly.
   - **False Positives (Crop vs. Weed): 5%** of weeds were mistakenly classified as crops.
   - **Misclassification with Background:** The model misclassified **63%** of weeds as background, indicating significant overlap between weeds and background in the feature space.
3. **Background Classification:**
   - **True Positives:** The model correctly identified **63%** of the background areas.
   - **False Positives (Weed vs. Background):** A relatively small percentage (**2%**) of the background was misclassified as weeds, indicating the model's capacity to differentiate these two classes.

8. Results



Training/Validation Losses: Both training and validation losses (box, obj, and cls) decrease smoothly, indicating that the model is learning well and there is no significant overfitting.

Precision and Recall: The model achieves a high level of both precision and recall (around 0.9), demonstrating it correctly identifies and classifies objects in most cases.

mAP Scores: The high mAP (0.9 for IoU 0.5 and 0.7 for IoU 0.5-0.95) suggests the model is highly accurate in detecting and classifying objects, even under more stringent evaluation metrics.

8 My learnings

During my internship program, I had the opportunity to learn a wide array of skills and concepts in the fields of data science and machine learning. I significantly deepened my understanding of key topics such as machine learning, statistics, deep learning, and data science, which provided a strong foundation for more advanced work. The program helped me clarify concepts that were previously challenging, particularly in the areas of model training, evaluation, and statistical methods used to interpret results.

Additionally, working on project gave me hands-on experience in several important technologies and algorithms. I learned to implement the YOLO (You Only Look Once) algorithm, a powerful real-time object detection method that enhanced my understanding of computer vision tasks. This project also introduced me to OpenCV, a widely-used library for image processing, which I found incredibly useful in handling visual data efficiently.

Through this experience, I further honed my Python programming skills, applying them in a range of tasks such as data preprocessing, model development, and object detection. The combination of learning new algorithms and applying them in practical settings helped solidify my skills in both machine learning and data science.

Overall, the internship was a valuable learning experience that not only reinforced my theoretical knowledge but also provided me with the practical skills to implement advanced techniques in real-world scenarios. The exposure to cutting-edge tools and methodologies has greatly boosted my confidence and prepared me for future challenges in the field of machine learning and data science.

9 Future Work Scope

While developing the object detection machine learning project for crop and weed detection, several areas for future improvement and expansion were identified. Here are some potential areas for further work and enhancements that could significantly improve the model's performance and broaden its application:

1. **Improved Detection Accuracy and Robustness**: The current implementation may face challenges with detecting objects in complex environments, such as varying lighting conditions, occlusions, and overlapping objects. Future work could focus on fine-tuning the model architecture, experimenting with different pre-trained models, or employing techniques like transfer learning to enhance detection accuracy. Incorporating ensemble models or advanced post-processing methods could also increase robustness.
2. **Real-Time Detection and Optimization**: To make the system suitable for real-world applications, it could be optimized for real-time detection in resource-constrained environments, such as embedded systems or edge devices. This would involve model compression techniques like quantization, pruning, or using lightweight architectures such as YOLOv5-nano or MobileNet, ensuring high-speed detection without sacrificing accuracy.
3. **Cross-Platform Compatibility**: Currently, the system is designed for desktop environments using Python-based frameworks. Expanding the project to other platforms, such as mobile applications (iOS, Android), web-based interfaces, or IoT devices, would allow users to access the detection system in various agricultural settings seamlessly. This could involve converting the model for use on mobile or web frameworks using TensorFlow Lite, ONNX, or WebAssembly.
4. **Cloud-Based Storage and Processing**: Cloud integration would enable the system to store and process large volumes of data, making it scalable for larger farms or regions. This could include real-time synchronization of detection results, cloud storage of images and labels, and leveraging cloud-based services like AWS Lambda or Google Cloud AI to perform detection tasks remotely. This would reduce the computational burden on local devices.
5. **Model Generalization and Adaptability**: The current dataset used for training the model is specific to certain crop and weed types. Future work could focus on improving the model’s generalization capabilities by expanding the dataset to include different crops, weeds, and environmental conditions across various regions. This would make the system adaptable to a wider variety of agricultural applications, increasing its practical use.
6. **Automated Weed Control Integration**: Beyond detection, future iterations of the project could integrate automated weed control mechanisms, such as connecting the object detection model to robotic systems or drones capable of spraying herbicides or performing mechanical weeding. This could lead to fully autonomous weed management systems, reducing the need for human intervention.

7. **Explain-ability and Interpretability of Results**: As object detection models are often treated as black boxes, incorporating model interpretability tools could help users understand why certain crops or weeds were detected. This would build user confidence in the system and provide insights for improving model performance. Techniques like Grad-CAM or feature visualization could be used to highlight regions of the image that contributed most to the model's predictions.
8. **User-Friendly Interface and Reporting**: A user-friendly graphical interface for non-technical users would enhance the accessibility of the system. Future work could involve developing a dashboard that provides visualizations of detection results, crop health trends, and suggestions for optimizing farm management. Additionally, periodic reports could be generated based on detection data to assist in decision-making for farmers and agronomists.

By focusing on these areas, the object detection system for crop and weed detection could become more robust, scalable, and useful in a wide range of agricultural scenarios, ultimately benefiting the broader agricultural community.