Real-time detection and classification of plant seeds using YOLOv8 object detection model

by

Pavan Kumar Pativada

B.Tech, Jawaharlal Nehru Technological University(JNTUK), 2019

A REPORT

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2024

Approved by:

Major Professor
Dr. Mitchell L. Neilsen

Copyright

Abstract

This study addresses the pressing need for efficient and accurate plant seed detection and classification in agriculture by leveraging deep learning and computer vision technologies. Traditional manual methods for seed identification are time-consuming and labor-intensive, prompting the adoption of deep learning-based object detection techniques. This study focuses on implementing a real-time plant seed detection and classification system using the You Only Look Once (YOLO) v8 object detection model.

The project workflow involves collecting a custom dataset of high-resolution images of various plant seeds, annotating them with bounding boxes, and preprocessing them to ensure uniformity and increase variability through data augmentation. The dataset is then partitioned into training, testing, and validation sets for model training and evaluation. The YOLOv8 model is trained on the annotated dataset, and its performance is rigorously evaluated.

Additionally, related works in the field of seed classification and quality testing using deep learning techniques are reviewed, providing valuable insights into the application of such methodologies in agricultural practices. Experimental results demonstrate the effectiveness of the proposed approach, with the YOLOv8 model achieving high precision and recall rates.

The deployment of the trained model on the Roboflow platform enables real-time seed detection and classification, showcasing its potential for practical agricultural applications. Overall, this study contributes to the advancement of agricultural automation and precision farming, offering a scalable and efficient solution for plant seed detection and classification that holds significant implications for improving crop management strategies.

Table of Contents

List of Figures vi
List of Tables vii
Acknowledgments viii
Chapter 1 - Introduction 1
1.1 Problem Definition 1
1.2 Objective 1
Chapter 2 – Related Work 3
2.1 Literature Survey 3
2.2 Established Methods and Approaches 5
2.2.1 YOLO 5
2.2.2 YOLOv5 and YOLOv8 6
2.2.3 Roboflow 7
Chapter 3 - Implementation 9
3.1 Overview of the Data 9
3.2 Dataset Preparation 9
3.2.1 Collection and Labelling 10
3.2.2 Data Pre-processing 12
3.2.3 Data Augmentation 13
3.3 Implementation Steps 14
Chapter 4 – Experiments 16
4.1 Training, Validation, and Test Data Split 16
4.2 Experimental Design 17
4.2.1 YOLOv5 17
4.2.2 YOLOv8 18
4.2.3 Deploying in Roboflow 19
Chapter 5 – Experimental Results 21
5.1 Evaluation Metrics 22
5.2 Mean Average Precision 23
5.3 Precision and Recall 25

5.4 Confusion Matrix 27
Chapter 6 – Summary and Future Work 31
6.1 Summary of Results 31
6.2 Future Scope 32
References 34

List of Figures

Figure 1 - YOLO Architecture (Redmon, et., al 2016) 6
Figure 2 - Canon EOS R6 Mark II Camera 10
Figure 3 - Sample images from the Dataset. 12
Figure 4 - Images after Data Augmentation (after flipping, rotating & hue) 14
Figure 5 - Project Pipeline 15
Figure 6 - Real-time detection samples from Roboflow 20
Figure 7 - YOLOv5 Final results 21
Figure 8 - YOLOv8 Final Results 22
Figure 9 - mAP comparison of YOLOv5 and YOLOv8 23
Figure 10 - YOLOv5 F1 Confidence Curve 24
Figure 11 - YOLOv8 F1 Confidence Curve 25
Figure 12 - YOLOv5 Precision and Recall Curve 26
Figure 13 - YOLOv8 Precision and Recall Curve 27
Figure 14 - YOLOv5 Confusion Matrix 28
Figure 15 - YOLOv8 Confusion Matrix 29
Figure 16 - Validation batch results 30

List of Tables

Table 1 - Performance comparison of YOLOv5 and YOLOv8 (Jocher, et al., 2023)..... 7
Table 2 - Comparison of YOLOv5 and YOLOv8 results..... 23

Acknowledgments

First and foremost, I wish to convey my heartfelt appreciation to my committee members Dr. Mitchell Neilsen, Dr. Torben Amtoft, and Dr. Lior Shamir for generously offering their time and support to serve on my committee throughout this project. I am particularly grateful to my Major Professor, Dr. Mitchell Neilsen, for his unwavering belief in my capabilities and consistent support since my second semester at Kansas State University. I also extend my thanks to Dr. Charles Rice and the RAIN team for their invaluable mentorship during my tenure as a Graduate Research Assistant.

I owe a tremendous debt of gratitude to my family members, whose unwavering support made this journey possible. To my mother, Aruna Kumari Pativada, father, Bala Suryanarayana Pativada, brother, Ravi Teja Pativada, and my uncle, Suresh Gedala, I am deeply thankful for their boundless love, encouragement, and emotional guidance, which have been indispensable in my life and achievements thus far.

Lastly, but certainly not least, I express my appreciation to my friends Rahul Karne, Akhil Dudhipala, Suhasnadh Reddy, Naveen Jonnalagadda, Ramya Kalam, Sai Teja, Akhil Joshi, Hemanth for their unwavering support and companionship. These past two years have been immensely enjoyable and memorable, thanks to their friendship and encouragement.

Chapter 1 - Introduction

1.1 Problem Definition

The rapid advancements in computer vision technology have enabled new avenues for addressing critical challenges in agriculture, particularly in the domain of plant seed detection and classification. Accurate identification of plant seeds is crucial for various agricultural processes, including crop monitoring, yield estimation, pest management, and especially mixed cropping. Traditional methods of seed identification often rely on manual inspection, which is time-consuming and labor-intensive. In recent years, deep learning-based object detection techniques have emerged as promising solutions for automating this task, offering the potential for real-time, accurate seed detection and classification.

1.2 Objective

This report presents a comprehensive study on the development and implementation of a real-time plant seed detection and classification system using the You Only Look Once (YOLO) v8 object detection model. The project aims to address the limitations of traditional seed identification methods by leveraging the capabilities of deep learning and computer vision techniques.

The project workflow begins with the collection of a custom dataset comprising high-resolution images of various plant seeds, including soybean, paddy, peas, and wheat. These images are captured using an EOS R6 Mark II Camera under controlled environmental conditions to ensure data quality and consistency. Subsequently, each image is annotated with

bounding boxes using the LabelImg detector tool, providing ground truth labels for model training.

To enhance model performance and robustness, extensive preprocessing steps are applied to the dataset. This includes auto-orientation and resizing of images to ensure uniformity, as well as data augmentation techniques such as flipping, rotating, and hue adjustment to increase dataset variability and improve model generalization.

The custom dataset is then partitioned into training, testing, and validation sets to facilitate model training and evaluation. The YOLOv8 object detection model is employed for training on the annotated dataset, leveraging its state-of-the-art architecture and efficiency for real-time inference.

Upon completion of model training, the performance of the trained model is rigorously evaluated through verification and testing procedures. Finally, the trained model is deployed on Roboflow, a platform for managing and deploying computer vision models, enabling real-time seed detection and classification in agricultural applications.

This study contributes to the growing body of research in agricultural automation and precision farming, offering a scalable and efficient solution for plant seed detection and classification. The findings and insights derived from this project hold significant implications for improving agricultural practices and enhancing crop management strategies.

Chapter 2 – Related Work

2.1 Literature Survey

In the research paper "Seeds Classification and Quality Testing Using Deep Learning and YOLO v5", Kundu, Rani, and Dhaka propose a system called the "Mixed Cropping Seed Classifier and Quality Tester (MCSCQT)" that utilizes the YOLO v5 algorithm for object detection (Kundu et al., 2021). They focus on automating seed classification and quality testing, specifically for maize and pearl millet seeds grown together in mixed cropping. The dataset used comprises images of healthy and diseased maize seeds, pearl millet seeds, and clustered images of both. The YOLO v5 model architecture is detailed, highlighting its backbone, neck, and head components for feature extraction, pyramid generation, and final detection. The study demonstrates high precision and recall rates of 99% for classifying seeds and segregating healthy and diseased maize seeds. The YOLO v5 model is praised for its efficiency in detecting objects and its faster processing speed compared to other models. The paper also discusses the experimental setup, training resources, hyperparameters, and the superior performance of YOLO v5 over the K-means algorithm in seed classification. This research provides valuable insights into the application of deep learning and object detection algorithms in agricultural practices, showcasing the potential for automation and efficiency in seed classification and quality testing processes.

In the research paper "Classification of Seeds using Domain Randomization on Self-Supervised Learning Frameworks", Margapuri and Neilsen focus on seed phenotyping using convolutional neural networks (CNNs) and self-supervised learning frameworks (Margapuri, et al., 2021). They address the challenges of requiring a large volume of labeled data for training CNNs by proposing the use of domain randomization and contrastive self-supervised learning.

The study evaluates the performance of three self-supervised learning frameworks (SimCLR, Momentum Contrast (MoCo), and BYOL) against a supervised learning model (ResNet-50) on synthetic image datasets of various seed types (canola, rough rice, sorghum, soy, and wheat). The results show that MoCo achieves an accuracy of 77% on the test dataset, which is only 13% less than the accuracy of ResNet-50 trained on 100% of the labels. This demonstrates the feasibility of domain randomization for seed classification using self-supervised learning frameworks. The study highlights the potential of using synthetic datasets and the effectiveness of self-supervised learning techniques in seed phenotyping applications.

In their study detailed in “YOLO9000: Better, Faster, Stronger”, researchers delved into the realm of tomato disease detection using a variety of deep learning models including VGG16, ResNet50, and ResNet101, coupled with faster R-CNN (Redmon et al., 2016). Among these models, ResNet101 emerged as the standout performer, boasting a remarkable mean Average Precision (mAP) of 90.87% on a dataset comprising 207 tomato images. This finding underscores the efficacy of ResNet101 in accurately identifying diseased tomatoes, showcasing its potential for real-world agricultural applications.

Similarly, in another research endeavor outlined in the research paper “Tomato Diseases and Pests Detection Based on Improved Yolo V3 Convolutional Neural Network.”, a refined version of the YOLO V3 model was explored for object detection tasks (Liu et al., 2020). This enhanced YOLO V3 model inherited key features from Darknet 53 and exhibited significant performance improvements compared to its predecessors. Through a comprehensive comparative analysis involving YOLO V3, faster R-CNN, SSD, and other YOLO V3 variations, the researchers highlighted YOLO V3's superior accuracy in object detection. They attributed this

success to YOLO V3's utilization of multiscale feature fusion on the image pyramid, which not only enhances accuracy but also optimizes computational efficiency.

Their experiments yielded promising results, with YOLO V3 achieving a remarkable accuracy rate of 92.395% on a sizable dataset comprising 15,000 images of tomatoes classified as healthy or diseased. These findings underscore the significance of advanced deep-learning models in revolutionizing agricultural practices, particularly in the domain of crop disease detection and management. Such advancements hold immense potential for enhancing crop yield, minimizing losses, and promoting sustainable farming practices.

2.2 Established Methods and Approaches

2.2.1 YOLO

YOLO, short for "You Only Look Once," represents a significant milestone in computer vision for real-time object detection. Its architecture, devised in 2016 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, stands out for its simplicity and efficiency (Redmon, et., al 2016). Unlike traditional methods, YOLO processes the entire image in one go, using a single neural network to predict bounding boxes and class probabilities directly. This streamlined approach eliminates the need for multiple stages of processing, resulting in remarkable speed without compromising accuracy.

At the heart of YOLO lies a convolutional neural network (CNN) backbone, typically built upon architectures like Darknet or MobileNet (Redmon, et., al 2016). This backbone extracts hierarchical features from the input image, enabling YOLO to understand both low-level details and high-level semantics crucial for object detection. Through a series of convolutional and pooling layers, the network progressively reduces the image's spatial dimensions while

enhancing its feature representations. YOLO then divides the final feature map into a grid, with each grid cell responsible for predicting bounding boxes and class probabilities for objects within its region. By combining speed, accuracy, and simplicity, YOLO has become a cornerstone in computer vision, powering applications ranging from autonomous vehicles to surveillance systems with its unparalleled performance in real-time object detection.

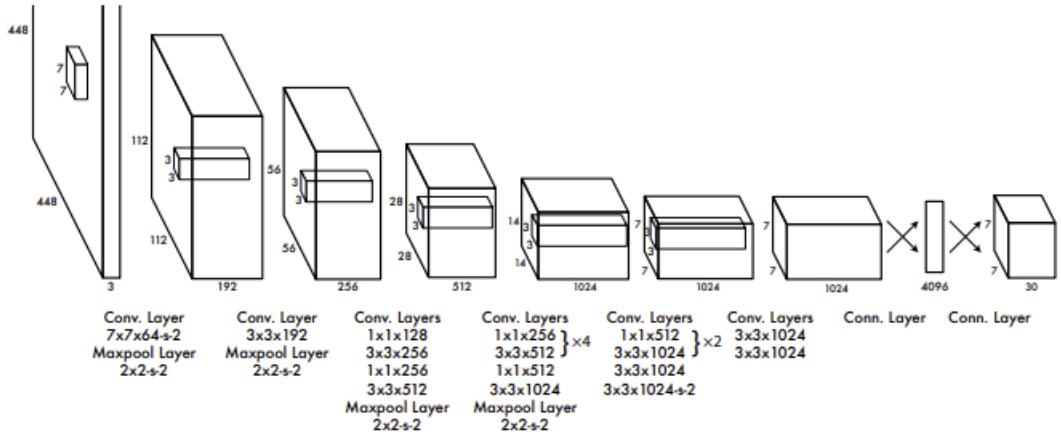

Figure 1 - YOLO Architecture (Redmon, et., al 2016)

2.2.2 YOLOv5 and YOLOv8

YOLOv5 is a previous version of the YOLO object detection model. It was created in 2020 by Ultralytics, the creators of YOLOv3, and employed the PyTorch framework (Terven, et al., 2023). YOLOv5 has gained fame for its speed, user-friendly interface, and its ability to deliver top-notch results in object detection tasks. It outperforms its earlier versions with improved accuracy and simpler training procedures, making it a preferred option among developers across the world.

In contrast, YOLOv8, introduced in 2022 by Ultralytics, stands as the newest addition to the YOLO lineage (Jocher, et al., 2023). Capitalizing on the foundation laid by YOLOv5, YOLOv8 incorporates various enhancements in architecture and developer usability (Terven, et

al., 2023). Offering superior speed and precision compared to YOLOv5, YOLOv8 presents a cohesive platform for training models across object detection, instance segmentation, and image classification tasks.

In the field of object detection, there are many models to choose from, but YOLOv8 and YOLOv5 stand out as top choices developed by Ultralytics. YOLOv8, being the recent version, expands on the successes of its predecessors by adding new features and improvements to boost performance and adaptability. On the other hand, YOLOv5 stands out for its speed, simplicity, and precision, making it a preferred option among professionals.

Table 1 - Performance comparison of YOLOv5 and YOLOv8 (Jocher, et al., 2023)

Model Size	***YOLOv5***	***YOLOv8***	***Difference***
Nano	28	37.3	+33.21%
Small	37.4	44.9	+20.05%
Medium	45.4	50.2	+10.57%
Large	49	52.9	+7.96%
Xtra Large	50.7	53.9	+6.31

2.2.3 Roboflow

Roboflow provides a powerful set of tools to help streamline the creation of computer vision workflows for real-world applications (Lin, et al., 2022). It's not just about organizing data or training models; Roboflow offers a range of features to enhance dataset quality, simplify model training, and speed up deployment. Roboflow can be used for classification, segmentation, or real-time detection.

In our project, we only use Roboflow for the real time detection task. Once we trained our YOLOv8 model and obtained a set of finely-tuned weights, we seamlessly incorporated them into the deployment process using Roboflow's deploy() function. This smooth integration

ensured that our deployment was both fast and successful, ultimately saving us valuable time and effort.

Chapter 3 - Implementation

3.1 Overview of the Data

Our initial dataset comprised 1037 images with a balanced class distribution: 260 soybean images, 256 paddy images, 263 pea images, and 258 wheat images. To address potential limitations arising from a finite dataset size and improve model generalizability, we employed data augmentation techniques. These techniques artificially expanded the dataset by introducing variations of the existing images that mimicked real-world scenarios. Flipping the images horizontally simulated different object orientations, while rotation augmentation introduced in-plane rotations that could occur during image capture. Additionally, hue augmentation was utilized to modify the color spectrum of the images, enhancing the model's robustness to lighting variations. Through the application of these data augmentation techniques, the dataset size was effectively doubled, reaching approximately 2237 images. This augmented dataset was then strategically divided for model training, validation, and testing purposes. A subset of 1811 images was designated as the training set, responsible for teaching the model the underlying patterns within the data. A separate set of 402 images served as the validation set, used to monitor model performance during training and prevent overfitting. Finally, a hold-out set of 24 images was allocated as the test set for the final evaluation of the model's generalizability on unseen data.

3.2 Dataset Preparation

In this section, we discuss in detail how the dataset is collected and how it is further processed before utilizing in our experiment.

3.2.1 Collection and Labelling

The data collection involved the manual capture of RGB images using a Canon EOS R6 Mark II Camera equipped with an RF24-105mm F4-7.1 IS STM lens. This specific camera lens combination provides a versatile focal range and good low-light performance, contributing to the capture of clear and informative images. Each image was captured at a resolution of 6000 x 4000 pixels, offering a substantial level of detail for accurate object identification. Standardized camera settings were employed to maintain consistency across the dataset. These settings included an ISO of 250, an exposure time of 1/250 seconds, an aperture of f/6.7, and a focal length of 31mm. The use of a moderate aperture (f/6.7) balances depth of field with background blur, ensuring that both the objects of interest and their surrounding context are captured in sufficient detail. Furthermore, the chosen focal length (31mm) minimizes perspective distortion, leading to a more natural representation of the objects within the scene.

Figure 2 - Canon EOS R6 Mark II Camera

The distance between the object and camera is maintained at ~ 6cms, and to enhance the model's ability to generalize to real-world scenarios, the images were captured under varying backgrounds and lighting conditions, for that an LED light source(5V/2A) is used which can switch between three modes: White, Warm, and Mix. This diversity within the dataset ensures that the model is not solely reliant on specific visual features associated with controlled environments. By incorporating images with diverse backgrounds and lighting variations, the model is trained to be robust to these factors and perform accurately on unseen data.

Following image capture, the annotation process was completed using LabelImg, a graphical image annotation tool specifically designed for this purpose. LabelImg, written in Python and utilizing Qt for its user interface, offers a user-friendly platform for defining object boundaries and assigning class labels. Using Labellmg, each image was meticulously annotated, with bounding boxes drawn around the objects of interest and corresponding class labels assigned. This annotation process resulted in the creation of a YOLO-formatted text file saved within the same folder as the corresponding image and sharing the same filename. Additionally, a separate file named "classes.txt" is created within each folder. This "classes.txt" file serves as a crucial reference point for the YOLO labels, defining the list of class names associated with the objects identified within the images. The creation and organization of these annotation files are essential for enabling the YOLO object detection model to effectively interpret the labeled data and learn the characteristics of each object class.

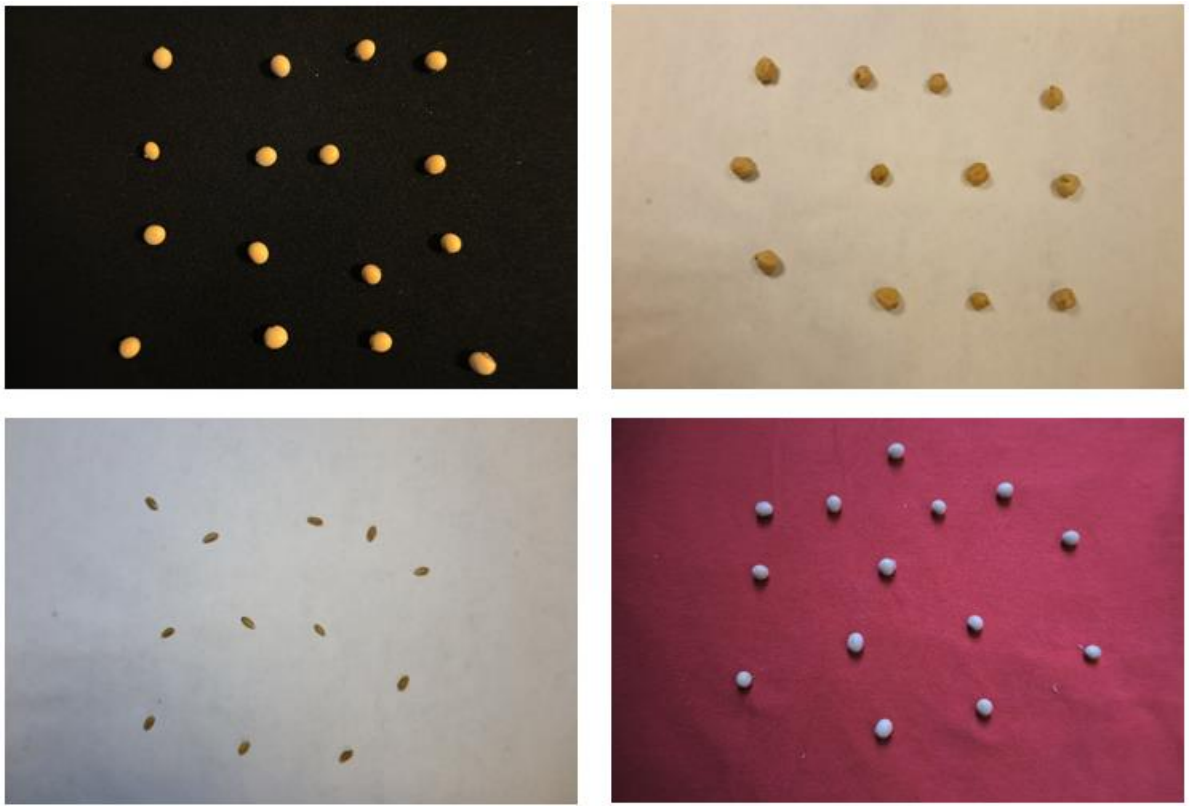

Figure 3 - Sample images from the Dataset.

3.2.2 Data Pre-processing

During the preprocessing stage, we implemented two key steps to prepare the images for model training. First, we employed an auto-orientation function. This technique automatically analyzes the image content and corrects any potential rotational inconsistencies. This ensures consistent object presentation regardless of the camera orientation during image capture. Secondly, all images were resized to a uniform dimension of 640 x 640 pixels. This standardization simplifies the computational demands placed on the model during training and allows for efficient processing of a large number of images. The combined effect of these preprocessing steps is a dataset of consistently oriented and sized images, facilitating more effective model learning and improved performance.

3.2.3 Data Augmentation

To enhance the dataset's robustness and the model's generalizability to real-world variations, we incorporated data augmentation techniques following the preprocessing stage. This strategy aimed to artificially expand the dataset by generating modified versions of the existing images. These modifications mimicked real-world scenarios that the model might encounter during deployment and improved its ability to perform accurately under diverse conditions.

One key augmentation technique employed was horizontal flipping. This process essentially creates mirror images of the originals, doubling the number of training samples while forcing the model to learn object features independent of their orientation within the scene. Additionally, random rotations between -15 and +15 degrees were applied. This augmentation simulates slight camera tilts or rotations that might occur during image capture, enhancing the model's capacity to recognize objects even when presented from slightly non-canonical viewpoints. Furthermore, hue augmentation was utilized to modify the color spectrum of the images within a range of -15 and +15 degrees. This manipulation introduced variations in color and image warmth, ensuring that the model is not solely reliant on specific color features present in the original dataset. Through the combined effect of these data augmentation techniques, the initial dataset size was significantly increased to approximately 2237 images. This augmented dataset provided a richer pool of training data, ultimately leading to a more robust and generalizable model capable of performing effectively under a wider range of real-world conditions.

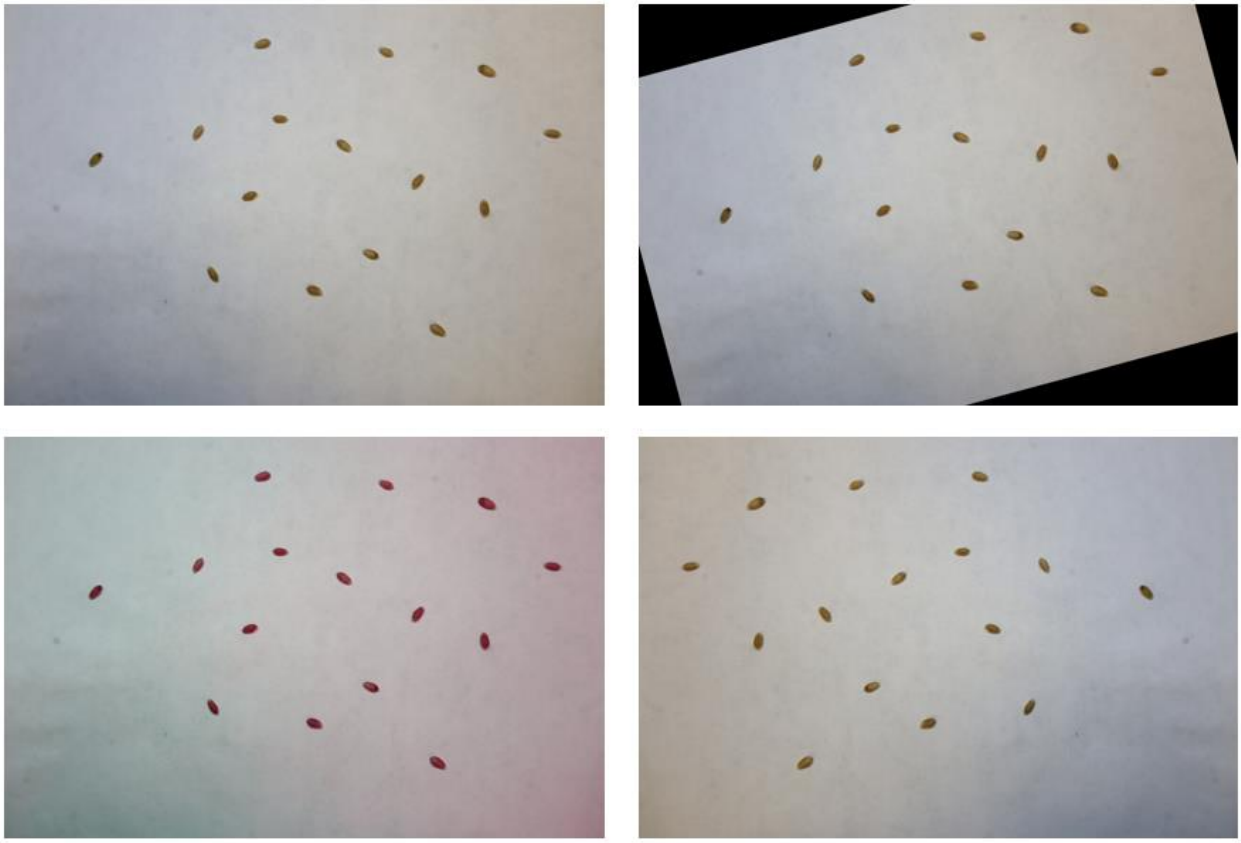

Figure 4 - Images after Data Augmentation (after flipping, rotating & hue)

3.3 Implementation Steps

This work outlines a workflow for real-time object detection using YOLO models as depicted in the accompanying diagram. The process starts with data collection and annotation of objects within the images. Subsequently, the dataset undergoes preprocessing and augmentation to enhance its quality and generalizability. This is followed by a strategic split into training, validation, and testing sets for robust model evaluation. The prepared dataset is then leveraged to train baseline YOLOv5 and then the YOLOv8 models with their performance evaluated and compared. Finally, the trained model is deployed on a Roboflow environment to assess its real-time object detection capabilities, validating the effectiveness of the entire workflow.

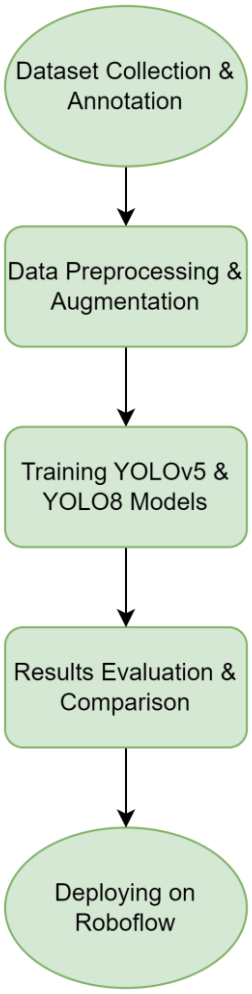

Figure 5 - Project Pipeline

Chapter 4 – Experiments

The following describes the set of experiments that were performed on the data after the data preprocessing and Augmentation. This chapter provides a quick summary of the project, outlining the steps involved in dividing the data into training and testing sets and then outlining how the various experiments were carried out using the data.

4.1 Training, Validation, and Test Data Split

To ensure optimal model training and evaluation, the dataset containing 2237 images was thoughtfully divided into three sets. Most of the dataset, approximately 80%, was allocated to the training set. This allowed the model to learn critical patterns from the data in a comprehensive yet manageable way.

The validation set, comprising roughly 19% of the dataset, was utilized to monitor the model's training progress and prevent overfitting. This occurs when a model becomes too centered on the training data and struggles to perform well on new data. The validation set provided valuable insights into the model's ability to generalize to new scenarios.

Finally, a small set of approximately 1% of the dataset was reserved for the test set. This set served as an impartial assessment of the model's generalizability to unseen data. By evaluating the model's performance on this set, we could gauge the model's expected performance in real-world situations.

The dataset was ultimately split into a training set of 1811 images, a validation set of 402 images, and a test set of 24 images. This approach resulted in a robust and effective framework for model training and evaluation.

4.2 Experimental Design

In this chapter, the steps required to complete each experiment are specified in detail. The first experiment was developed using the YOLOv5 baseline model, capturing the findings; it was then trained on the YOLOv8 model, and the results were compared with the baseline model. Finally, the model is deployed on the Roboflow framework for real-time detection, allowing us to verify our model's performance in real time.

4.2.1 YOLOv5

Our initial foray into training the model involved utilizing the YOLOv5 architecture. We configured the training process with specific hyperparameters, including a learning rate of 0.01, a batch size of 16, and a set of 300 epochs. This training was initially conducted on a laboratory computer with the following specifications: 10th Gen Intel Core i5-13400, 16GB RAM, 8 GB NVIDIA GPU.

However, the training process on the lab computer proved to be time-consuming, taking approximately 14 hours to complete. To expedite the training and explore the potential benefits of a more powerful computing environment, we migrated the training process to Google Colab Pro. This cloud-based platform provided access to an A100 GPU with 40 GB RAM resources. Notably, the training on Colab Pro utilized the same hyperparameters as the initial attempt.

The training process on Colab Pro yielded significant improvements in terms of training time. While the exact number of epochs used is not reported, we observed that the model achieved its best results around the 175th epoch. This indicated a point of diminishing returns, where further training did not lead to a significant increase in model accuracy over the last 100 epochs. We opted to halt training at this point to optimize efficiency.

The final training results are captured and subsequently plotted for visual analysis, which will be discussed in detail in subsequent chapters. Additionally, we evaluated the model's performance on a dedicated testing dataset. This evaluation confirmed that the trained YOLOv5 model achieved satisfactory performance, demonstrating its ability to effectively detect objects within the given data.

4.2.2 YOLOv8

Following the successful training and evaluation of the YOLOv5 model, we proceeded to explore the potential of the YOLOv8 architecture. We configured the training process with similar hyperparameters, including a learning rate of 0.01 and a batch size of 16. This time, however, we leveraged the computational power of Google Colab Pro, equipped with an A100 GPU w 40 GB RAM, to expedite the training process.

The training on Colab Pro proved significantly faster, taking approximately 4 hours to complete. Similar to YOLOv5 training, we observed a point of diminishing returns around the 223rd epoch. There was no substantial improvement in model accuracy after this point, suggesting the model had achieved its optimal performance. We opted to halt training at this epoch to ensure efficiency.

The final training results for YOLOv8 were meticulously captured and visualized through plots, which will be further discussed in subsequent chapters. We then conducted a comparative analysis between the YOLOv5 and YOLOv8 models, using the YOLOv5 results as the baseline. This analysis revealed a slight improvement in performance with YOLOv8. While the difference wasn't substantial, it indicated a potential advantage for YOLOv8 in this specific task.

Finally, to assess the generalizability of the YOLOv8 model, we evaluated its performance on the dedicated testing dataset. This evaluation confirmed that the trained

YOLOv8 model also achieved satisfactory results, demonstrating its effectiveness in real-world object detection scenarios. The positive outcomes from both YOLOv5 and YOLOv8 models suggest the overall effectiveness of the chosen training methodology and dataset preparation.

4.2.3 Deploying in Roboflow

After the completion of training the model on the dataset, as a final step of our project, the model is then deployed on the Roboflow framework to test its real-time performance. We have done this by using Ultralytics and Google Colab, it has a special feature to directly deploy our trained model on Roboflow. Below are some images that were generated using Roboflow's real-time detection engine.

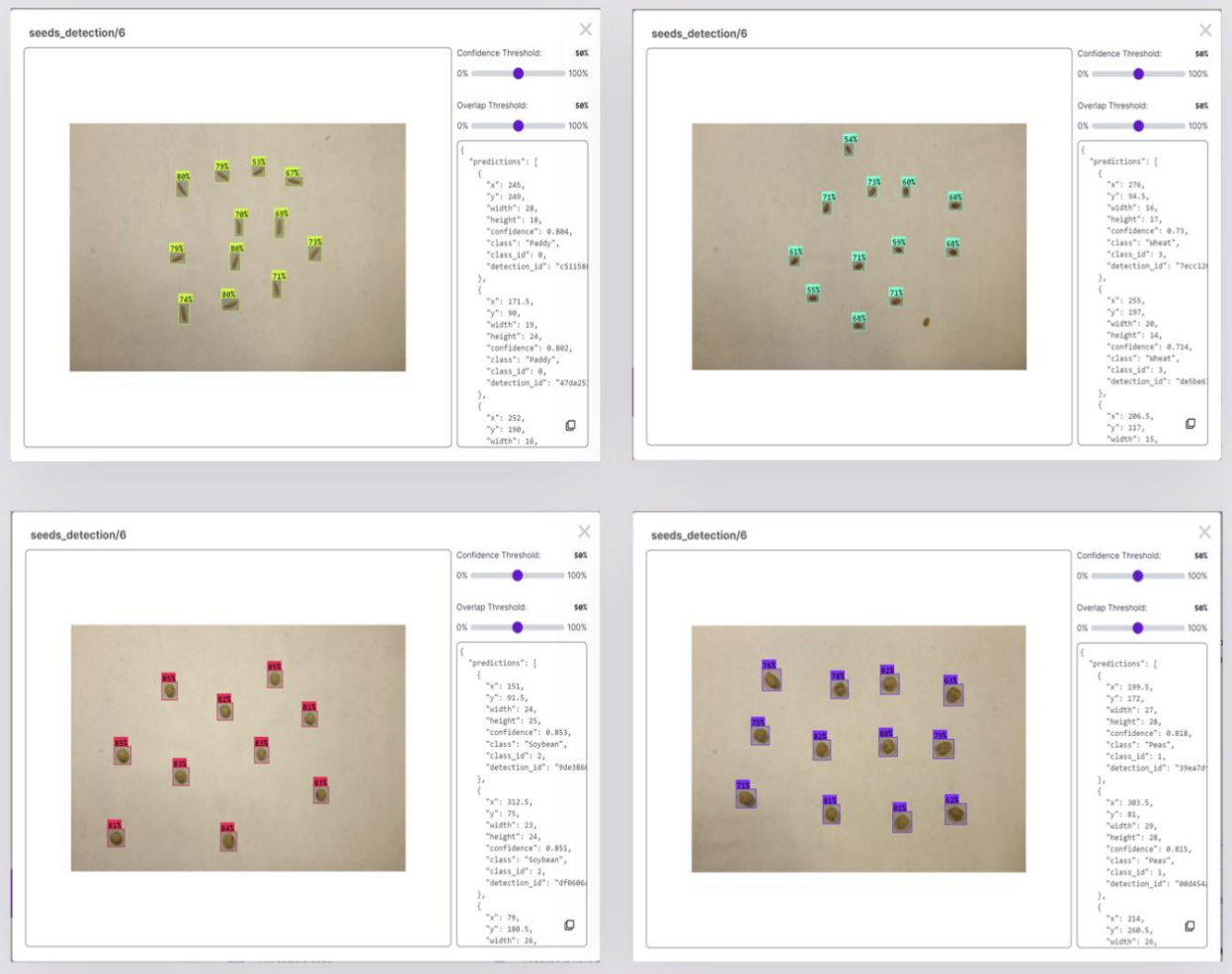

Figure 6 - Real-time detection samples from Roboflow

Chapter 5 – Experimental Results

In this chapter, we talk more about the results obtained after validating and testing the model. We examine the mean Average Precision metric (mAP), Precision, and Recall in addition to using a variety of graphical representations, such as confusion matrices F1 Confidence curves and validation batch data, to discuss this.

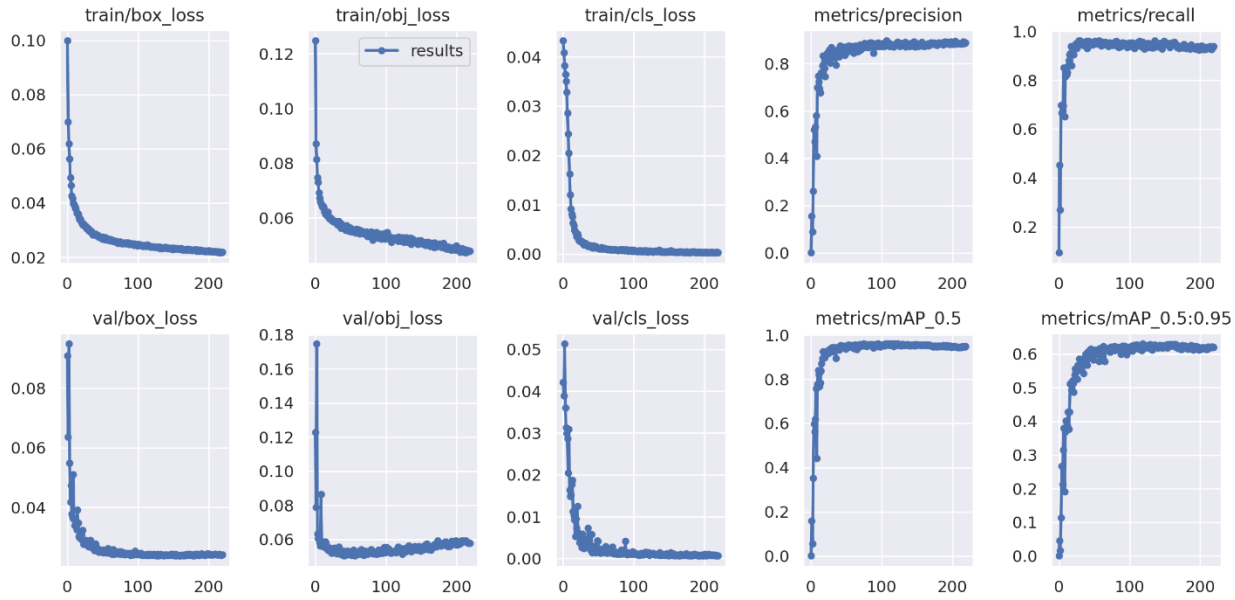

Figure 7 - YOLOv5 Final results

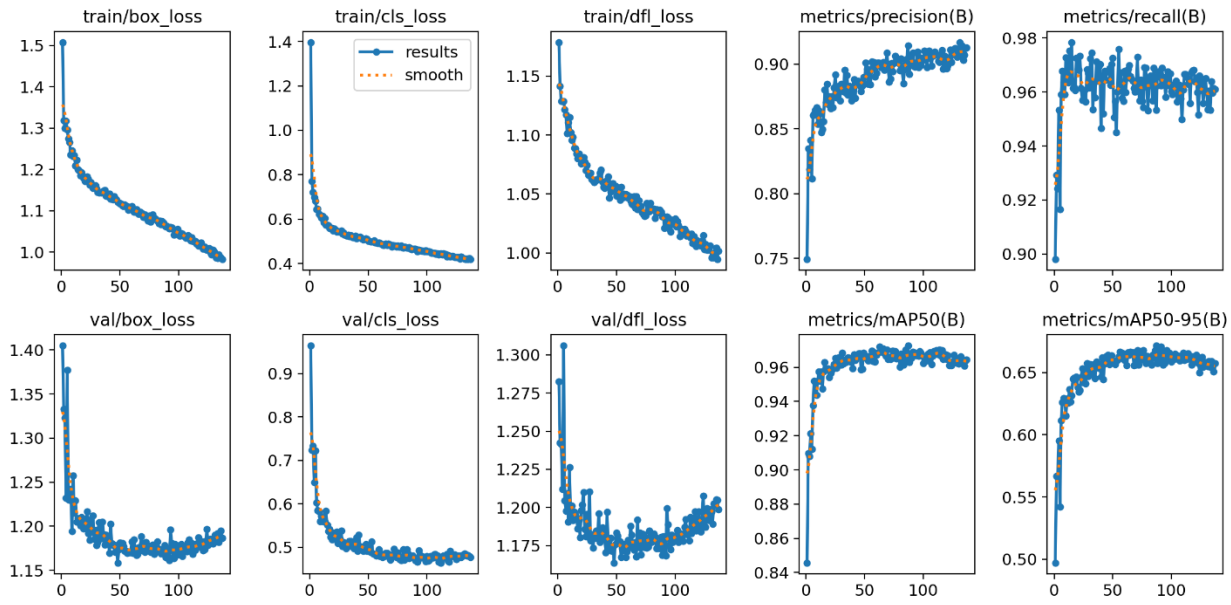

Figure 8 - YOLOv8 Final Results

5.1 Evaluation Metrics

Evaluation metrics are essential for assessing the performance of object detection models such as YOLO (You Only Look Once). They provide quantitative measures to gauge accuracy, robustness, and generalization capabilities, enabling researchers and practitioners to compare models effectively and fine-tune their approaches. In YOLO-based object detection tasks, several key evaluation metrics are commonly utilized to evaluate the model's performance in detecting and localizing objects within images. These metrics help in understanding the model's strengths and areas for improvement. Here are some of the primary evaluation metrics used in this project:

Table 2 - Comparison of YOLOv5 and YOLOv8 results

	mAP		Precision		Recall	
Class	YOLOv5	YOLOv8	YOLOv5	YOLOv8	YOLOv5	YOLOv8
All	96	97.3	87.9	91.5	95.2	95.3
Paddy	91.7	97.7	84.7	93.3	92	95.3
Peas	95	97.2	86.4	93.4	94.9	92
Soybean	97.8	98.3	90.6	94.1	99	98.9
Wheat	94.2	95.9	82.2	85	96.8	95

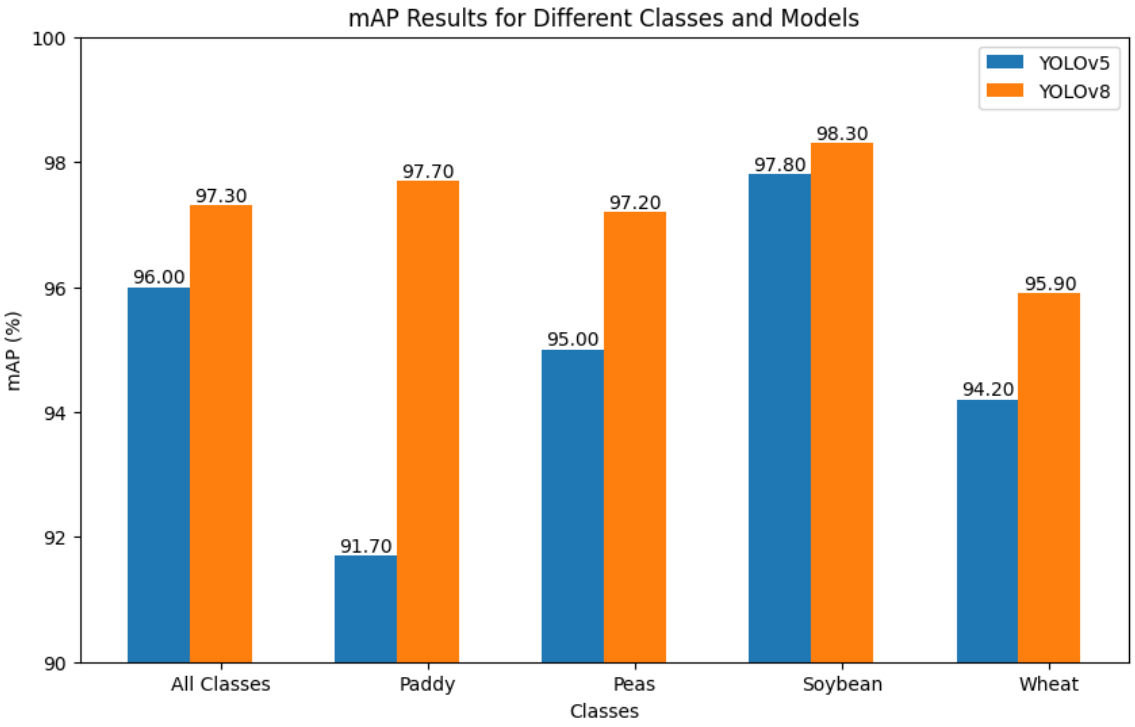

Figure 9 - mAP comparison of YOLOv5 and YOLOv8

5.2 Mean Average Precision

Mean Average Precision (mAP) is a widely adopted metric for assessing object detection models, including YOLO. It calculates the average precision across different object categories and then averages these values to obtain an overall performance measure. In YOLO, mAP is particularly valuable as it provides a single scalar value representing the model's overall

detection performance, making it easier to compare different models and configurations. The mean average precision (mAP) of the YOLOv5 object detection model is 96% when tested on all classes. On the other hand, the YOLOv8 model achieves a mAP of 97.3%. These results show that YOLOv8 performs slightly better than YOLOv5 in detecting objects across all classes.

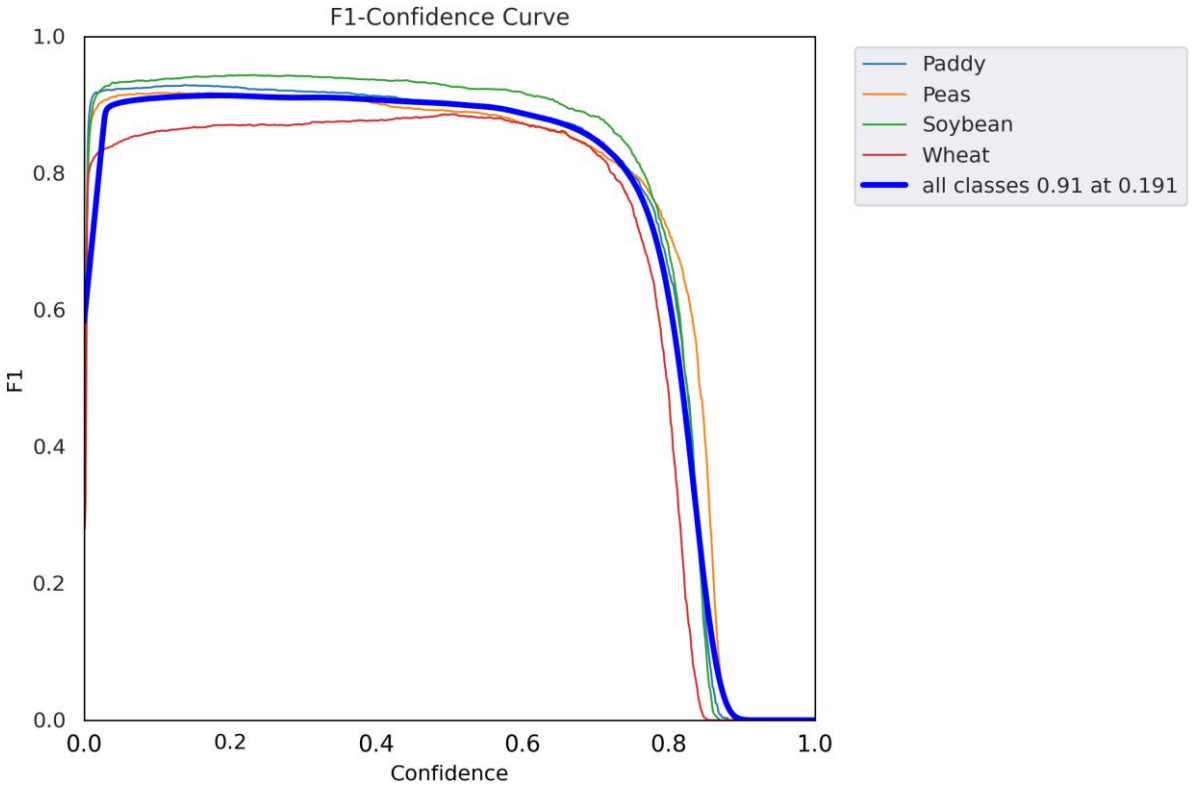

Figure 10 - YOLOv5 F1 Confidence Curve

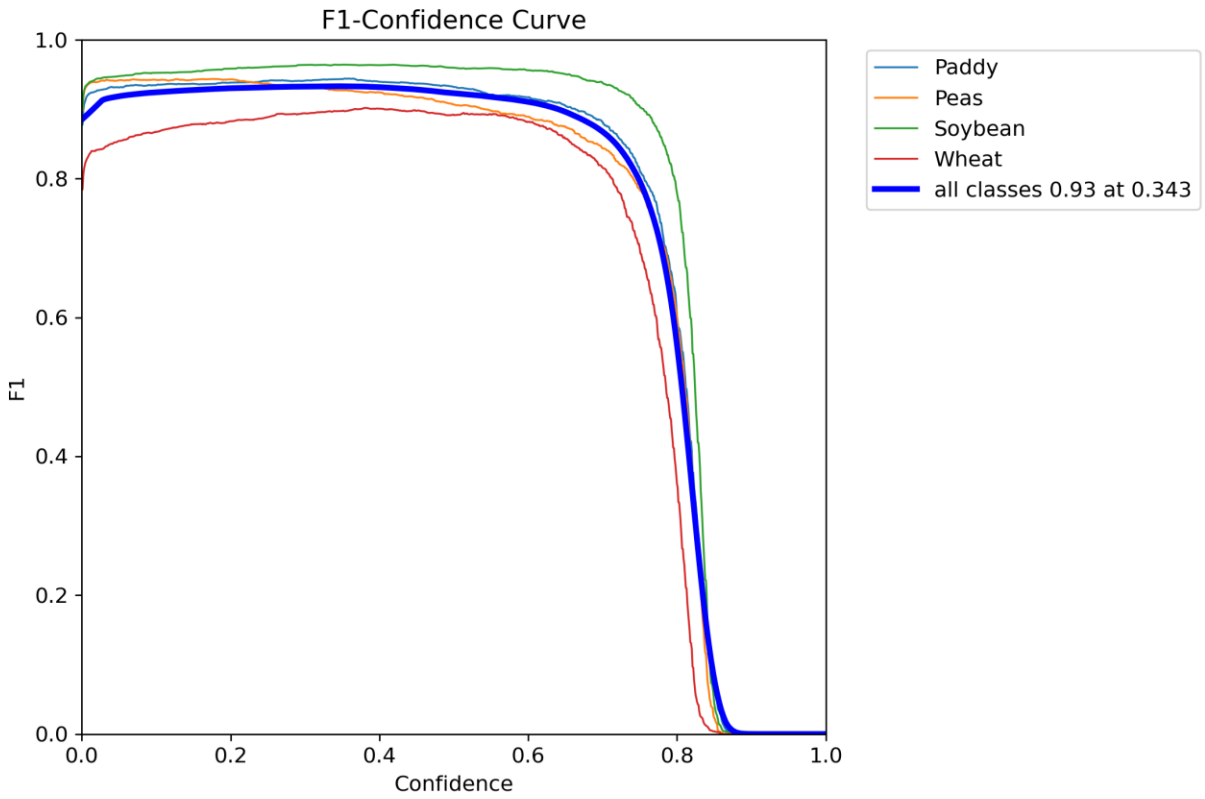

Figure 11 - YOLOv8 F1 Confidence Curve

5.3 Precision and Recall

Precision measures the accuracy of the model in identifying objects. It quantifies the proportion of correctly predicted positive detections (true positives) among all predicted positive detections (true positives + false positives).

Recall measures the model's capability to capture all instances of the target objects in the image. It quantifies the proportion of correctly predicted positive detections (true positives) among all ground truth positive instances (true positives + false negatives).

The evaluation of the object detection models YOLOv5 and YOLOv8 shows that YOLOv8 performs better than YOLOv5. The Precision and Recall metrics for all classes combined are 87.9% and 95.2%, respectively, for YOLOv5, while for YOLOv8, the metrics are

91.4% and 95.3%, respectively. This implies that YOLOv8 is more accurate and reliable in detecting and recognizing objects than YOLOv5. The figures below show the Precision-Recall Curve for YOLOv5 and YOLOv8.

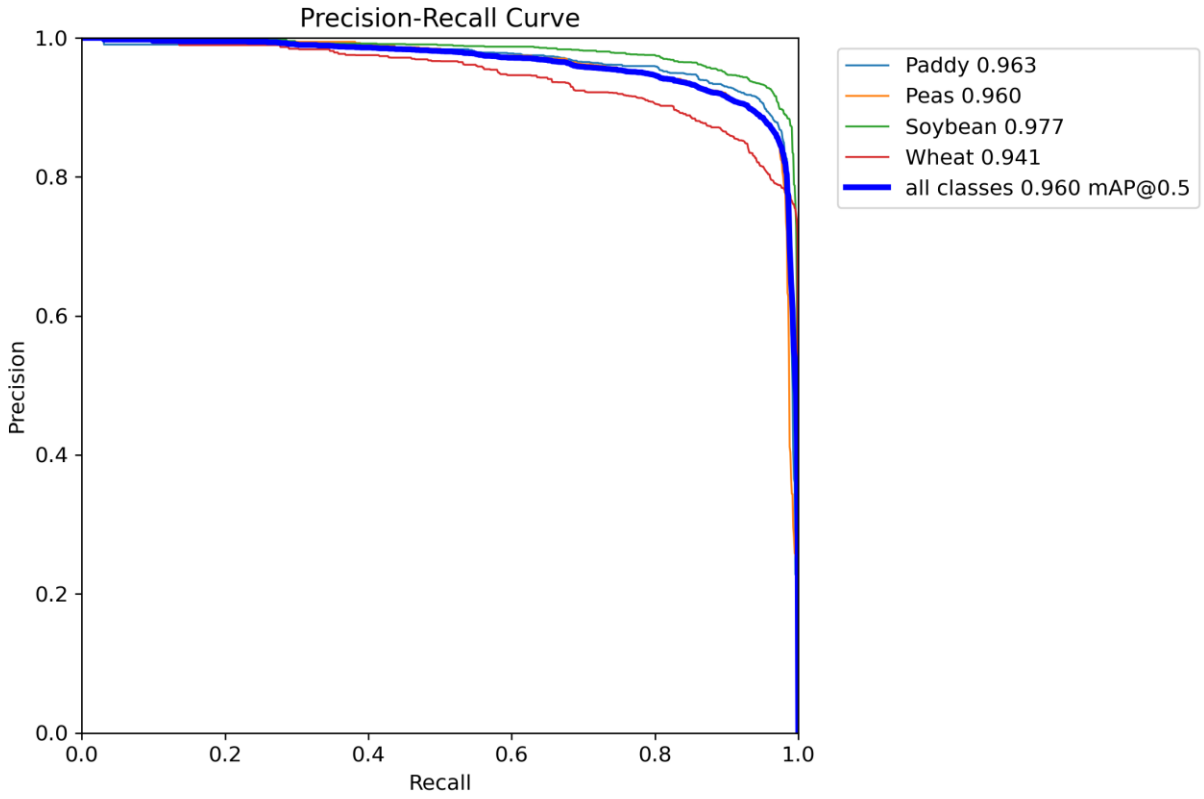

Figure 12 - YOLOv5 Precision and Recall Curve

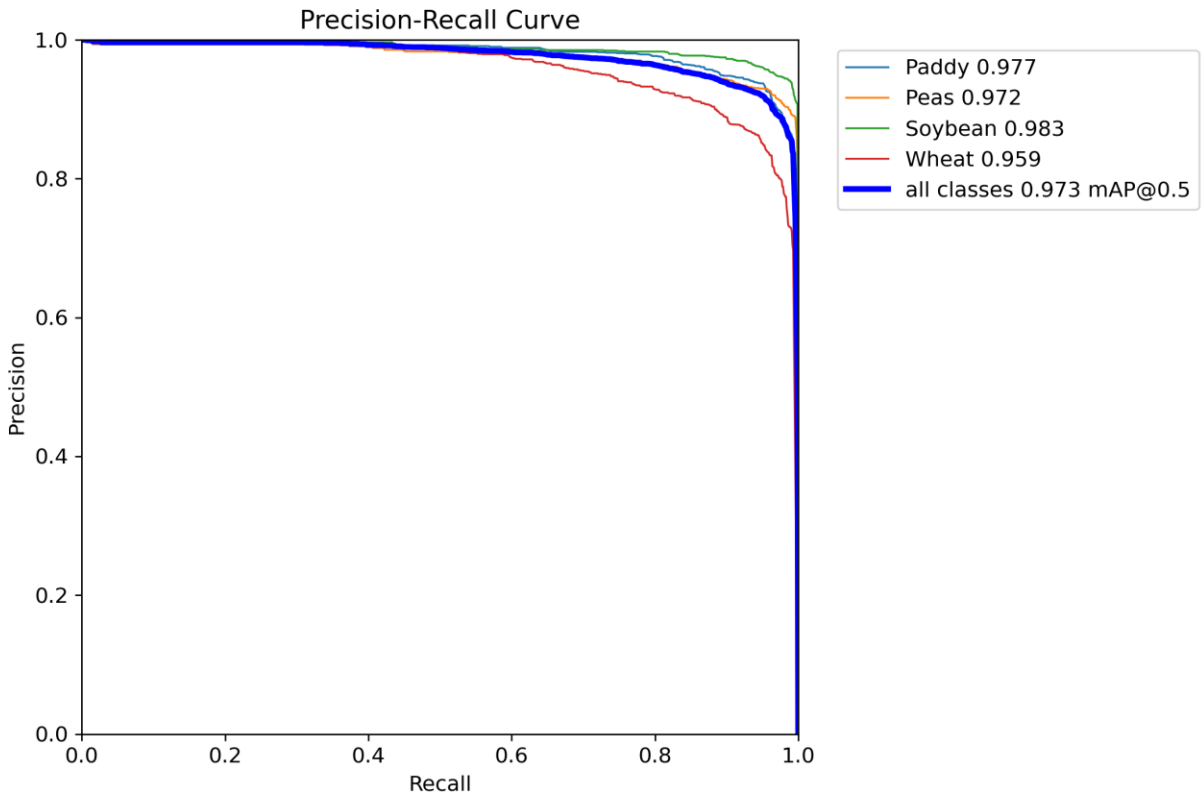

Figure 13 - YOLOv8 Precision and Recall Curve

5.4 Confusion Matrix

We employed confusion matrices to gain insights into the models' performance for classifying distinct seed categories. These matrices revealed minimal off-diagonal elements, signifying a low occurrence of misclassification events. This observation suggests that the models developed robust classification capabilities, effectively distinguishing between various seed types within the dataset. The following figures show the confusion matrices for YOLOv5 and YOLOv8.

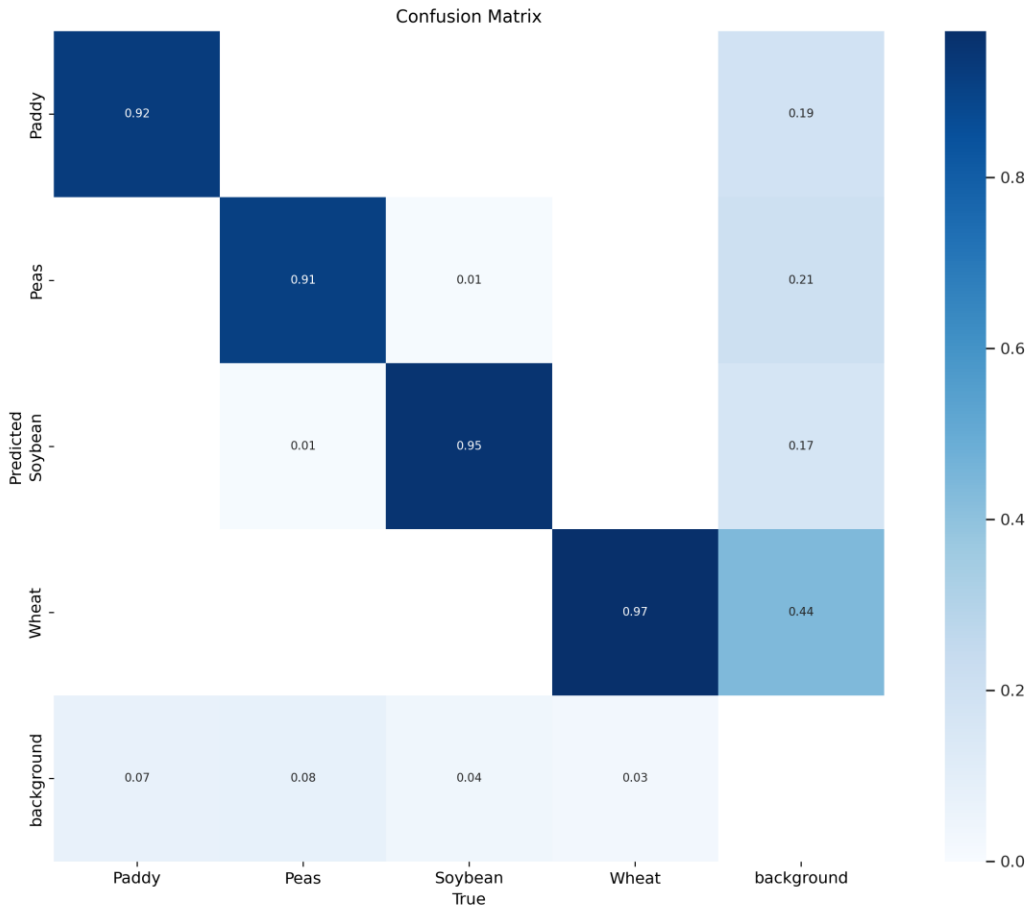

Figure 14 - YOLOv5 Confusion Matrix

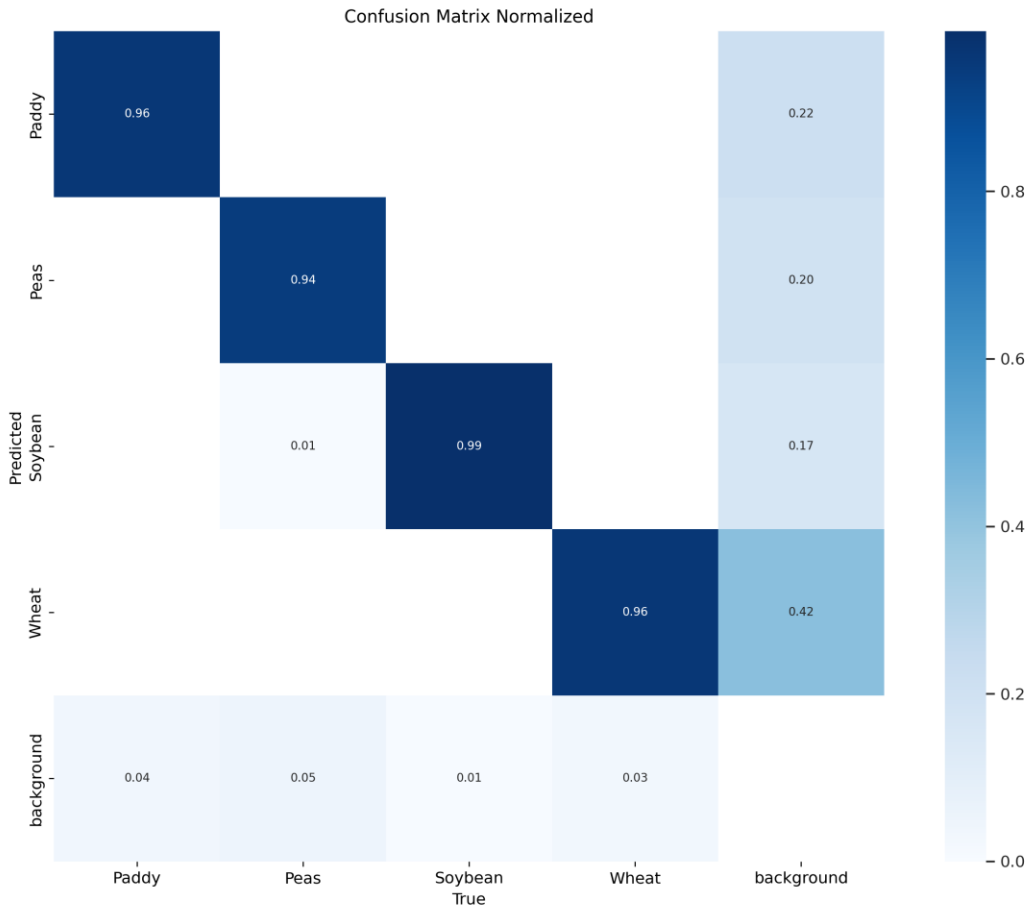

Figure 15 - YOLOv8 Confusion Matrix

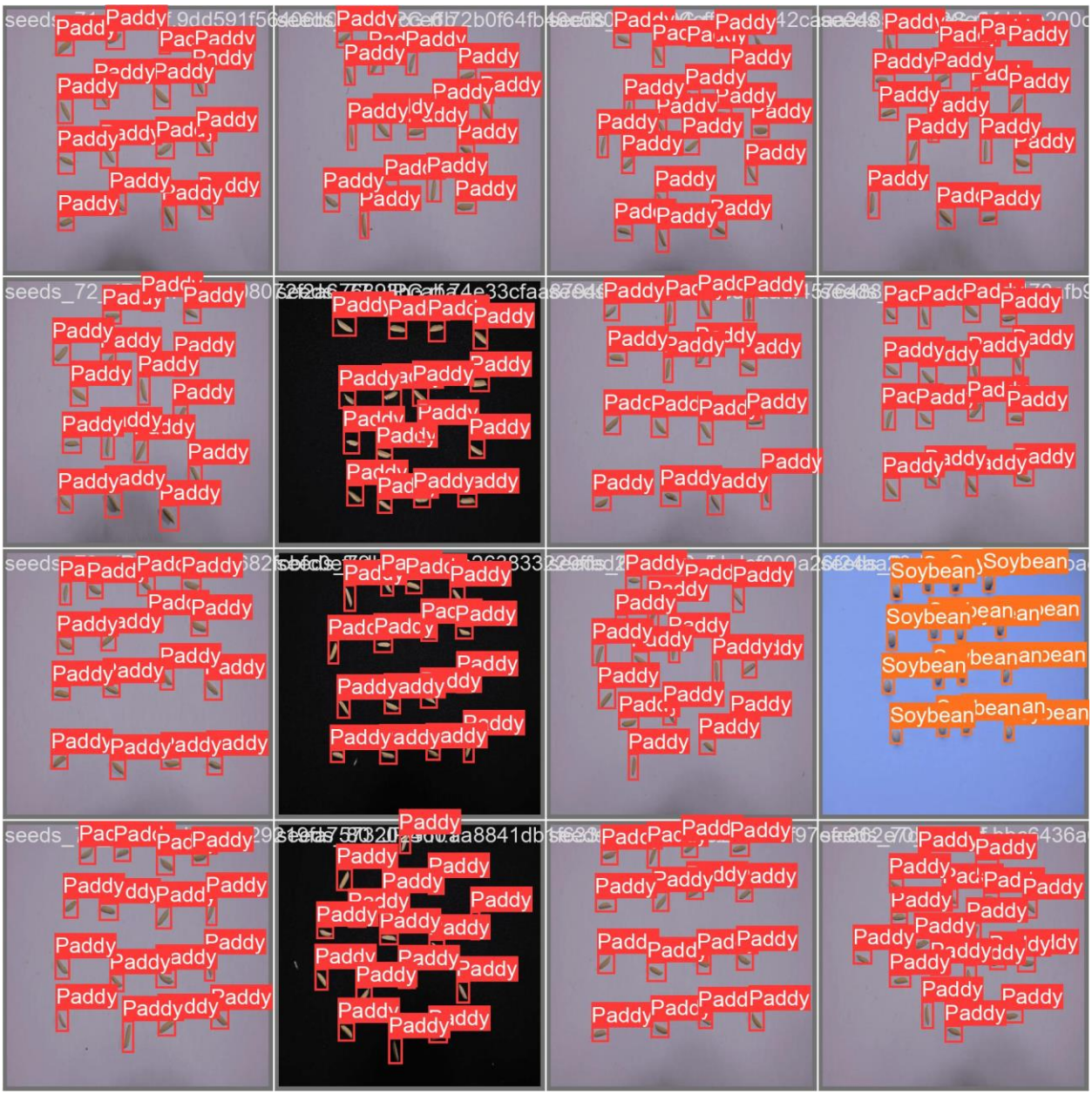

Figure 16 - Validation batch results

Chapter 6 – Summary and Future Work

6.1 Summary of Results

The project successfully automated plant seed detection and classification tasks using the YOLO v8 object detection model. The model was trained to identify and classify various plant seeds in real time and demonstrated high precision and recall rates. Comparative analyses with other deep learning models showed that the YOLO v8 model is superior in terms of both accuracy and computational efficiency. Additionally, the trained model was deployed on the Roboflow platform, enabling seamless real-time seed detection and classification in practical settings. These results highlight the potential of deep learning and computer vision techniques in revolutionizing agricultural practices, offering scalable and efficient solutions for crop management and automation.

The project successfully automated plant seed detection and classification tasks using the YOLO v8 object detection model. The model was trained to identify and classify various plant seeds in real time and demonstrated high precision and recall rates. Comparative analyses with other deep learning models showed that the YOLO v8 model is superior in terms of both accuracy and computational efficiency. Additionally, the trained model was deployed on the Roboflow platform, enabling seamless integration into agricultural workflows, and facilitating real-time seed detection and classification in practical settings. These results highlight the potential of deep learning and computer vision techniques in revolutionizing agricultural practices, offering scalable and efficient solutions for crop management and automation.

6.2 Future Scope

While the current project has successfully demonstrated the efficacy of the YOLO v8 object detection model in automating plant seed detection and classification tasks, there exist several avenues for future exploration and enhancement:

- We can develop and integrate a **counting algorithm**, such that it returns the count of seeds for each class.
- Continuously **expanding and diversifying the dataset** by including more varieties of plant seeds and capturing images under various environmental conditions would enhance the model's robustness and generalization capabilities.
- Exploring **real-time applications** of the model in agricultural settings by deploying it on edge devices or drones could facilitate on-the-fly seed detection and classification, enabling timely interventions and decision-making.
- Investigating **multi-modal approaches** by combining visual data from images with other types of data such as spectral or thermal images could provide additional insights and improve overall performance, especially in challenging conditions like low light or adverse weather.
- **Collaboration** with agricultural experts, botanists, and agronomists could provide domain-specific insights and help tailor the model to address specific agricultural challenges and requirements.
- Enhancing the **scalability** and ease of deployment of the model by developing user-friendly interfaces and integrating them into existing agricultural machinery and systems would facilitate widespread adoption and practical implementation.

By exploring these future avenues, the project can continue to advance the field of agricultural automation, offering scalable and efficient solutions for seed detection, classification, and mixed crop management, ultimately contributing to improved agricultural productivity and sustainability.

References

Jocher, G., Chaurasia, A., Stoken, A., Borovec, J., NanoCode012, Kwon, Y., Michael, K., TaoXie, Fang, J. Lorna, Yifu, Z., Wong, C., Abhiram, V., Montes, D., Wang, Z., Fati, C., Nadar, J., Laughing, … Jain, M. (2022). ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation (v7.0). Zenodo. https://doi.org/10.5281/zenodo.7347926

Jocher, G., Chaurasia, A., and Qiu, J. (2023). Ultralytics YOLOv8. Version 8.0.0. Available online: https://github.com/ultralytics/ultralytics (retrieved on Dec. 1, 2023).

Jun Liu and Xuewei Wang. (2020). Tomato Diseases and Pests Detection Based on Improved Yolo V3 Convolutional Neural Network. Front. Plant Sci. 11, June (2020), pp. 1–12. https://doi.org/10.3389/fpls.2020.00898

Kundu, Nidhi, Rani, Geeta, and Dhaka, Vijaypal. (2021). Seeds Classification and Quality Testing Using Deep Learning and YOLO v5. pp. 153-160. https://doi.org/10.1145/3484824.3484913

Lin, Q., Ye, G., Wang, J. & Liu, H.. (2022). RoboFlow: a Data-centric Workflow Management System for Developing AI-enhanced Robots. Proceedings of the 5th Conference on Robot Learning, in Proceedings of Machine Learning Research 164:1789-1794 Available from https://proceedings.mlr.press/v164/lin22c.html

Margapuri, V., and Neilsen, M. (2021). Classification of Seeds using Domain Randomization on Self-Supervised Learning Frameworks. In Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI), Orlando, FL, USA. pp. 01-08. https://doi.org/10.1109/SSCI50451.2021.9659998

Redmon, J., and Farhadi, A. (2017). "YOLO9000: Better, Faster, Stronger," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017 pp. 6517-6525. https://doi.org/10.1109/CVPR.2017.690

Selcuk, B., Serif, T. (2023). A Comparison of YOLOv5 and YOLOv8 in the Context of Mobile UI Detection. In: Younas, M., Awan, I., Grønli, TM. (eds) Mobile Web and Intelligent Information Systems. MobiWIS 2023. Lecture Notes in Computer Science, vol 13977. Springer, Cham. https://doi.org/10.1007/978-3-031-39764-6_11

Terven J, Córdova-Esparza D-M, Romero-González J-A. (2023). A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and

YOLO-NAS. Machine Learning and Knowledge Extraction; 5(4):1680-1716. https://doi.org/10.3390/make5040083

Wang, Q., and Qi, F. (2019). Tomato Diseases Recognition Based on Faster RCNN. In Proceedings of the 10th International Conference on Information Technology in Medicine and Education (ITME), Qingdao, China. pp. 772-776. https://doi.org/10.1109/ITME.2019.00176

Zaidi SSA, Ansari MS, Aslam A, Kanwal N, Asghar M, Lee B. (2022). A survey of modern deep learning-based object detection models, Digital Signal Processing, Volume 126, 103514, ISSN 1051-2004. https://doi.org/10.1016/j.dsp.2022.103514