On convergence of neural networks for applications in agriculture

by

Rahul Harsha Cheppally

B.Tech, Jawaharlal Nehru Technological University, India, 2015

M.S., Cleveland State University, 2017

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Carl and Melinda Helwig Biology and Agricultural Engineering
Carl Rice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2024

Abstract

Neural networks, as universal function approximators, have gained increasing popularity due to the availability of abundant computational resources and data from IoT devices. This thesis investigates the applications of neural networks in agriculture, focusing specifically on seed detection, distance estimation, row detection, and multi-robot pesticide spraying tasks. The first study aims to develop an automated system for obtaining seed placement information and reducing the time required compared to manual methods like using a pogo-stick or a ruler. A 12-row planter was instrumented with cameras and a GPS, and various object detection models (YOLOR-P6, YOLOR-CSPX, YOLOX-S, YOLOX-M, YOLOX-L, YOLOX-TINY, and YOLOV4) were trained on a dedicated dataset. An algorithm was designed to estimate the distance between consecutive seeds by filtering out old detections using the Intersection over Union (IoU) metric and GPS stream. The system's performance was evaluated using metrics such as Jensen–Shannon Divergenceplanting (JSD), mean, standard deviation, δCount, and RMS. Results indicate that the developed system and algorithm performed better at a planting speed of 9.66 kmph compared to 12.87 kmph, reducing the time taken to detect seeds and estimate seed spacing from 2 hours (manual method) to 1 minute 14 seconds using YOLOR-CSPX. The second study addresses the challenges of multi-robotic pesticide delivery tasks, where traditional task planning and allocation strategies require prior knowledge of crop infection levels, typically obtained through preliminary field scouting. As this assumption becomes obsolete with autonomous robots equipped with integrated sensing and spraying capabilities, the study reformulates the problem within the context of a Partially Observable Markov Decision Process (POMDP). A novel architecture, similar to Cross Transformer, is designed to address these challenges, and comparative evaluations against expert models trained on out-of-distribution datasets are conducted. The study demonstrated that the proposed model performs almost as well as the expert model,

with a 0.5% difference in performance, and is more robust to out-of-distribution data. The evalutation metrics are the total distance travelled by the robots as well as the total reward obtained by the robots. The third study focuses on under-canopy robot navigation, which has seen significant demand for crop phenotyping, crop water stress assessment, and liquid application. Traditional GPS fails under canopies due to multi path issues, posing challenges. The study introduces a novel approach using smooth polynomials for row and canopy identification, leveraging image space prediction and a customized loss function to enhance generalization. This method delineates navigation boundaries with polynomial functions in image space and extrapolates seamlessly into real-world navigation due to these polynomials being smooth, down streams task such as control and state estimate become easier. The approach achieves 1ms latency on edge devices and is evaluated using a metric similar Intersection over Union named MPD(Mean Polynomial Distance), ensuring high accuracy for under-canopy navigation tasks.

On convergence of neural networks for applications in agriculture

by

Rahul Harsha Cheppally

B.Tech, Jawaharlal Nehru Technological University, India, 2015

M.S., Cleveland State University, 2017

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Carl and Melinda Helwig Biology and Agricultural Engineering
Carl Rice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2024

Approved by:

Major Professor
Ajay Sharda

Copyright

© Rahul Harsha Cheppally 2024.

Abstract

Neural networks, as universal function approximators, have gained increasing popularity due to the availability of abundant computational resources and data from IoT devices. This thesis investigates the applications of neural networks in agriculture, focusing specifically on seed detection, distance estimation, row detection, and multi-robot pesticide spraying tasks. The first study aims to develop an automated system for obtaining seed placement information and reducing the time required compared to manual methods like using a pogo-stick or a ruler. A 12-row planter was instrumented with cameras and a GPS, and various object detection models (YOLOR-P6, YOLOR-CSPX, YOLOX-S, YOLOX-M, YOLOX-L, YOLOX-TINY, and YOLOV4) were trained on a dedicated dataset. An algorithm was designed to estimate the distance between consecutive seeds by filtering out old detections using the Intersection over Union (IoU) metric and GPS stream. The system's performance was evaluated using metrics such as Jensen–Shannon Divergenceplanting (JSD), mean, standard deviation, δCount, and RMS. Results indicate that the developed system and algorithm performed better at a planting speed of 9.66 kmph compared to 12.87 kmph, reducing the time taken to detect seeds and estimate seed spacing from 2 hours (manual method) to 1 minute 14 seconds using YOLOR-CSPX. The second study addresses the challenges of multi-robotic pesticide delivery tasks, where traditional task planning and allocation strategies require prior knowledge of crop infection levels, typically obtained through preliminary field scouting. As this assumption becomes obsolete with autonomous robots equipped with integrated sensing and spraying capabilities, the study reformulates the problem within the context of a Partially Observable Markov Decision Process (POMDP). A novel architecture, similar to Cross Transformer, is designed to address these challenges, and comparative evaluations against expert models trained on out-of-distribution datasets are conducted. The study demonstrated that the proposed model performs almost as well as the expert model,

with a 0.5% difference in performance, and is more robust to out-of-distribution data. The evalutation metrics are the total distance travelled by the robots as well as the total reward obtained by the robots. The third study focuses on under-canopy robot navigation, which has seen significant demand for crop phenotyping, crop water stress assessment, and liquid application. Traditional GPS fails under canopies due to multi path issues, posing challenges. The study introduces a novel approach using smooth polynomials for row and canopy identification, leveraging image space prediction and a customized loss function to enhance generalization. This method delineates navigation boundaries with polynomial functions in image space and extrapolates seamlessly into real-world navigation due to these polynomials being smooth, down streams task such as control and state estimate become easier. The approach achieves 1ms latency on edge devices and is evaluated using a metric similar Intersection over Union named MPD(Mean Polynomial Distance), ensuring high accuracy for under-canopy navigation tasks.

Contents

List of Figures ix

List of Tables x

Acknowledgements xi

Dedication xii

Preface 1

1 Introduction 1

1.1 Overview 1

1.2 History of Agricultural Revolutions 1

1.2.1 Hunter-Gatherer Era 1

1.2.2 Agricultural Revolution 2

1.2.3 Industrial Revolution 2

1.2.4 The Impact of Computing on Agriculture 2

1.2.5 Intelligence in Agriculture 3

1.3 Thesis Objectives and Structure 3

1.3.1 Objectives 3

1.3.2 Structure 4

2 Literature Review 5

2.1 Neural Networks 5

2.2 Applications in Agriculture 7

2.3 Neural Networks Based Computer Vision Applications in Agriculture 8
2.4 Reinforcement Learning in Agriculture 10

3 Seed Localization System Suite with CNNs for Seed Spacing Estimation, Population Estimation and Doubles Identification 11
3.1 Abstract 11
3.2 Introduction 13
3.3 Setup 15
3.3.1 Planting System Setup 15
3.3.2 Data Acquisition System 16
3.4 Methodology 18
3.4.1 Seed Detection 18
3.4.2 Distance Estimation 26
3.5 Evaluation Metrics 27
3.5.1 Jensen–Shannon divergence [JSD] 27
3.5.2 Mean and Standard deviation 28
3.5.3 Δ Count 28
3.5.4 RMS 28
3.6 Algorithm 28
3.7 Results and Analysis 34
3.8 Conclusions 35
3.9 Future Work 36

4 Graph based Reinforcement Learning Approach For Multi-Robot Task Allocation in Pesticide Spraying 38
4.1 Abstract 38
4.2 Introduction 39
4.2.1 Contributions of this Study 41

4.2.2 Limitations . . . 42
4.2.3 Societal Impacts . . . 42
4.3 Background . . . 42
4.3.1 Deep Reinforcement Learning . . . 42
4.3.2 Graph Neural Networks (GNNs) . . . 45
4.4 Problem Formulation . . . 46
4.5 Methodology . . . 47
4.5.1 State Representation . . . 47
4.5.2 Action Representation and Reward . . . 48
4.5.3 Model Architecture . . . 49
4.6 Experimental Setup . . . 50
4.7 Results and Discussion . . . 52
4.8 Supplementary Material . . . 54
4.8.1 Hyper Parameters . . . 54

5 NeuRowNet : End to End Row Detection for Robot Under The Canopy Row Navigation . . . 56
5.1 abstract . . . 56
5.2 Introduction . . . 57
5.3 Methodology . . . 59
5.3.1 Dataset and Augmentations . . . 61
5.3.2 Architecture . . . 61
5.3.3 Learning . . . 65
5.4 Experiments . . . 65
5.4.1 Objective . . . 65
5.4.2 Experimental Setup . . . 66
5.4.3 Evaluation Metrics . . . 66
5.5 Discussion . . . 67

5.5.1 Conclusion 68
5.5.2 Limitations 69

6 Conclusion & Future Work 70
6.1 Future Work 73

List of Figures

2.1 A simple Neural Network architecture with an input layer, two hidden layers, and an output layer. Each layer contains multiple neurons connected by weighted edges. 6

3.1 Pogo stick device manufactured by sun co-operative 15

3.2 Camera and LED within SLS mount (white 3-D printed part) installed on the row unit 16

3.3 Tractor and planter with SLS for field data collection 17

3.4 Some images in the Dataset 19

3.5 Radom Letter Box Transformed Image 20

3.6 Images After Augmentation 23

3.7 Field of view calculation test 24

3.8 Precision Recall Curve On Test Dataset 26

3.9 Algorithm Overview 31

3.10 Box plots for JSD, ΔMean and ΔCount for the distance estimator using the 7 models 35

4.1 In Figure 4.1a, an example farm requiring pesticide application is depicted. This farm is segmented into various zones, as detailed in Section 4.4, highlighted with a yellow overlay. The centroids of these zones are marked with red dots, while the blue dots indicate the locations of tanks in the headland. As illustrated in Figure 4.1c, the graph representing these details is subsequently compressed to manage the increasing number of nodes, thereby enhancing the scalability of the problem. The labels M0 and M1 in the compressed graph are muxes to join the edges used to aid in visualization. 40

4.2 The proposed architecture, comprising a graph neural network, a causal Transformer and a cross attention Transformer operation A is selecting the embedding of the nodes available in action queue 51

4.3 Overall results of the experimental run using Task Transformer and Cross Transformer vs Expert 53

5.1 Samples from the train dataset: labeled samples (Blue), positive predictions with confidence greater than 0.5 (Green), and non-existent row predictions with confidence less than 0.5 (White). 58

5.2 The under-the-canopy robot equipped with Jetson Orin and Luxonis OAK-D Pro for image acquisition. 60

5.3 Model Architecture 62

5.4 Samples from test set for the three backbones from Batch 1 (MSE Loss) . . . 63

5.5 Samples from test set for the three backbones from Batch 2 (PolyLoss) . . . 64

5.6 Samples from test set for the three backbones from Batch 3 (PolyLoss, CircleLoss) 66

List of Tables

3.1 Comparison of various Object detection algorithms 26

3.2 Operating speeds, seeding populations, mean observed between seeds and standard deviation of the seed distribution 27

3.3 Evaluation Metrics (section 3.5) for the seed spacing estimation algorithm for the different detection models 33

4.1 Expert Scenarios 52

4.2 Percentage of Expert Trajectories 53

5.1 Experimentation Batches 66

5.2 Performance Metrics 67

Acknowledgments

I would like to express my deepest gratitude to my parents, Amma and Nanna, for their unwavering belief in me. I am also sincerely thankful to Dr. Ajay Sharda and my supervisory committee for their invaluable guidance and support throughout this journey. I am especially grateful to my girlfriend and friends for their constant encouragement and understanding. Lastly, I extend my appreciation to my lab mates for their assistance; without their help, I might still be labeling seeds.

Dedication

Dedicated to my Mom and Dad.

Chapter 1

Introduction

1.1 Overview

Agriculture, the cultivation of food and goods through farming, produces the vast majority of the world's food supply. Technological advancements in agriculture are crucial to meet the growing global food demand and to ensure sustainable practices. This chapter provides a comprehensive historical background, highlighting the evolution of agriculture through various revolutions and the significant impact of technological advancements, to lay the foreground for the current on going agricultural revolution regarding neural networks and their applications in modern agriculture.

1.2 History of Agricultural Revolutions

1.2.1 Hunter-Gatherer Era

For nearly 300,000 years, human ancestors led a nomadic lifestyle, relying on hunting, gathering, and scavenging for sustenance. These early societies were small, mobile, and depended heavily on the natural availability of resources. Wild plant-based foods and fungi were important staples in the Paleolithic diet, and the hunt for wild animals was often depicted as epic conflicts against prehistoric megafauna[?] . This was during the Pleistocene Epoch.

1.2.2 Agricultural Revolution

Towards the end of Pleistocene Epoch. The Climate started to become warmer and around 10,000 BCE. A significant transition occurred as societies began to shift from a hunter-gatherer lifestyle to sedentary agricultural communities. This Neolithic Revolution[? ?] saw the domestication of plants like wheat and barley and animals such as sheep and cattle. These advancements enabled the establishment of permanent settlements, leading to population growth and the formation of complex societies . The ability to store surplus food led to the differentiation of labor, with individuals specializing in various tasks, thus laying the groundwork for trade and economic systems.[?] .

1.2.3 Industrial Revolution

The 18th and 19th centuries heralded the Industrial Revolution, a period of significant transformation in agriculture. This era saw the advent of mechanization, which supplanted numerous labor-intensive tasks and markedly enhanced productivity and efficiency. Pioneering innovations, such as the steam engine, mechanical reapers, and threshers, revolutionized agricultural methods, thereby reducing the reliance on manual labor and boosting per-acre productivity. This surge in productivity was instrumental in supporting urbanization by ensuring a steady food supply for burgeoning urban populations[?] . Nonetheless, the imprudent application of these technologies precipitated soil degradation, underscoring the critical need for sustainable agricultural practices[?] .

1.2.4 The Impact of Computing on Agriculture

The end of 20th century marked the advent of computational technology, revolutionizing agricultural practices. Precision agriculture emerged as a significant innovation, leveraging data analytics and computer technology to enhance crop yields and optimize resource management. Advanced technologies such as Geographic Information Systems (GIS), remote sensing, and automated machinery facilitated the precise application of inputs, including water, fertilizers, and pesticides, thereby reducing waste and minimizing environmental im-

pact[?] . This technological advancement not only increased agricultural productivity but also generated extensive datasets, which could be further analyzed and utilized to continuously improve farming practices through internet connectivity.

1.2.5 Intelligence in Agriculture

Neural networks, known for their high demand for large datasets, possess the capability to learn intricate patterns within data and make predictions based on these learned patterns[? ?] . The convergence of neural networks and agriculture has unveiled numerous opportunities to revolutionize farming practices. By harnessing the power of artificial intelligence (AI) and machine learning, neural networks can analyze vast agricultural datasets to optimize crop management, enhance yield predictions, and automate labor-intensive tasks[? ?] . The integration of neural networks in agriculture holds significant potential to transform traditional farming methods, rendering them more efficient, sustainable, and profitable. This thesis delves into the applications of neural networks in modern agriculture, with a focus on seed spacing detection, multi-robot-task allocation for pesticide spraying, and row detection for under-the-canopy navigation.

1.3 Thesis Objectives and Structure

1.3.1 Objectives

The primary objective of this thesis is to explore the applications of neural networks in modern agriculture. Specifically, the thesis aims to:

- Detect and estimate seed spacing using neural networks.
- Develop multi-robot-task allocation algorithm for pesticide spraying using neural networks.
- Implement row detection techniques for under-the-canopy navigation using neural networks.

1.3.2 Structure

The thesis is structured as follows:

1. **Chapter 1: Introduction** - Provides a historical background and sets the stage for the thesis.

2. **Chapter 2: Detection and Estimation of Seed Spacing** - Discusses the methodologies and results of using neural networks for seed spacing detection.

3. **Chapter 3: Multi-Task Allocation for Pesticide Spraying** - Explores the development and implementation of multi-robot-task allocation algorithms.

4. **Chapter 4: Row Detection for Under-the-Canopy Navigation** - Details the techniques and outcomes of row detection using neural networks.

5. **Chapter 5: Conclusion** - Summarizes the findings and discusses future research directions.

Chapter 2

Literature Review

2.1 Neural Networks

Neural Networks (NN) are a class of machine learning algorithms inspired by the structure and function of the human brain. These algorithms consist of interconnected nodes, or neurons, that process and transmit information. Neural Networks have gained widespread popularity due to their ability to learn complex patterns and relationships in data, making them well-suited for a variety of tasks, including image recognition, natural language processing, and predictive modeling[?] . The architecture of a Neural Network typically consists of an input layer, one or more hidden layers, and an output layer. Each layer contains multiple neurons that perform computations on the input data and pass the results to the next layer. The neurons in each layer are connected by weighted edges, which determine the strength of the connections between neurons. During the training process, the Neural Network adjusts these weights to minimize a function between the predicted output and the actual output. This process, known as backpropagation, involves iteratively updating the weights using an optimization algorithm, such as stochastic gradient descent, to improve the model's performance. Neural Networks are also known as universal function approximators due to their ability to approximate any function given enough data and computational resources[?] .

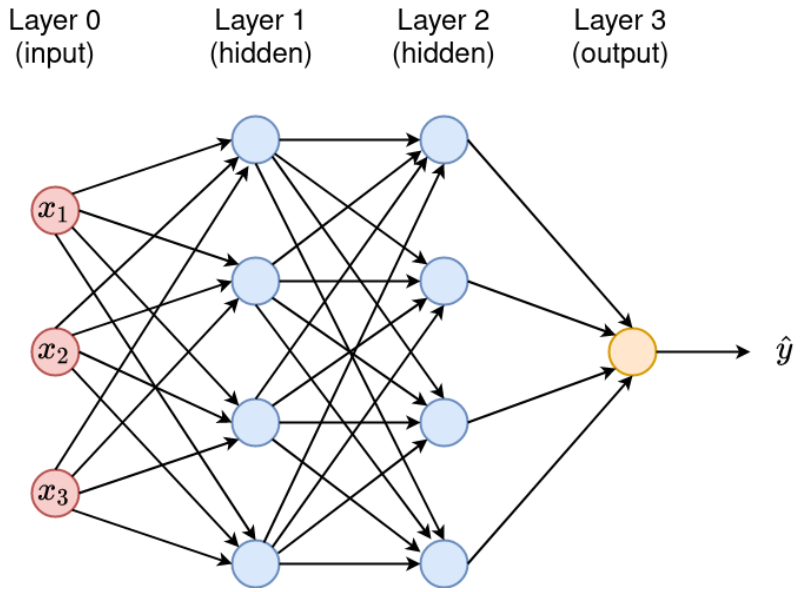


Figure 2.1: A simple Neural Network architecture with an input layer, two hidden layers, and an output layer. Each layer contains multiple neurons connected by weighted edges.

Figure 2.1 shows a simple neural network which has 3 inputs x_1, x_2, x_3 and an output $\hat{y}$. Here the input layer has 3 neurons, the first hidden layer has 4 neurons, and the second hidden layer has 4 neurons. The output layer has a single neuron which outputs the predicted value $\hat{y}$. The connections between the neurons are represented by the weighted edges, and the weights are adjusted during the training process to minimize a function between the predicted output and the actual output.

Since the advancement of computing and the use of GPUs in neural networks, deep neural networks have become more popular, first shown in[?] . Deep Neural Networks (DNNs) are a class of neural networks with multiple hidden layers, enabling them to learn complex patterns and relationships in data. These networks have been successfully applied to a wide range of tasks, including image recognition, speech recognition, and natural language processing. Convolutional Neural Networks (CNNs) are a type of DNN that are particularly well-suited for image processing tasks due to their ability to capture spatial hierarchies in data. Recurrent Neural Networks (RNNs) are another type of DNN that are commonly used for sequential data, such as time series data or natural language data. The Long Short-Term Memory (LSTM) network is a variant of RNN that is designed to capture long-

term dependencies in sequential data. These advanced neural network architectures have significantly improved the performance of machine learning models and have enabled the development of sophisticated applications in various domains.

The functions which these neural networks are trained on can be obtained using supervised learning, unsupervised learning, or reinforcement learning. In supervised learning, the model is trained on labeled data, where the input data is paired with the corresponding output labels. The model learns to map the input data to the output labels by minimizing a loss function that measures the difference between the predicted output and the actual output. In unsupervised learning, the model is trained on unlabeled data, where the goal is to learn the underlying structure of the data. The model learns to identify patterns and relationships in the data without explicit input-output pairs. In reinforcement learning, the model learns to make sequential decisions by interacting with an environment and receiving feedback in the form of rewards or penalties. The model learns to maximize the cumulative reward by exploring different actions and learning the optimal policy. These different learning paradigms enable neural networks to learn a wide range of tasks and make them versatile tools for solving complex problems.

2.2 Applications in Agriculture

Agriculture, a cornerstone of human civilization, sustains not only food production but also economic stability and livelihoods for billions of people worldwide. However, this essential sector continually faces challenges, primarily from pests and diseases that threaten crop health. These adversaries, ranging from minute insects to microscopic pathogens, can significantly impair agricultural productivity, leading to substantial economic losses and threatening food security.

Technological advancements have proven invaluable in addressing these agricultural challenges. The integration of machine learning and computer vision presents a promising frontier for early pest detection, marking the dawn of intelligent and eco-friendly farming practices. Central to this revolution is the significance of high-quality labeled datasets. These datasets

are crucial for training machine learning algorithms, enabling them to identify subtle signs of crop diseases and accurately detect pest presence. Furthermore, the application of Reinforcement Learning (RL) algorithms has the potential to optimize the deployment of robotic agents for pest control, ensuring efficient and targeted interventions. However, the exploration of RL in agricultural applications remains relatively underrepresented in the literature.

This chapter provides a comprehensive review of the existing literature on the applications of machine learning and computer vision in agriculture, with a focus on pest detection and control. Additionally, it examines the potential applications of Neural Networks (NN) and RL algorithms within the agricultural domain. The chapter is structured as follows: Section 2.3 presents an overview of computer vision applications with neural networks, providing a literature review for Chapter 3 and 5, and Section 2.4 explores the potential applications of Reinforcement Learning in agriculture, setting the stage for Chapter 4, which discusses the use of RL for multi-robot task allocation.

2.3 Neural Networks Based Computer Vision Applications in Agriculture

Computer vision makes the machines understand the pixels of an image and make them predict meaningful information or actions. In agriculture, computer vision has been widely used for various applications, including crop monitoring, disease detection, and pest identification. The Computer vision techniques majorly used along with neural networks in agriculture as follows:

- **Image Classification :** The task in image classification is to categorize an image into a specific class. For instance, classifying an image of a plant as either healthy or diseased[?] . Neural networks used for image classification are typically Convolutional Neural Networks (CNNs)[?] . However, there has been a recent trend towards using Transformer-based models, such as Vision Transformer (ViT)[?] , for image classification tasks which are proven to be more effective.

- **Object Detection :** The Task in object detection is to identify and locate objects within an image. Object detection is crucial for identifying which region in the given image contains Object of interest. Object detection models like Faster R-CNN[?] , YOLO[?] , and SSD[?] have been widely used in agriculture[? ?] .

- **Segmentation :** This taks is quite similar to object detection but instead of using bounding boxes, segmentation uses pixel-level masks to identify the objects. Segmentation models like U-Net[?] , Mask R-CNN[?] , and SegFormer[?] have been used in agriculture[? ? ?] .

Chapters 3 and 5 discuss the applications of detection methods in agricultural settings. In Chapter 3, the focus is on seed spacing detection, which is achieved using advanced object detection methods. This chapter elaborates on the process of detecting seed placement and estimating the optimal spacing through the use of an Intersection Over Union (IOU) filter. This filter helps in accurately identifying and classifying the detected seeds, ensuring precise spacing that can significantly improve crop yields.

In Chapter 5, the emphasis shifts to row detection within agricultural fields. This chapter introduces a novel polynomial detection method that simplifies the row detection process. Unlike traditional methods that often require extensive computational power and memory resources, this polynomial detection approach is designed to be efficient, reducing the computational load and memory requirements. This is particularly beneficial for deployment in real-time agricultural applications where resources may be limited.

The development of custom models tailored for agricultural detection tasks is highlighted as crucial due to the limitations of existing models, which are often not optimized for the specific needs of the agricultural domain. For instance, many conventional models may not effectively handle the unique challenges posed by varying crop types, field conditions, and environmental factors.

Furthermore, the novel methods presented in Chapter 5 not only reduce the need for higher computational power and memory but also maintain low latency. This is a significant advantage over traditional segmentation methods, which can be computationally intensive

and slow. The reduced latency ensures that the detection systems can operate in real-time, providing immediate feedback and allowing for timely interventions in agricultural practices.

Overall, these chapters underscore the importance of developing specialized detection models that cater to the unique requirements of agriculture, thereby enhancing efficiency and productivity while minimizing resource usage.

2.4 Reinforcement Learning in Agriculture

Reinforcement Learning (RL) is a subfield of machine learning that focuses on training agents to make sequential decisions by interacting with an environment. RL has gained significant attention in various domains, including robotics, gaming, and finance, due to its ability to learn complex decision-making policies. In agriculture, RL offers promising applications for optimizing resource allocation, crop management, and pest control strategies. However, the adoption of RL in agriculture remains relatively new, with limited studies exploring its full potential. One domain where RL has been applied is crop management support, as detailed in the comprehensive review[?] . Most of the models rely on crop models like[?] used RL to maximize wheat yield under strong limitations of nitrogen pollution. In[?] optimized maize yield while scheduling irrigation strategies using RL Various similar methods have been used in agriculture to optimize crop yield. However, studies related to pest control using RL are very limited such as 4 which discusses the use of RL for multi-robot task allocation in pesticide spraying. This chapter explores the potential of RL algorithms to optimize the deployment of robotic agents for pest control, ensuring efficient and targeted interventions. The chapter provides a detailed methodology for implementing a graph-based RL approach for multi-robot task allocation, highlighting the benefits of using RL in agricultural settings. The chapter also discusses the contributions of the study, its limitations, and the societal impacts of employing RL algorithms for pest control in agriculture.

Chapter 3

Seed Localization System Suite with CNNs for Seed Spacing Estimation, Population Estimation and Doubles Identification

3.1 Abstract

Seed placement information for evaluating planters is obtained by tedious manual methods like using a pogostick or a ruler. To obtain more information and reduce the time required. There is a need for an automated system. The goal of this study is to design such a system with the use of GPS and seed detection.

A 12-row planter was instrumented with cameras and a GPS. The planter was operated at planting speeds of 9.66 kmph and 12.87 kmph with a seed population of 74,131 and 86,4868 seeds per hectare respectively and seed spacing was measured manually using a ruler. YOLOR-P6, YOLOR-CSPX, YOLOX-S, YOLOX-M, YOLOX-L, YOLOX-TINY, and YOLOV4 were trained to detect seeds on a different dataset. An algorithm was designed to estimate the distance between two consecutive seeds by filtering out old detections from the

detector with the use of the IOU metric and GPS stream.

The seed spacing estimator was evaluated on Jensen–Shannon Divergence (JSD), mean, standard deviation, ΔCount, and RMS metrics. Results indicate that the developed system along with the algorithm was able to perform better at 9.66 kmph with JSD distances of 0.223 and 0.244 (74,131 and 86,4868 seeds per hectare respectively) compared to 0.298 and 0.258 (74,131 and 86,4868 seeds per hectare respectively) at 12.87 kmph. Moreover, this algorithm was able to reduce the total time taken to detect seed and estimate seed spacing information from 2hrs by manual method to 1 min 14 seconds using YOLOR-CSPX.

3.2 Introduction

Planting is one of the most important stages in crop cultivation. Producers around the U.S. are adopting wider toolbars and systems equipped with technologies including electric seed meters, hydraulic downforce, pneumatically controlled row cleaners and closing systems, and row-by-row nutrient application. One of the factors motivating the adoption of technology is to achieve accurate seed spacing, and seeding depth. Controlling seed population variability could potentially result in? increase in corn yield percentage by as much as 15%. Additionally, seeding systems accurately maintaining accurate seed spacing and depth have shown corn yield gains by 200 to 1,200 kgpha without changing the planting rate? . All these factors have been pushing the boundaries for more precise operation for a wider spectrum of planting conditions.

Corn is among the most abundantly planted crops in the USA, with approximately 89.5 million acres planted in 2022? . With the rapidly increasing size of farming operations and increasing weather uncertainties, potentially due to climate change? , the number of days when the planter could be operated are decreasing. The planters with wider toolbars and integrated with newer technologies, including electric seed meters, hydraulic downforce and pneumatic controls for cleaning and closing systems, have been designed to accurately place seed at ground speeds of 9.6-12.67 kmph, which is in certain situations not sufficient to plant all the land operation within available days of operation and increase. Manufacturers are continuously making design improvements to develop planting systems which have capabilities to place seed (maintain spacing and trench depth) accurately at faster planting speeds.

As manufacturers innovate planting systems, one of the key metrics for efficiency and operational quantification of a planter is usually done by characterizing seeding depth and seed-to-seed spacing. To iterate through the design process of planters, these measurements have to be taken at numerous sites and test strips comprising thousands of plants. These seeding depth measurements are usually obtained by a tedious process of digging through the soil and finding seed in the trench. Then the depth is measured by the use of a ruler or

digital calipers. Seed-to-seed spacing is measured by using a traditional measuring tape or a pogo stick (Figure. 3.1). These methods, as can be imagined, are very time-consuming, labor-intensive and prone to human errors. However, the above mentioned methods are still the most common approach used by industry, academia, and by service providers to quantify seeding depth and seed spacing ([?] ,[?] ,[?] ,[?]). The current approach also significantly limits the extent to which the data can be collected from the field. Therefore there is a necessity to automate the process to quantify seed placement metrics.

Advances in deep learning techniques are outperforming traditional object detection methods like template matching. Traditional methods involve an expert to hand craft kernels and filters to detect an object. Whereas, in deep learning the network learns the lower-level representation of the dataset which is quite similar to hand-crafted kernels created by an expert without the intervention of a human[?] . So, When trained on a large diverse dataset like Image net[?] the network is bound to learn lower level representations generalizing itself. When a model which is pre-trained on a task, like image classification, is re-trained on a new task, this is known as transfer learning[?] . In this work pre-trained networks on imagenet classification were used as backbones for the detectors.

A recent work[?] showed that it is possible to attain a high AP50 of 0.852 for sugar cane billet detection in a furrow using YOLOv3.This dataset is under the furrow, Hence it has a lot of noise due to lighting, soil derbies etc. This indicates that deep learning neural networks can predict object in the trench.

Therefore, the objective of this study was to use deep learning networks to detect seeds along with GPS information to create an automated system to quantify seed spacing information, with a sole goal to reduce the amount of time spent on gathering the data.

Figure 3.1: Pogo stick device manufactured by sun co-operative

3.3 Setup

3.3.1 Planting System Setup

A Basler (acA1920-40gc, Basler AG, Ahrensburg, Germany) color camera was selected for image acquisition of seeds in the seed trench. The camera provided 1920x1200 pixel resolution (2.3 MP), 42 frames per second (FPS), and gigabit ethernet (GigE) connectivity. A 10 m GigE cable (2000028341, Basler AG, Ahrensburg, Germany) was selected to connect the camera (RJ45) to the data acquisition system (RJ45). The camera body was paired with a ruggedized 12.5 mm fixed focal length Kowa (LM12HC-V, Basler AG, Ahrensburg, Germany) lens. A laser line scanner (OXH7-Z0500.HI0660.VI, Baumer Ltd., Southington, USA) with a measurement range of 100-500mm from the sensor surface; measuring frequency of 1520 Hz, measurement resolution of 4 to 25 μm; and measuring width range of 13 to 66 mm to provide surface profile characteristics was selected for seeding (trench) depth measurement. The camera with lens and laser line scanner together forms a "seed localization sensor" (SLS). A custom SLS mount was designed using CAD models of the camera, lens, and laser line scanner available from the sensor manufacturers. SLS mount design included an enclosure for the back of the camera to limit dust accumulation; a detachable enclosure around the

Figure 3.2: Camera and LED within SLS mount (white 3-D printed part) installed on the row unit

camera lens for ease of focusing during field setup; and an enclosure to install a light source (Fig. 3.2). The SLS mount when mounted on the extension bracket on the planter row unit, provided a consistent distance of 5.6 inches between the lens face and the bottom of the seed trench. The constant distance between lens and target surface would not require users to re-focus cameras with changes in planting depths. The lab tests with a checkerboard showed that the camera has a FOV of 3.25x5.25 inches. Exposure time was set at 300 μs to prevent capturing blurred objects or features in the images. Because the amount of light is proportional to the exposure time, a 213 lm waterproof IP 68 rated three LEDs in aluminum casing with a 150-degree beam angle was utilized as the light source for illuminating the FOV (RLM-W9, Super Bright LEDs, Inc., Earth City, MO. U.S.).

A Case IH 2150 12-row planter with 76.2 cm row spacing (Case IH, Racine, WI, U.S.) was used to collect data during field experiments (Fig. 3.3). SLS units were mounted on four randomly selected rows 1, 4, 9, and 12.

3.3.2 Data Acquisition System

The data acquisition system was developed comprising NI cDAQ 9174; NI c-series module NI-9203; and fanless pc (LEC-2580P-711A, Lanner Electronics, Mississauga, ON, Canada) to capture simultaneous data from all four cameras, laser line scanner, and GPS data. The fanless Industrial PC had Intel® Core™ i7-6600U CPU, 2x RJ45 & 4x PoE High-performance

Figure 3.3: Tractor and planter with SLS for field data collection

IPC with Intel® 6th Gen Skylake Core™ i7 CPU and overall memory of 16 GB. The software was written in the national instruments LabVIEW platform. The DAQ program at the point of initialization recognizes all four cameras CAM3 through CAM 6, connected to the designated ports (LAN 3 through 6 ports on the fanless pc). As the cameras were recognized, the program reads the initialization files cam*.iid files containing camera properties programmed for operation. This file also has a link to the cam*.icd files that have all the necessary camera attribute defined for the desired operation. The program had all laser line scanner inputs (NI9203) and GPS through cDAQ. The program captured frames from each camera at 40 FPS, appends GPS and average trench depth to the frame, and holds the frames in a separate buffer for each camera. When the acquisition is complete, corresponding frames from each camera's buffer are stitched into a single frame and written into an *.avi file, which preserves the frame-by-frame metadata for post-processing. The data from the laser line scanners was not utilized in this study because of the scope of the

project. The DAQ system components were packaged in a hard case providing rapid access to all connectors for laser line scanners and LED lights. Five IP65 RJ45 female to female coupler connectors for remote pc login and four camera systems were mounted on the case to quickly connect and disconnect the system for use in the field. The software program window allowed users to select image size by selecting the "Reduction Factor" tab. The maximum buffer size designed for data recording with a fanless pc was kept at 12 GB (while keeping 4 GB for the LabVIEW program and other pc functions). In this study 2 datasets were collected

- Training Dataset : This dataset was collected to train the detection algorithms.The resolution was 480x175 pixels.
- Validation Dataset: This dataset was collected to validate the overall algorithm (seed spacing distance estimation). The resolution was 480x300

3.4 Methodology

The distance estimation process can be separated into two steps

1. Seed Detection: Seeds are localized using deep learning based object detection algorithms. This involves the following steps described in the section 3.4.1
2. Distance Estimation: Distance estimation is done by filtering seen seeds and by using gps data. This involved collecting the dataset 3.4.2

3.4.1 Seed Detection

Training Dataset

The training dataset was collected using the SLS system at a farmers field in Westmoreland, Kansas with 4 cameras. The collected dataset contained over 166.6 minutes of video stream at 40 FPS with varying speeds of 2mph, 4mph, and 6mph and with varying depths of 1in, 2in,

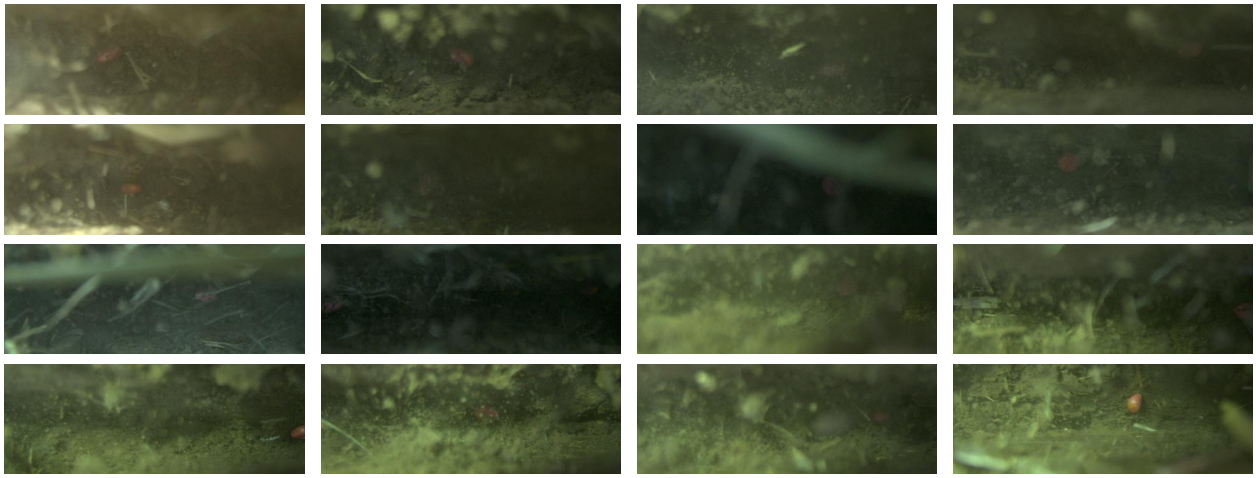


Figure 3.4: Some images in the Dataset

and 3in. This generated over 400,000 images. Since most of these images didn't even contain seeds, especially at 2mph because the frame rate was too high for the population. It would have been very tedious task to sort through all these images, hence a resnet-50 classifier was trained on a small dataset of roughly 2000 images to classify presence or absence of seeds. The classifier attained a precision of 99.2% for the class presence of seed. This reduced the dataset to 40,000 images.

Out of the 40,000 filtered images 8717 images were randomly selected and were labeled using a labeling tool from (LabelBox.com).16 Images are shown in Fig. 3.4, to illustrate how disturbed the dataset is. As it can be seen the brightness ranges drastically. Some of the seeds in this dataset had occlusions from things like grass, debris, soil, etc, it can also be seen that seeds are sometimes out of focus. Furthermore, there is a slight chance for a well calibrated planting system to drop two seeds at the same place this has a very minuscule chance of happening hence it is an edge case.

Manual labelers were tasked with labeling the seeds into 2 classes 1) a polygon to label a seed which was very clear and, 2) a bounding box for seeds which are slightly challenging to view. This helped to further filter the dataset into 2 parts. Ones with the polygon were augmented and challenging images were not augmented.

Figure 3.5: Radom Letter Box Transformed Image

Data Transforms

Following transforms were applied to all the images to ensure that the training process runs smoothly and quickly.

- Random Letter Box Transform: Images were initially resized to 188x512 to maintain the aspect ratio of the images. Then a random number n was chosen to be a real number in the range $(0, 161)$ to stack a rectangle on top of the image with the dimension nx512. The bottom of the image was stacked with $(161 - n)$x512 to bring the size of the image to 512x512. The choosing of random numbers was done to ensure that the model learns the semantic information that back pixels don't matter rather than overfitting the model to look for objects in a set size. The transformed image can be seen in Fig. 3.5.

Data Augmentations

Since the dataset at hand was a small one for a deep learning task, to generalize more accurately data augmentation techniques were used to generate more images[?] .

Only the images with a clearly-visible seeds were augmented.

- Clone Augmentation: This augmentation is very similar to one employed by [?] , However in this study, it was used to insert extra seeds into the image by cloning them rather than generating synthetic images. This augmentation is applied in 2 steps namely.

    1. Register: Segmentation polygon of clearly visible seed is cropped from the image and is stored in a list of set size of 200. If the list size exceeds the set size then an object is randomly dropped out from the list to make space for the new object. This step takes place once for every clearly visible seed.

    2. Augment: A seed was randomly chosen from the list and was added to the image at a random location such that the whole image of the seed fits into the image. This step is repeated N random times which is always less than a given number to avoid computation, in this case this number was set to 8.

- Doubles Augmentation: Multiple seed placement can happen due to an inaccurate seed meter setup, seed meter performance, worn components and inaccurate control setup for planter operation. Double identification is therefore a very critical part for the detector. Hence this augmentation. Similar to clone augmentation described above with a minor change in the augment step, here instead of randomly selecting the insertion point, a point is selected very near to the seed in the image to insert the object from the queue. Although the instances of doubles in the image dataset were very few, detection of doubles is very critical for planting systems operation quantification. Therefore this augmentation was applied such that the dataset would contain at least 15 percent of doubles.

- Random HSL Augmentation: Similar augmentation to the one applied to generate more data by varying HSL color space. The variability applied to the H color channel as 1 ± 0.5, to the S channel as 1 ± 0.25, and the L channel as 1 ± 0.5. This was applied to replicate the various lighting scenarios as the dataset was captured in semi-controlled lighting.

- Random Jitter Augmentation: A random array with a mean of 0 and standard deviation of 1.6 was added to the image. This was done to create noise in the image.
- Random Gaussian Blur: A Gaussian blur augmentation was applied with a randomized kernel size with max size limit being 15. This was done to replicate the out-of-focus seeds as seen in the dataset

Furthermore, all the augmentations were applied in the order described above.Some of the resulting images can be seen in Fig. 3.6

Calibration

The cameras were calibrated using the checkerboard method[? ?]. Over a minute of data was captured on the checkerboard resulting in 2,400 images. Out of 2,400 images, 150 were randomly selected to calibrate the camera.

From the information in the cad files the camera is always at a constant distance away from the bottom of the trench.A checkerboard(24x24 checkers, 25.4 x25.4 mm for each checker) as shown in Fig. 3.7 was placed at the same distance away from the camera. Number of visible boxes along the width (N) were counted.The conversion factor was calculated by $\frac{N}{px}$. Where px is the number of pixels in x direction.

Object Detection

This section goes over the object detection task and the evaluation metrics that were used. For detection or classification tasks in computer vision convolution neural networks are usually used but in the recent work from[?] transformers have been used, However this creates an overhead on the network and it is computationally taxing during training and inference. For CNNs object detection is majorly done in one or two methods 1)Two-stage detection networks 2) single-stage detection networks. Two-stage networks like RCNN and Faster RCNN are very accurate but since they use two stages they are not fast enough to run in real-time. Whereas single-stage detectors like YOLO, YOLOR, YOLOX, Retinanet, Efficeintdet,

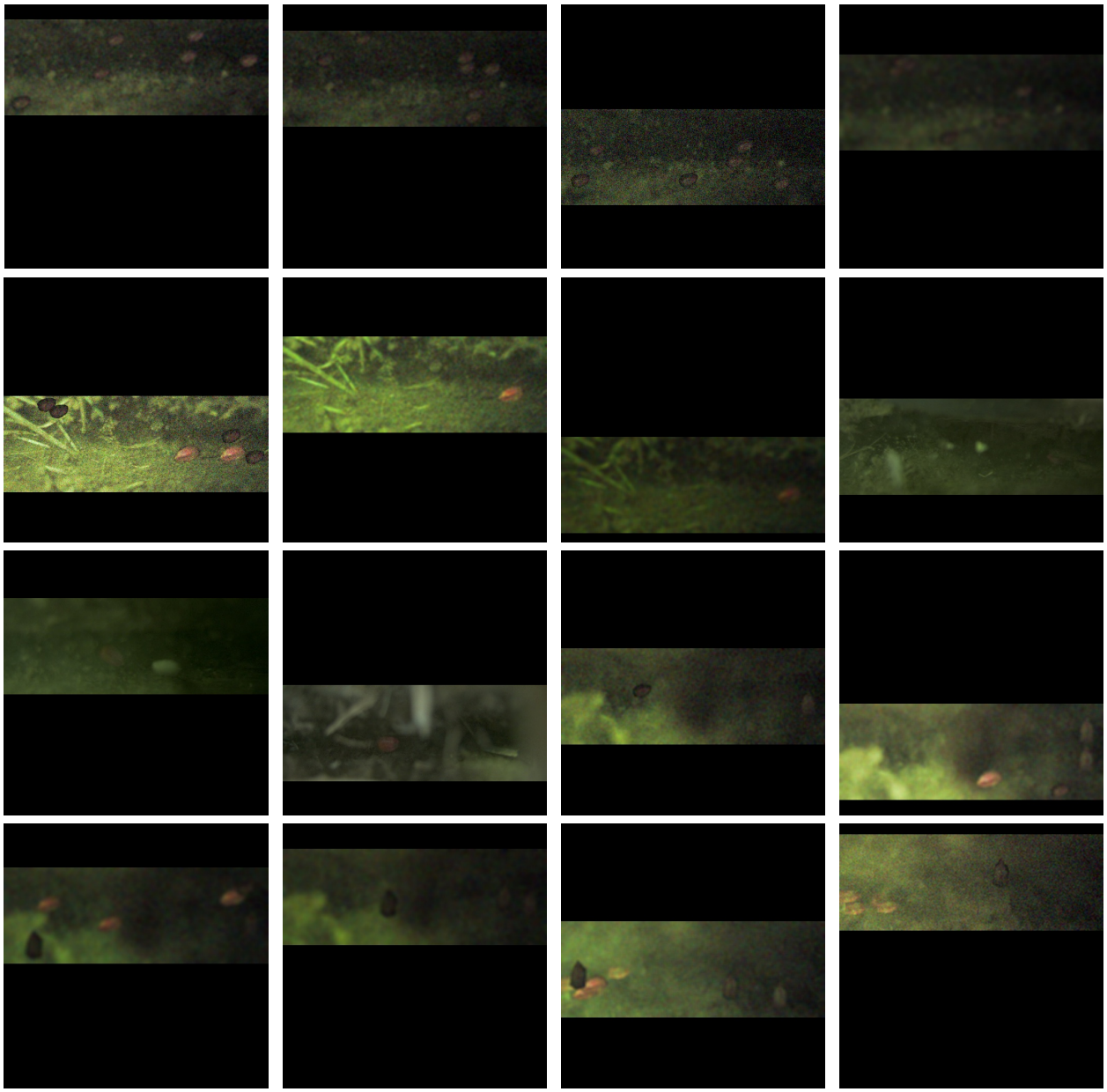


Figure 3.6: Images After Augmentation

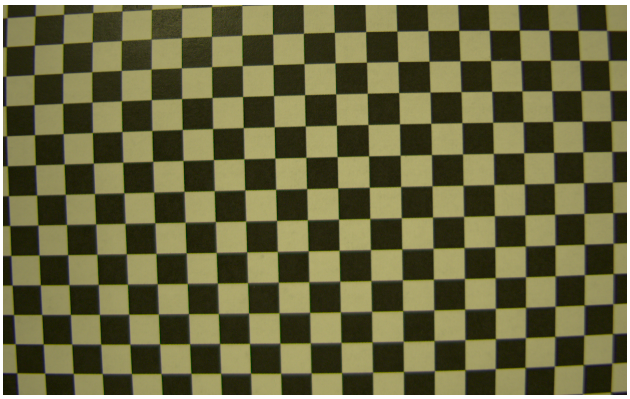


Figure 3.7: Field of view calculation test

and Center net are fast enough to run in a real-time setting. All the state-of-the-art object detectors have a similar structure

1. Back Bone : This part of the network is responsible for taking in an image and converting it into feature sets of various sizes or levels.

2. Feature Pyramid Network (FPN) : This part of the detector is responsible for attaining the multi level from the back bone feature map and fusing them together. This prevents the network from losing information in the very lower levels which might help in predicting the objects which are small.

3. Detection Heads : This part of the detector is responsible for detecting and classifying the objects from the fused outputs given by the Feature pyramid network. There are multiple heads which can be used for detecting objects. The YOLO head has an objectiveness score, number of anchor boxes, number of classes etc. YOLOX has a decoupled head which has classification and regression heads for the bounding boxes. Centernet changes the idea of detecting anchor boxes and instead it detects key-point triplets.

In this study, YOLOX (tiny, small, medium and large), YOLOR (P6, CSPX) and YOLOV4 (CSPX) networks were used, architecture of each network is as follows :

- YOLOV4-CSPX : This network uses CSPDarket53 as backbone, PANet as FPN and a standard YOLO head[?] .

- YOLOX : This network uses CSPDarket53 as backbone, PANet as FPN and a decoupled head for final prediction.

- YOLOR: This network uses CSPDarket53 as backbone, PANet as FPN and a standard YOLO head for detection. Moreover, YOLOR has implicit and explicit networks to aid in detection the explicit part takes in the input and implicit part remains constant throughout a task.

1. *Evaluation Metrics*: Usually, one of the best metrics to evaluate object detection models is mAP (mean Average precision). Since this dataset contains one class, mAP is similar to AP (Average precision).

   IOU: Intersection over union, this specifies how much of an overlap is there between the label and detection

   $$IOU = \frac{Y \cap \hat{Y}}{Y \cup \hat{Y}} \tag{3.1}$$

   Where Y is the predicted bounding box's area and $\hat{Y}$ is the ground truth bounding box's area.

   $Precision(P)$ is calculated by $TP/(TP+FP)$, $Recall$ is calculated by $TP/(TP+FN)$ where TP is True Positives i.e object over a certain IOU threshold, FP is False Positives which are wrong predictions and FN is false negatives i.e objects under the IOU threshold. Now AP is calculated by the area underneath the Precision, Recall curve as shown in Fig. 3.8.

   In case of this study, AP50 Metrics were used. These were evaluated on 2 testset, one with standard split other an augmented testset with just doubles in it. The Table 3.1 shows the results as AP50 Overall and AP50 for doubles.

2. *Object Detection Models* : YOLORP6, YOLOX-TINY, YOLOX-M. YOLO-L. YOLOX S, YOLOR-CSPX and YOLOV4-CSPX were trained on the dataset described in section 3.4.1. These object detection models were evaluated on the test dataset with the Metrics described in the above. The precision recall graph is shown in Fig. 3.8 and

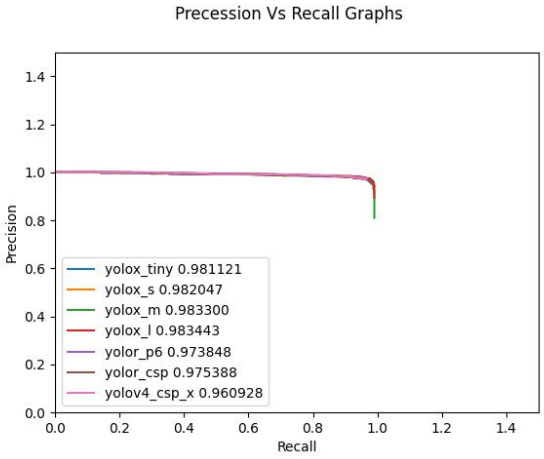


Figure 3.8: Precision Recall Curve On Test Dataset

from AP Overall results and AP50 for doubles in the Table 3.1, it can clearly be seen that the model YOLOX family performs better than YOLOR and YOLOV4.

Table 3.1: Comparison of various Object detection algorithms

S.No	Model	AP50 Overall	AP50 for Doubles
1	YOLOR-P6	0.973	0.898
2	YOLOR-CSPX	0.975	0.892
3	YOLOV4-CSPX	0.960	0.878
4	YOLOX-Tiny	0.9811	0.894
5	YOLOX-S	0.982	0.894
6	YOLOX-M	0.9833	0.900
7	YOLOX-L	0.983	0.886

3.4.2 Distance Estimation

Experimental Dataset

Four datasets were collected at a field in Morton, IL. The camera was calibrated as described in the section 3.4.1 and imaging and GPS data was logged using the SLS system.These datasets have the planter operated at two target speeds with desired seed population densities. The Table 3.2 describes the speeds and population density that were used.

The soil closing system was removed from the planter to aid in quick collection of manual validation data. Seed spacing from each pass was collected using a measuring tape with a

Table 3.2: Operating speeds, seeding populations, mean observed between seeds and standard deviation of the seed distribution

Dataset No	Speed(Kmph)	Population(KSPH)	Mean(m)	σ(m)
0	9.66	74,131	0.16197	0.065
1	12.87	74,131	0.18837	0.099
2	9.66	86,868	0.19289	0.080
3	12.87	86,868	0.180713	0.059

least count of 2.54 mm. Total number of seeds and distances between them were noted for all the seeds that were within the first 15.24 m from the first seed.

The SLS system was started a few seconds earlier than the start of planting. This allowed the user to manually state the first frame where the seed was visible and this was how the data was synced.

3.5 Evaluation Metrics

The following metrics were used to evaluate the algorithm based on the 4 datasets which were collected using a tape measure. All the datasets were converted into a histogram format with number of samples on x-axis and distance between the consecutive seeds on y axis

3.5.1 Jensen–Shannon divergence [JSD]

This is a probability distance metric which finds the distance between two probability distributions

$$JSD(P,Q) = \sqrt{\frac{kld(P,M) + kld(Q,M)}{2}} \tag{3.2}$$

where $M = (P+Q)/2$ and $kld(P,Q) = \sum_i^{i=n}(P_i * \log_2(P_i Q_i))$ which is standard kl divergence. This method was shown to be square of proper distance metric. The predictions from the algorithm were converted into a histogram and then normalized by the total count predicted by it to attain the probability distribution, which is represented by Q. Manually collected distances were converted into a histogram and normalized to attain the probability distribution which is represented by P.

JSD distance is bounded between [0, 1] because it uses $\log_2$. 0 means both the histograms are correlated whereas 1 means they are far away from each other.

3.5.2 Mean and Standard deviation

This is the standard mean and standard deviation metrics which are usually used to understand the data distribution.

3.5.3 Δ Count

This represents the difference between the count of seeds seen by the algorithm in the first 15.24 m from the first detection and the count of seeds collected by the traditional method.

3.5.4 RMS

This is standard root mean square distance metric is given by

$$RMS(P,Q) = \frac{\sqrt{\sum_i^{i=N}(P_i - Q_i)^2}}{N} \tag{3.3}$$

where N is the total number of samples, In this case P_i is the predicted distance between the seeds and Q_i is the hand collected distance between the two seeds.

3.6 Algorithm

The Fig. 3.9 shows the flow chart of the total post processing pipeline. It was developed in python 3.8 on ubuntu 20.04 machine. There were two input data streams one for the GPS which was at a rate of 5 Hz and Images at 40 Hz.

GPS Data

A GPS unit which was running at 5 Hz was used, to make the dataset more consistent with the imaging information each data row was replicated 8 times. Since, it is not required to map all the corn seeds with a same frame of reference as distance between seeds is relative.So, just the velocity information was used. As the tractor was traveling straight in a linear path, heading was considered constant with time. The machine velocity was integrated over time to find the distance between the seeds and then reset to 0 whenever a new seed was registered. It was equivalent to resetting the frame whenever a seed was detected and its distance to nearest seed was calculated.

Image

The DAQ system was acquiring the imaging data at 40 FPS which is equivalent to 40 Hz with a resolution 1920x1200 that was further reduced to 480x300.

Detector

This module performs the task of detecting the seeds in the frame. It was containerized so that different models can be used. The module takes in the image and returns the list of detected bounding boxes in (x_c, y_c, w, h) format.

Update

This module works on the seed queue, takes in velocity and acceleration from GPS and calibration information gathered from Section 3.4.1 and updates the location of the previous detection with respect to this frame. If a seed goes out of bounds it is removed from the queue. As shown in equation 3.4, where x_c is the seed center for along width direction and $\dot{x}$ is GPS estimated velocity and $\ddot{x}$ is calculated acceleration.

$$x_c = x_c - \dot{x}dt + \frac{1}{2}\ddot{x}dt \tag{3.4}$$

IOU Filter

To determine if a prediction in this image is a new detection or an old one, the metric defined in equation 3.1 was used but in this case Y is the predicted bounding box's area and $\hat{Y}$ is the updated bounding box area from the previous detection. If this value is under a certain threshold then the object is a new one or else it is considered to be an old detection and is filtered from the input list. The IOU filter takes in a list of detections and outputs a list of new detections.This metric is very similar to object association step in any object tracker like[?] ,[?] but instead of running a Hungarian matching algorithm to match the targets they are simply eliminated. As tracking objects was not necessary for this study. All the new seeds were inserted into the seed queue.

Distance Calculator

This module is used to calculate the distance between the two unique subsequent seeds as seen in Fig. 3.9, this module takes in the most recent seed from the seed queue and the filtered new seed predictions from the IOU filter and measures the distance between the new detection and the most recent seed.

Algorithm 1 Seed Detection Algorithm

procedure MAIN(G_s, F_s, C_c)
    $distance_estimator \leftarrow SeedDistanceEstimator()$
    $camera_calibration \leftarrow CameraCalibration(C_C)$
    $seed_filter \leftarrow SeedFilter(iou_threshold)$
    for all $F \in F_s, G \in G_s$ **do**
        $I \leftarrow Undistort(F, C_c)$
        $Seeds \leftarrow Detect(I)$
        $camspace \leftarrow convertToCamSpace(Seeds)$
        $new_seeds \leftarrow FILTER(camspace, G)$
        $seeds_distances \leftarrow seeds_distance(new_seeds, gps)$
    end forreturn seeds_distances
end procedure

Algorithm 2 Seed Spacing Calculator

procedure INIT
    $count \leftarrow 0$
    $first_seed \leftarrow 0$
    $res_dist \leftarrow 0$
end procedure
procedure MAIN(new_seeds, lat, long, vel, accel, dt)
    $seeds \leftarrow \emptyset$
    $res_dist+ = first_seed(vel\dot{d}t + 0.5accel\dot{d}t^2)$
    $distance \leftarrow res_dist + new_seeds[0].x_c$
    $i \leftarrow 0$
    ret
    for $seed$ in new_seeds **do**
        $count+ = 1$
        if $i > 0$ **then**
            $distance = seed.x_c - new_seeds[i-1].x_c$
        end if
        $ret.append(distance)$
        $i+ = 1$
    end forreturn ret
end procedure

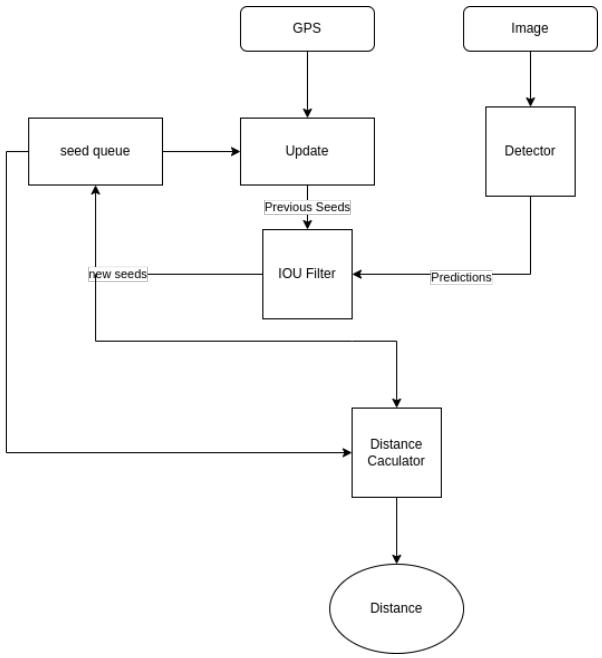


Figure 3.9: Algorithm Overview

Algorithm 3 Seed Filter

```
1: procedure INIT(iou_threshold)
2:     seed_queue ← ∅
3:     iou_threshold
4: end procedure
5: procedure MAIN(detections, lat, long, vel, accel, dt)
6:     update(lat, long, vel, accel, dt)
7:     sort(detections)
8:     new_seeds = filter(detections)
9:     for seed in new_seeds do
10:        seed_queue.append(seed)
11:    end forreturn new_seeds
12: end procedure
13: procedure UPDATE(lat, long, vel, accel, dt)
14:    for seed in seed_queue do
15:        seed.x_c− = velocity(dt) + ½accel(dt)
16:        EliminateSeedIfOutOfBounds()
17:    end for
18: end procedure
19: procedure FILTER(detections, iou_thres)
20:    ret ← ∅
21:    for detection in detections do
22:        flag ← FALSE
23:        for seed in seed_queue do
24:            if calculateIOU(seed, detection) > iou_thres then
25:                seed_queue.remove(seed)
26:                break
27:                flag ← TRUE
28:            end if
29:        end for
30:        if not flag then
31:            ret.append(detection)
32:        end if
33:    end forreturn ret
34: end procedure
```

Table 3.3: Evaluation Metrics (section 3.5) for the seed spacing estimation algorithm for the different detection models

Model Name	Dataset	JSD	RMS (m)	Count (N)	Mean (m)	σ (m)	FPS (N)	Δcount
YOLOX-S	0	0.195	0.012	98	0.163	0.097	10.955	4
	1	0.262	0.021	78	0.208	0.16	11.369	-4
	2	0.234	0.017	84	0.189	0.106	11.041	0
	3	0.299	0.018	73	0.223	0.135	11.028	-5
YOLOV4-CSPX	0	0.223	0.012	88	0.182	0.104	21.611	-6
	1	0.313	0.024	78	0.208	0.167	22.743	-4
	2	0.247	0.015	83	0.19	0.114	22.459	-1
	3	0.294	0.02	68	0.239	0.153	23.566	-10
YOLOR-CSPX	0	0.223	0.012	100	0.16	0.096	23.719	6
	1	0.298	0.018	84	0.193	0.131	25.405	2
	2	0.244	0.015	86	0.184	0.107	24.271	2
	3	0.258	0.017	75	0.217	0.131	25.431	-3
YOLOR-P6	0	0.241	0.011	89	0.18	0.101	21.127	-5
	1	0.314	0.019	81	0.2	0.149	22.626	-1
	2	0.249	0.015	85	0.187	0.109	21.723	1
	3	0.268	0.02	67	0.243	0.138	23.101	-11
YOLOX-M	0	0.255	0.011	92	0.174	0.104	5.876	-2
	1	0.301	0.021	79	0.205	0.162	6.031	-3
	2	0.252	0.015	86	0.184	0.113	6.03	2
	3	0.303	0.019	73	0.223	0.144	6.003	-5
YOLOX-Tiny	0	0.25	0.013	78	0.205	0.111	13.766	-16
	1	0.234	0.026	60	0.266	0.233	14.88	-22
	2	0.283	0.016	71	0.222	0.115	14.487	-13
	3	0.299	0.019	65	0.251	0.143	12.927	-13
YOLOX-L	0	0.237	0.011	89	0.18	0.098	4.022	-5
	1	0.267	0.02	78	0.208	0.159	3.96	-4
	2	0.225	0.012	82	0.193	0.103	3.979	-2
	3	0.276	0.017	70	0.232	0.13	3.888	-8

3.7 Results and Analysis

The accuracy comparison of the 7 models is shown in Table 3.3. The Fig. 3.10 shows the box plots for distance estimator from the results in Table 3.3 for the 4 datasets in the Table 3.2. From the Fig. 3.10 it can be observed that the spacing estimator worked better in datasets with lower speeds (Datasets 0 and 2) which was indicated by lower values of JSD distances, and RMS distances. This was due to the fact that the camera captured frames covering all of the trench with an overlap at 40 FPS for lower speeds. However, in case of higher speed the RMS and ΔCount were not as effective as at lower operating speeds, which can clearly be seen for JSD Box plot from Fig. 3.10. This was due to the fact that at higher speed the camera had lesser overlap and few blind spots in the trench, which resulted in the camera missing some seeds (ΔCount). The ΔCount Box Plot for dataset 3, which had the highest speed and highest population density, clearly showed the offset from the other three datasets primarily due to greater number of seeds in blind spots, which was expected. Moreover, the predicted σ values were high compared to the dataset's σ (Table 3.2). This disparity was potentially due to the fact that GPS unit used during field experiment did not have an RTK fix. The reduced accuracy added the variance from the GPS into σ of the algorithm. This could potentially be reduced by using a GPS unit with 40 Hz update rate and an RTK fix.

The total time taken for distance estimation for 4 datasets shown in Table 3.2 by the 7 models is given below:

1. YOLOX-tiny : 120.22 seconds
2. YOLOX-S : 155.6 seconds
3. YOLOX-M : 263.97 seconds
4. YOLOX-L : 394.71 seconds
5. YOLOR-P6 : 79.4 seconds
6. YOLOR-CSPX : 74.91 seconds

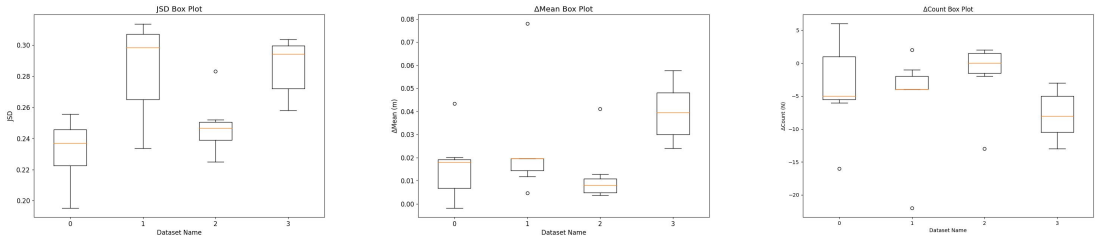


Figure 3.10: Box plots for JSD, ΔMean and ΔCount for the distance estimator using the 7 models

7. YOLOV4-CSPX : 78.14 seconds

From the AP50 metric from the Table 3.1 it can be observed that YOLOX family showed better accuracy by at least 1% AP50 for all the models compared to YOLOR but their inference time for the smallest model (YOLOX-tiny) was high (120.22 sec) compared to the largest in YOLOR (YOLOR-P6) family (79.4 sec). The results exhibited that YOLOR-Family could potentially provide better solutions for real time data analytic capabilities. From the Table 3.3 it can be observed that for the YOLOR family YOLOR-P6 has JSD distances of 0.241 and 0.249 for lower speeds and 0.314 and 0.268 for higher speeds with RMS values of 0.18 and 0.187 and 0.2 and 0.243 respectively. YOLOR-CSPX has JSD distances of 0.223 and 0.244 for lower speeds and 0.298 and 0.258 for higher speeds with an RMS of 0.16 and 0.193 and 0.184 and 0.217 respectively. The observed results indicated that YOLOR-CSPX could provide greater seed detection and seed spacing estimation accuracy and hence be a more viable solution compared to YOLOR-P6 as it had higher FPS and lower error.

3.8 Conclusions

From the results following conclusions can be drawn

- For seed detection accuracy YOLOX family appeared to perform better than YOLOR family (Table 3.1). However, for this dataset YOLOR-CSPX proved to have better FPS (approximately 25) compared to YOLOX Family (Table 3.3). Detection models with

higher FPS would provide advantage of faster computation when developing real-time solutions.

- The results also indicated that the accuracy of the seed placing estimator dropped at 12.87 kmph with JSD Distances of 0.298 and 0.258 (Dataset 1 and 3 respectively) compared with 0.223 and 0.244 (Dataset 0 and 2 respectively) at 9.66 kmph. This is due to blind spots in the trench.These results indicated that the FPS of the camera puts a threshold on accuracy relative to speed of planting.

- By using SLS system and YOLOR-CSPX as seed detector algorithm, the total time taken to detect and quantify spacing for all dataset was reduced by a factor of 96 (74.91 seconds), compared to manual method (2 hours).

- This study can be adapted to other types of seeds, such as soybeans, sunflower, cotton and others, by retraining the detector on those specific types of seeds. The detector and tracker are two separate modules in this study, and the retraining process would involve specifically targeting the detector module to recognize and distinguish the new type of seed. This would allow the study to be applied to a wider range of crops and could potentially have useful applications in various agricultural settings.

3.9 Future Work

The accuracy of distance estimator can be improved by using a higher frequency GPS or by using an IMU as a Kalman filter will do a better job at updating the position of camera rather than simple integration[?] . Furthermore, real time capability of seed spacing estimation can be attained by pruning the detection networks of non-significant weights or by designing special hardware to run the detection networks.

Acknowledgment

We would like to thank CNHi for their support of a tractor and a planter for field data collection; and Aashavi Dua, Bhaskar Upadhyay Aryal and Rahul Singh for the help with labeling and Jose Perrie for the help with Training data collection.

Chapter 4

Graph based Reinforcement Learning Approach For Multi-Robot Task Allocation in Pesticide Spraying

4.1 Abstract

In multi-robotic pesticide delivery tasks, traditional task planning and allocation strategies often require prior knowledge of crop infection levels, typically obtained through preliminary field scouting. This assumption is becoming increasingly obsolete with the advent of autonomous robots equipped with integrated sensing and spraying capabilities.A similar situation applies to many other applications of agricultural robotics. Furthermore, conventional methods struggle to scale effectively with the increase in the number of infected plants, presenting significant challenges in expansive agricultural settings. To overcome these limitations, this study reformulates the problem within the context of a Partially Observable Markov Decision Process (POMDP). A novel architecture, similar to Cross Transformer is designed to address these challenges. Additionally, comparative evaluations of this architecture against expert models trained on out-of-distribution datasets have been conducted to verify its generalization capabilities.

4.2 Introduction

Multi-Robotic Systems are increasingly becoming integral in various sectors such as agriculture, mining, and space exploration ([? ? ?]). A common characteristic of these systems is their interaction with the environment, which varies significantly across different applications. For instance, in agriculture or household robotics, the environment is generally known beforehand, but specific details like the level of task completion, such as the extent of pesticide or herbicide spraying needed in a farm, are not predetermined due to unknown factors like plant infectivity.

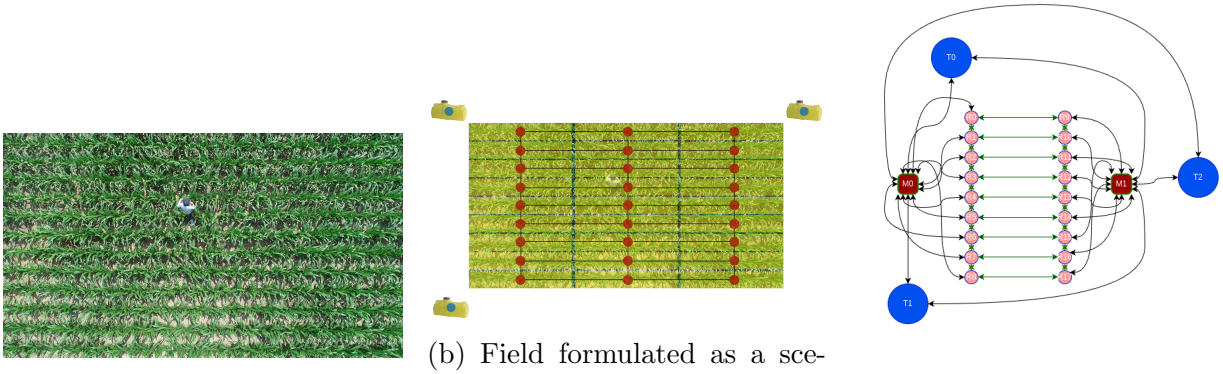
The planning and decision-making in multi-robotic systems can typically be categorized into three main approaches:

- *Centralized Planning*: Here, a central system undertakes the decision-making and planning, which the robots then execute. This approach, while straightforward to design, faces practical challenges, especially the 'curse of dimensionality', making it impractical for complex scenarios[?] .

- *Decentralized Planning*: In this model, each robot autonomously makes decisions and plans tasks within its local environment, contributing to the collective objective. This approach enhances robustness against individual robot failures but poses challenges in maintaining global coherence, as each robot must align its actions with overarching goals[?] .

- *Hybrid Planning*: Combining elements of both centralized and decentralized planning, this approach involves some level of centralized decision-making, complemented by autonomous planning by the robots. For example, a central planner might assign location targets, while the robots independently navigate and execute tasks[?] .

In the hybrid approach, centralized task planning sets broad objectives, while local navigation and control duties are relegated to individual robots. This method skillfully merges the strengths of both centralized and decentralized systems, enhancing efficiency and flexibility. Such an arrangement is ideally suited for pesticide spraying tasks where robots



(a) Aerial view of a field

(b) Field formulated as a scenario

(c) Field Nodes Compressed

Figure 4.1: In Figure 4.1a, an example farm requiring pesticide application is depicted. This farm is segmented into various zones, as detailed in Section 4.4, highlighted with a yellow overlay. The centroids of these zones are marked with red dots, while the blue dots indicate the locations of tanks in the headland. As illustrated in Figure 4.1c, the graph representing these details is subsequently compressed to manage the increasing number of nodes, thereby enhancing the scalability of the problem. The labels M0 and M1 in the compressed graph are muxes to join the edges used to aid in visualization.

manage intricate navigation and path planning, and a separate system coordinates higher-level scheduling decisions, like determining whether a robot should scout or be refilled. The problem of task allocation in multi-agent systems is often likened to a variant of the Multiple Traveling Salesman Problem (MTSP), which is a well known NP-Hard problem[?] . A variety of innovative strategies have been developed to address this issue, each offering distinct advantages and suitable applications.

Auction-based methods, introduced by Bertsekas et al.[?] , foster a dynamic, market-like setting where agents competitively bid for tasks, effectively emulating real-world economic behaviors and proving versatile in changing environments. Lagoudakis et al.[?] adapted this model to robotic task allocation, demonstrating its applicability in real-time and fluctuating conditions. Conversely, Mixed Integer Optimization (MIO) offers a structured solution for complex allocation problems that incorporate multiple constraints. Fischetti et al.[?] highlighted the efficacy of MIO in sophisticated scheduling tasks, while Lippi et al.[?] applied it in multi-agent settings involving robots and humans, optimizing aspects like total task completion time and showcasing MIO's capacity to manage diverse and constraint-laden scenarios. However, these methods face challenges when the required work or the cost of the tasks is

not predetermined. Furthermore, as indicated by[?] , MIO techniques suffer from the curse of dimensionality, with computation time increasing exponentially as the number of infected zones rises.

Reinforcement Learning (RL) has emerged as a powerful method, particularly suited to uncertain and dynamically changing environments. Shibata et al.[?] successfully employed RL in multi-agent settings for dynamic task allocation in cooperative robot transport, illustrating its capacity for continuous adaptation and learning. Vinyals et al.[?] further explored RL's impact in complex multi-agent interactions, providing insights into distributed decision-making processes.[?] used RL to solve Multi robot task allocation in warehouse settings with an attention inspired arhciture.

Additionally, the convergence of Graph Neural Networks (GNNs) with Reinforcement Learning marks a significant trend, particularly in addressing sophisticated decision-making challenges like the Travelling Salesman Problem (TSP). Kool et al.[?] contributed to this field by integrating a contextual node into an Encoder-Decoder architecture, employing the REINFORCE algorithm for training. This method effectively captures the intricate dependencies characteristic of such problems. Moreover, the ECO-DQN framework[?] introduces novel concepts in RL for combinatorial optimization, such as test-time exploration and action reversibility, enhancing performance in complex, graph-based tasks. The synergistic use of GNNs and RL enhances decision-making capabilities in multi-agent systems, opening avenues for efficient, scalable solutions in dynamically evolving environments.

4.2.1 Contributions of this Study

This study makes significant contributions to the field of robotics and automated systems, specifically focusing on the application of Multi Robot Pesticide Spraying in agricultural settings. The key contributions of this research are:

1. The conceptualization and framing of Multi Robot Pesticide Spraying within the framework of Partially Observable Markov Decision Processes (POMDPs). This approach offers a more robust and realistic representation of the operational environment, ac-

counting for uncertainties and partial observability inherent in agricultural settings.

2. The creation of an innovative attention module inspired by cross-attention mechanisms. This module effectively incorporates the states of individual robots, the collective state of other robots, and overarching objectives to optimize an objective.

3. Demonstrating the capability of Graph Neural Networks (GNNs) in effectively solving the Multi Robot Task Assignment problem within a POMDP framework. This research shows that GNNs can achieve near-optimal solutions, employing a value function approach rather than relying on simpler, less effective greedy baselines.

4.2.2 Limitations

This work assumes the layout of the farms to be similar to grids and the speed of the robots to be constant throughout the farm. However, this is not usually the case. In reality, farm layouts can vary significantly, and the speed of the robots often fluctuates due to constraints on dynamics and traction. This study is preliminary and aims to evaluate whether the created model can generalize well under these simplified assumptions.

4.2.3 Societal Impacts

With the ever growing population, the requirement of food is ever growing. This work enables management of lower cost robots which can sense and act in the environment hence reducing the excessive need for pesticides and also making the scouting or surveillance drones obsolete for agricultural purposes.

4.3 Background

4.3.1 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) algorithms aim to learn stochastic policies $\pi_\theta : \mathcal{S} \times \mathcal{A} \rightarrow \Delta^{|\mathcal{A}|}$, parametrized through θ (such as DNN weights), where $\mathcal{S}$ and $\mathcal{A}$ denote the sets of states

and actions. The environment is initialized to a state $s_0 \in \mathcal{S}$. At any time instant t when the environment is in state $s_t \in \mathcal{S}$, the agent chooses an action $a_t \sim \pi_\theta(s_t) \in \mathcal{A}$. In response, the environment probabilistically transitions to another state s_{t+1} while returning a reward $r_t \in \mathbb{R}$ to the agent, based on the action's effectiveness. The goal is to learn a policy π_θ to increase the cumulative rewards accrued over all time instants, until time T when the episode terminates. The cumulative reward of episode $\tau = (s_0, a_0, r_0, s_1, ..., s_T)$ is given by,

$$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t, \tag{4.1}$$

where $\gamma \in (0, 1]$ is the discount factor. For conciseness, the episode's probability is denoted as $\pi_\theta(\tau)$, although it also dependents on state transition probabilities $p(s_{t+1}|s_t, a_t)$.

Policy gradient methods[?] are a key class of DRL methods, that apply stochastic/minibatch gradient ascent to learn parameter θ that maximizes the objective function shown below,

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]. \tag{4.2}$$

It can be established that the gradient of $J(\theta)$ is given by the following expression,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)\nabla_\theta \log p(\tau)]. \tag{4.3}$$

Policy gradient algorithms use Monte Carlo estimates to approximate expectations. The variance in $R(\tau)$ can be very high. Actor-critic methods address this issue by using two DNNs, an 'actor' that applies policy gradient, and a 'critic'. The latter network is trained to estimate either $V : \mathcal{S} \to \mathbb{R}$ or $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, which are the expected cumulative rewards, $V_\theta(s) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)|s_0 = s]$, and $Q_\theta(s) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)|s_0 = s, a_0 = a]$. The actor network uses the advantage function $A_\theta(s, a) = Q_\theta(s, a) - V_\theta(s)$, which can be interpreted as the overall benefit of picking action a when the e

nvironment is at state s. Given an episode τ (where $\tau \sim \pi_\theta$), the gradient may be

estimated as,

$$\nabla_\theta J(\theta) \approx \hat{g} \ = \sum_t \nabla_\theta \log(a_t|s_t)\, \hat{A}_t. \tag{4.4}$$

The quantity $\hat{A}_t$ is the estimated advantage at t, using the critic's output.

Proximal Policy Optimization (PPO)[?] is an actor-critic method that stabilizes the learning algorithm by avoiding large increments to θ. If θ' and θ are the parameter's old and new values, let their ratio at instant t be represented as $\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}$. PPO applies a clipping function to restrict ρ_t to lie within $1 \pm \epsilon$. It makes use of a clipped surrogate objective function,

$$L^{\text{CLIP}}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \tag{4.5}$$

These methods can be best understood in terms of the advantage function A() in such a manner that the probabilities $p(\tau)$ of episodes τ that fetch higher rewards are incrementally increased. The discounted

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right] \tag{4.6}$$

Here, $J(\theta)$ represents the expected return, π_θ the stochastic policy, τ a trajectory, r_t the reward at time t, and γ the discount factor. The policy gradient is estimated as:

$$\hat{g} = \hat{\mathbb{E}}_t\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t\right] \tag{4.7}$$

PPO[?] , a variant of policy gradient methods, addresses stability and complexity issues found in similar algorithms. It limits policy updates to ensure more stable learning, using a clipped surrogate objective function:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \tag{4.8}$$

In this function, θ are the policy parameters, $r_t(\theta)$ the action probability ratio, $\hat{A}_t$ the advantage estimator, and ϵ a hyperparameter controlling the clipping range. This approach enables PPO to effectively balance exploration and exploitation, making it suitable for applications

with high-dimensional action spaces.

4.3.2 Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) have emerged as a powerful tool for learning representations of graph-structured data. Unlike traditional neural networks that operate on Euclidean data (like images or text), GNNs are designed to capture the relationships and interactions in data that can be naturally represented as graphs.

A key concept in GNNs is the neighborhood aggregation or message passing scheme, where a node's representation is updated by aggregating features from its neighbors. This process is formulated as:

$$h_v^{(l+1)} = \text{COMBINE}^{(l)}\left(h_v^{(l)}, \text{AGGREGATE}^{(l)}\left(\left\{h_u^{(l)} : u \in \mathcal{N}(v)\right\}\right)\right) \tag{4.9}$$

where $h_v^{(l)}$ is the feature vector of node v at layer l, and $\mathcal{N}(v)$ denotes the neighbors of v. The functions AGGREGATE and COMBINE are learnable and vary across different GNN architectures.

Several types of GNNs have been proposed, such as Graph Convolutional Networks (GCNs)[?], Graph Attention Networks (GATs)[?], and Graph Isomorphism Networks (GINs)[?]. Each offers unique approaches to aggregating and transforming node features. For example, GCNs use a spectral approach based on graph Laplacians, GATs introduce attention mechanisms to weigh neighbor contributions, and GINs aim to learn representations that capture graph isomorphism.

GNNs have shown remarkable success in various tasks such as node classification, graph classification, and link prediction. They are particularly valuable in domains like social network analysis, molecular biology, and recommendation systems, where data is inherently graph-structured.

One of the challenges in GNNs is handling large-scale graphs due to computational and memory constraints. Recent advancements like GraphSAGE[?] and GNNs with sampling

strategies have addressed this issue by enabling efficient learning on large graphs.

4.4 Problem Formulation

The study delineates an agricultural field managed by a multi-robot system for pesticide spraying, characterized by parameters $headland_x$, $headland_y$ (representing buffer zones around the field to facilitate machinery maneuverability), $plant_{spacing}$ (distance between plants within a row), $row_{spacing}$ (distance between rows), and $tanks_locations$ (indicative of pesticide refill stations). Based on these parameters, a grid overlay is constructed to model the field, dividing it into $P = R \times C$ zones for targeted pesticide application.

Each zone, p_i, is defined by its centroid coordinates x_i and y_i, normalized against the total field dimensions to ensure a consistent representation across varying field sizes. Specifically, x_i and y_i are scaled from 0 to 1, where 0 corresponds to the edge of the field inclusive of the headland ($headland_x$ for x_i and $headland_y$ for y_i), and 1 represents the opposite boundary. This normalization facilitates the precise positioning of zones within the field's spatial context.

The predefined distance x is defined as the amount of distance travelled by the robots in the time it takes for one robot to fill its tank. This parameter introduces a practical constraint that directly impacts the operational efficiency and strategic planning of the robotic fleet.

Infectivity, in_i, quantifies the proportion of plants within a zone requiring pesticide application. For example, in a zone where half the plants are infected (5 out of 10), the infectivity level is set to 0.5, indicating that the robot should spray pesticide over 50% of the zone's area to adequately address the infection. This infectivity level is not known from the beginning, requiring the robots to scout the area to determine the specific needs of each location. This element introduces the Partially Observable Markov Decision Processes (POMDP) aspect of the problem, accommodating the uncertainty and partial observability inherent in the operational environment.

Moreover, the robots are constrained to travel between rows, as moving over the crops is not permissible. With each row, R, having two entrances—one at the start and one at the

end—this results in a total of $2 \times R$ possible actions. Additionally, considering T tanks, the total number of actions extends to $2R+T$. Each tank is assigned normalized x_i, y_i positions, analogous to the zoning system for the field.

For the robots, each state is represented by x_r, y_r, and s_r, where x_r, y_r are the normalized positions of the robot, and s_r indicates the normalized volume of sprayer fluid the robot is carrying. It is presumed that all robots possess the same maximum fluid capacity, ensuring uniformity in operational capability.

This comprehensive framework allows for the precise modeling of the field and the targeted application of pesticides. By incorporating spatial normalization, a constant effective distance of application, and a quantifiable measure of zone infectivity, we lay the groundwork for an efficient and responsive multi-robot task assignment strategy within a POMDP framework.

For example, the farm in the Figure 4.1a is divided into 9×3 zones in the Figure 4.1b, it is assumed there are 3 tanks in the headland. Where the distance between two red nodes is the amount of time it takes for a robot to fill its tank.

4.5 Methodology

4.5.1 State Representation

The state of the system is captured by a graph $G = (N, E)$, where $N = C \times R + T$ denotes the graph's nodes, combining zones determined by rows R and columns C, and the positions of tanks T and E are the paths the robot can take. Attributes for each node include spatial coordinates (x_i, y_i), a robot presence flag (r_i), a designation of the node as either a plant zone or tank (p_i), and the infectivity level (in_i), which is set to -1 for zones not yet observed.

To efficiently represent the extensive number of plants, a cosine compression approach is utilized, resulting in a compressed graph G_c with $2R + T$ nodes. This method adjusts the infectivity features of entry and exit nodes in each row using a cosine function to reflect proximity and state information essential for task assignment. For a given node n, representing

an entry or exit of a row, the aggregated infectivity feature is updated as follows:

$$inf_n = \frac{\sum_{i=0}^{C-1} \cos\left(\frac{i\pi}{C}\right) \cdot in_i}{\sum_{i=0}^{C-1} \cos\left(\frac{i\pi}{C}\right)} \tag{4.10}$$

When a robot is positioned at column v within row r, with n denoting the entrance node, the position pos_r is calculated by:

$$pos_r = \cos\left(\frac{v\pi}{2C}\right) \tag{4.11}$$

Similarly, the sprayer fluid level, considering the robot's position, is given by:

$$sc_r = \cos\left(\frac{v\pi}{2C}\right) \cdot s_r \tag{4.12}$$

where s_r denotes the sprayer fluid level of the robot. This compression strategy reduces the graph to G_c, maintaining vital information with a reduced node set, thus enhancing decision-making capabilities within a POMDP framework through improved scalability and efficiency.

Thus attributes of each compressed node n are thus specified as

$$N_n = \left\{ x_n, y_n, t_n, p_n, inf_n, \frac{1}{C}, pos_r, sc_r, pos_{or}, sc_{or} \right\}$$

.

4.5.2 Action Representation and Reward

Action Space: In the constrained action space, each node within the state representation corresponds to a potential action that the robot could execute. However, not all actions contribute meaningfully to the task's objectives. Consider, for example, the scenario depicted in Figure 4.1b, where a robot has completed disinfection of the topmost row. Subsequently, including this action in the action queue becomes redundant. Moreover, if the robot's tank is full at this juncture, visiting a refill station would not offer any additional benefit. To address

these inefficiencies, we construct a refined list of actions. This list exclusively comprises actions that yield tangible benefits, such as targeting rows where observations indicate a zero disinfectivity level or where disinfection is scheduled but not yet commenced. Furthermore, actions involving refill stations are excluded from the list if the robot in question already possesses a full tank. By filtering out non-contributory actions in this manner, the policy is able to converge more rapidly.

Reward: The reward function that was used in this study was as follows. This was done to minimize total distance travelled while ensuring safe exploration.

$$\text{total_reward} = -\left(\frac{\sum_{i=1}^{N}|inf_i|}{N}\right) \times 0.5 - cost \times 0.25 + 0.1 \times \text{dis_inf} + 0.05 \times \text{obs} \quad (4.13)$$

Here *cost* is the total distance travelled by the robot, dis_inf are the total number of zones disinfected by the robot, obs are new zone observation by the robot.

4.5.3 Model Architecture

The state, represented as a tuple of node features from n previous observations, is critical for modeling changes in infectivity. This tuple is input into a Graph Neural Network (GNN), which may be implemented as one of the following architectures: a) Graph Convolutional Network (GCN), b) SAGEnet, c) Graph Isomorphism Network (GIN), or d) Graph Attention Network (GAT). Our experiments have demonstrated that SAGEnet is particularly efficient in terms of memory usage, thus facilitating the processing of larger minibatch sizes. Conversely, GIN was found to be the most demanding in memory resources. Additionally, a residual connection is employed within the network to enhance the gradient flow, as depicted in Figure 4.2. This study was evaluated with SAGEnet as it had the minimal memory requirement for the selected Embedding size.

Positional embeddings of size $1 \times \text{MaxNodes} \times N$ are subsequently added to the nodes. This is followed by the introduction of temporal embeddings, sized $n_{\text{obs}} \times 1 \times N$ for each node. These embeddings are then flattened, and a causal Transformer architecture is utilized

to generate a temporal representation of the data.

Finally, embeddings are extracted from the nodes in the action queue. Action logits are subsequently derived through either a cross-attention or task transformer mechanism. This process involves the interaction between the extracted embeddings, the embeddings of the robot queue, and the embeddings of the robot of interest. Such an arrangement facilitates a nuanced understanding of the dependencies between the robot's current actions and its queue, optimizing the decision-making process.

Cross Attention Transformer: The primary objective of the Cross Attention Transformer is to leverage cross-attention between the robot's action queue and the collective task queue, allowing for focused correlation maximization between the current robot action and the selected task. The forward pass in cross attention transformer is as follows:

Algorithm 4 CrossAttentionTransformer($X_{actions}, X_{robots}$)

Initialize W_Q, W_K, W_V
$Q \leftarrow W_Q \times norm(X_{actions}), K \leftarrow W_K \times X_{robots}, V \leftarrow W_V \times X_{robots}$
$out \leftarrow multiheadattention(Q, K, V)$
$out \leftarrow mlp(out) + ffn(norm(out))$

In Algorithm. 4, mlp is a multi layer perceptron, and ffn is a feed forward neural network one similar to the one used in[?] .

Value Network: Value network is a multi-layer perceptron, which takes in the max on actions dimension for action embeddings, robot states and attention scores such that the input dimension is 3× embedding dimension.

4.6 Experimental Setup

The training of reinforcement learning agents on an Ubuntu 22.04 machine with two RTX 4090 using PyTorch 2.3 and RLlib 2.9.1, utilizing Proximal Policy Optimization (PPO). SAGEnet[?] —was trained over 4 million steps (one week in real time) in varied scenarios, where both the farm layout, number of robots and plant infectivity are randomized. Plant infectivity is varied using a Laplacian function whose centers are randomly chosen over the

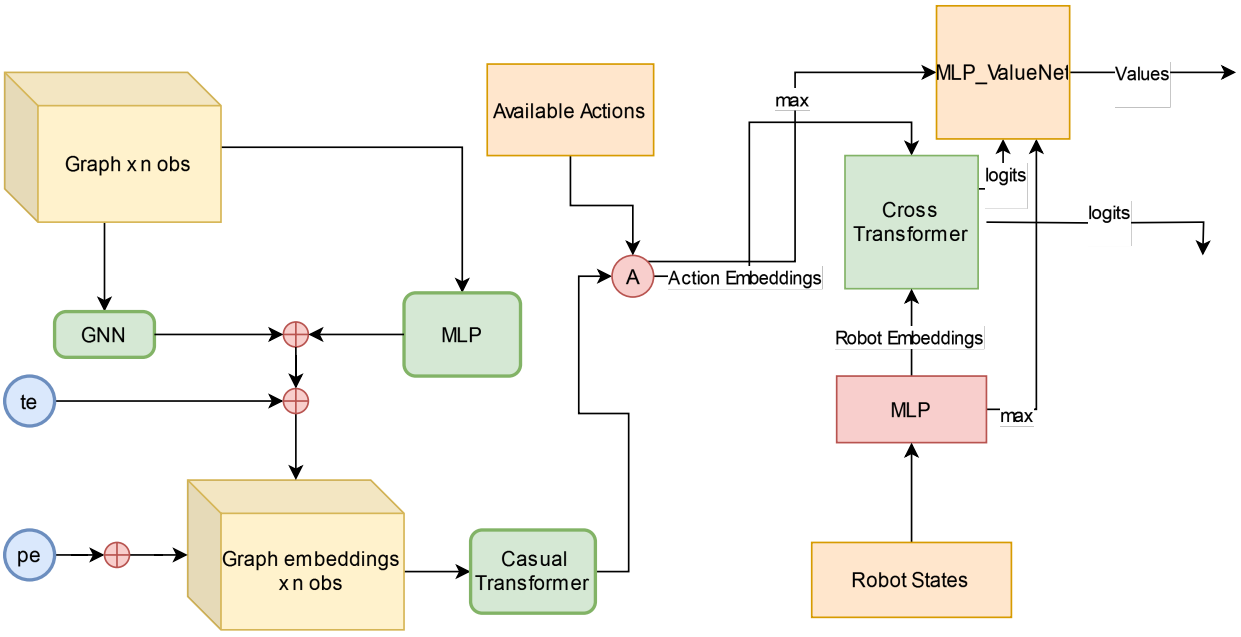


Figure 4.2: The proposed architecture, comprising a graph neural network, a causal Transformer and a cross attention Transformer operation A is selecting the embedding of the nodes available in action queue

field area. Number of centers are a drawn from uniform random distribution between 2 and 5. The models are trained with the Hyper parameters shown in Section 4.8.1. Expert policies are specifically trained under fixed scenarios with predefined infectivity, the Farm Layout and Number of Robots remain the same and three infectivity types

- Center Infectivity: The Center for laplacian is selected at (500, 500) Which is the dead center for the Graph and its variance is selected as half of the width.
- Right Infectivity: The center is selected at the top right conner and its variance is similar to center infectivity
- Left Infectivity: The Center is selected at top left conner and its variance is similar to Center Infectivity.

Moreover, the tanks are placed within the field, creating an out-of-distribution scenario for the robots, which is not practical but useful for testing. Further information is provided in Table 4.1. These trained policies are subsequently evaluated against those from expert scenarios, with some of the test scenarios presenting conditions that differ significantly from

Table 4.1: Expert Scenarios

Expert ID	0	1	2	3	4	5
Infectivity Type	Center	Left	Right	Center	Left	Right
No. Robots	2	2	2	5	3	5
No. Tanks	5	5	5	5	5	5
Head Land Height	267	152	231	290	216	275
Head Land Width	162	158	277	276	271	230

the training environments. This evaluation helps determine the generalizability of the models to new, unseen scenarios compared to the original training conditions. The necessity for this evaluation stems from the challenges presented when deploying current optimal algorithms in unknown infectivity scenarios, where the complexity of the task escalates with an increase in infected zones.

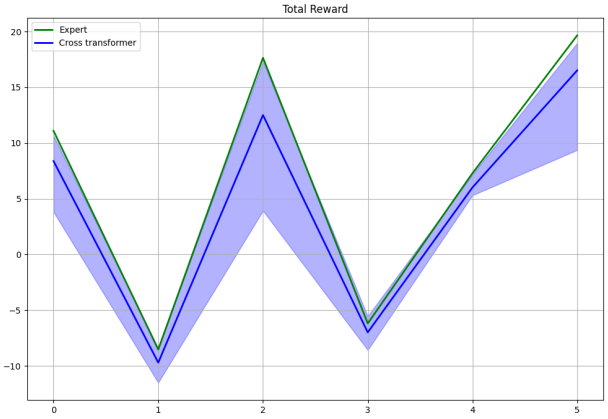
4.7 Results and Discussion

For the five expert scenarios in Table. 4.1, the expert for the respective scenario is tasked on solving it for 50 times, and the best case scenario is reported vs the range for cost and reward for the cross transformer on the same scenario is also reported. In the Table. 4.2, the percent deviation for the cross transformers trajectory from the expert trajectory is shown.This method of evaluation was chosen as there is no method for solving the problem of scouting and spraying at the same time.There are some methods such as[?] in which infectivity is considered to be known as prior but this method doesn't scale very well with increase in the number of infected plants.Hence, it is infeasible computationally to solve this problem over and over again as new information (Plant Infectivity) is acquired .

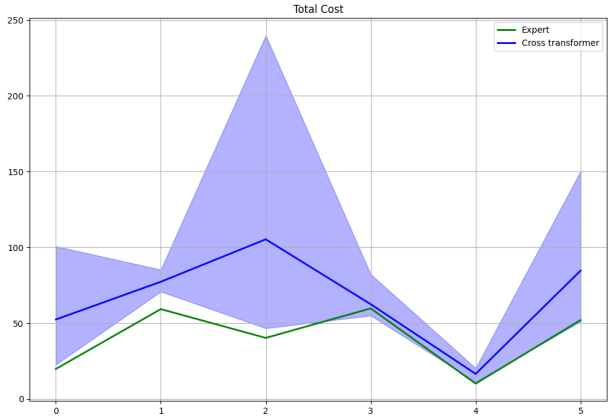
The best performance for the Cross Transformer is rather very close to that of the expert trajectories of scenarios trained in out of distribution. The Mean performance of the cross transformer is rather very close (around 50%) of the best trajectory as seen in the Table. 4.2. Hence, it can be stated that the Cross Transformer generalizes well for these distributions.

Table 4.2: Percentage of Expert Trajectories

scenario	reward max	reward mean	reward min	cost max	cost mean	cost min
0	0.037754	0.244901	0.656197	0.803628	0.622637	0.126405
1	0.011257	0.138405	0.349713	0.305002	0.233405	0.162988
2	0.020152	0.291543	0.778247	0.832206	0.617866	0.136522
3	0.104243	0.129594	0.386218	0.273849	0.043707	0.087060
4	0.015420	0.175536	0.273452	0.503669	0.389653	0.104178
5	0.034831	0.159712	0.523693	0.653769	0.385585	0.018238



(a) Total Reward of an experimental run using Task Transformer and Cross Transformer vs Expert



(b) Total Cost of an experimental run using Task Transformer and Cross Transformer vs Expert

Figure 4.3: Overall results of the experimental run using Task Transformer and Cross Transformer vs Expert

The success rate for all scenarios was 1 for both the expert and Cross Transformer. For all scenarios, the cost of the Cross Transformer is half that of the best trajectory. Experts, trained with a single distribution of infectivity, tend to understand the layout and optimize the trajectory better for the particular infectivity. In contrast, the Cross Transformer, as it is trained with the varying distributions of infectivity it doesn't overfit to a particular scenario but since the results are very close around 50% to the optimial it can be stated that the Cross Transformer Generalizes well. Given that the success rate for all scenarios is 1, it is evident that the POMDP formulation of the infectivity is successful.

4.8 Supplementary Material

4.8.1 Hyper Parameters

Hyperparameter	**Value**
Framework	Torch
Number of Rollout Workers	8
Number of GPUs	2
Number of CPUs per Worker	2
Number of GPUs per Worker	1/8
Final Layer Environment	Linear
Embedding Dimension	32
Environment Hidden Layers	{(32, 'Linear'), (32, 'Linear')}
Number of Observations	4
Temporal Blocks	1
Cross Blocks	1
Heads	4
Output Size	1
Graph Embedding	SAGENet
Input Block	TemporalTransformerBlock
Train Batch Size	2048
Lambda	0.975
SGD Minibatch Size	1024
VF Clip Parameter	50
Number of SGD Iterations	16
Use GAE	True
Learning Rate	0.00001225

Hyperparameter	**Value**
Learning Rate Schedule	{[0, 0.002], [50,000, 0.001], [100,000, 0.00015], [400,000, 0.0000125]}
Gamma	0.99
Use Critic	True
Gradient Clip	10.0
Entropy Coefficient	0.01
Entropy Coefficient Schedule	{[0, 0.05], [50,000, 0.01], [200,000, 0.001], [500,000, 0.001]}
Shuffle Sequences	True
VF Loss Coefficient	0.5

Chapter 5

NeuRowNet : End to End Row Detection for Robot Under The Canopy Row Navigation

5.1 abstract

Under the canopy robot navigation has seen a significant demand due increased interest in agricultural community for crop phenotyping, crop water stress assessment, liquid application etc. Traditional GPS fails under canopies, posing challenges. We introduce a novel approach using smooth polynomials for row and canopy identification, leveraging image space prediction and a customized loss function to enhance generalization. This method delineates navigation boundaries with polynomial functions in image space and extrapolates seamlessly into real-world navigation. Our approach achieves 1ms latency on edge devices and is evaluated using the Intersection over Union (IOU) metric, ensuring high accuracy for under-canopy navigation tasks

5.2 Introduction

The application of autonomous robots in high-throughput phenotyping is witnessing a significant surge in popularity, as recent literature attests[?] . These advanced robotic systems are revolutionizing the field by automating the intricate task of phenotyping with unparalleled precision and efficiency. While autonomous solutions in agriculture are not entirely new, as evidenced by historical insights[?] , the latest developments mark a remarkable evolution in this technology.

For nearly two decades, GPS-based autonomy has been extensively researched and implemented. Notably, studies by[?] and[?] have demonstrated the use of RTK and GPS-based systems to guide tractors and harvesters with high accuracy. However, these technologies face significant challenges under dense canopy cover due to GPS multi-pathing errors, which render them less effective in such environments. A prime example is the RoboBotanist project, which encountered navigation difficulties beneath the canopy when leaf coverage interfered with GPS signals[?] . This highlights the pressing need for innovative solutions specifically designed for under-canopy navigation, where traditional GPS-based systems fall short.

To address these challenges, vision-based strategies have been developed. For instance,[?] utilized object detection to identify plants, visual odometry with fast features, and various steps to obtain transformations from one frame to another. Additionally, the same research team[?] proposed a method using a Kalman filter and neural network to predict the robot's orientation relative to the crop row and its distance from the crop. This information, fused with data from other sensors using a moving horizon estimator, aimed to enhance position prediction. However, follow-up work[?] identified challenges such as algorithm failures in scenarios lacking crop row continuity or involving curved crop rows. Further advancements include an end-to-end detection network developed by[?] , which predicts line segments of rows with an FPS of 166. Although effective for straight rows, this method struggles with curved rows due to the absence of curvature in the prediction. Meanwhile,[?] employed YOLO and bounding box extraction to detect boundaries from images, estimating crop boundaries using

Figure 5.1: Samples from the train dataset: labeled samples (Blue), positive predictions with confidence greater than 0.5 (Green), and non-existent row predictions with confidence less than 0.5 (White).

ROI which requires post processing. Segmentation-based approaches proposed by[? ?] utilize full segmentation for under-canopy robot navigation. These methods, however, employ deconvolution layers, leading to performance issues compared to convolutional layers alone.

In the realm of lane detection for autonomous driving, methods such as those by[? ? ?] use polynomial regression-based loss to detect lanes. These techniques effectively identify lanes by employing multiple polynomials. Inspired by this work, we propose a novel method for row detection using polynomials. Unlike the lane detection networks, which predict multiple

polynomials for each lane, we predict two polynomials coefficients, namely:

$$[U\hat{(\lambda)}, P\hat{(\lambda)}]$$

If (u, v) are the normalized pixels of the said row, then $U\hat{(\lambda)} = u$ and $P\hat{(\lambda)} = \frac{1}{v}$. These polynomials are third-degree polynomials parameterized by $\lambda \in [0, 1]$. This allows the prediction of smooth polynomials in the image space. To overcome the issue of multiple rows, we predict 10 polynomial candidates and a classification probability for the occurrence of a row. Moreover, to eliminate the need for Non-Maximum Suppression (NMS), we train the network based on Hungarian matching loss. Assuming that the robot is always on the ground plane, given the height of the camera from the ground plane (after correcting for roll from the IMU), we can project back the predictions onto the ground plane from image space. This approach achieves a latency of 1ms on Jetson Orin's deep learning accelerator. We evaluate the accuracy of our method using the Intersection over Union (IoU) metric with trapezoidal integration between the predicted polynomial and the original polynomial. Our results indicate a 0.04 pixel error in U and V. However, upon further examination, it is observed that the polynomials fit very well for predicting the navigation boundaries and reveals a problem with the labels instead which can been seen in the Figure 5.1.

5.3 Methodology

An under-the-canopy robot, as shown in Figure 5.2, was meticulously designed to address the unique challenges posed by dense agricultural environments. This robot features a Jetson Orin[?] for onboard computation, which provides powerful processing capabilities necessary for real-time image analysis and decision making. The communication with actuators is facilitated through the Controller Area Network (CAN) protocol, ensuring reliable and efficient operation.

For image acquisition, the robot is equipped with a Luxonis OAK-D Pro[?] , a versatile depth camera known for its advanced vision processing capabilities. Despite its potential,

Figure 5.2: The under-the-canopy robot equipped with Jetson Orin and Luxonis OAK-D Pro for image acquisition.

the depth camera's performance under dense canopy conditions is limited, as the minimum distance required to acquire accurate depth images is 20 cm. This constraint is due to the camera's baseline and the complex light interactions within the canopy. Consequently, the RGB images from the camera were primarily used for training the neural network, bypassing the limitations of depth imaging in such environments.

The software integration for this system was achieved using ROS2[?] , a robust and flexible framework that supports the development of complex robotic systems. ROS2 facilitates seamless communication between various software components, enabling efficient data handling and processing.

The robot employs a skid-steer drive system, which is well-suited for navigating the uneven and often challenging terrain found beneath the canopy.

5.3.1 Dataset and Augmentations

To create accumulate a robust dataset for training the neural network, multiple ROS2 Bags were recorded while the robot was manually controlled around and underneath the canopy, capturing images as shown in Figure 5.1. The labeling process was conducted using the[?] platform. Labelers were tasked with labeling each row as a polyline, distinguishing between the navigable area and the canopy area. Third-degree polynomials $(U(\lambda), V(\lambda))$ were fitted to these points, parameterized by λ, which represents the normalized curvature of the (u, v), depicted as blue curves in the Figure 5.1. From the collected dataset, every 25th (approximately 0.8sec) frame was selected to avoid redundant data points at the same location, resulting in approximately 2000 images. This dataset was then split into training and testing sets in an 80-20 ratio. The training set was further divided into training and validation subsets, also in an 80-20 ratio. This means that the final training process used 64% of the total dataset, with the remaining 16% serving as the validation set.

Data augmentation techniques using Kornia[?] were applied to enhance the robustness of the model. These techniques included random rotations, flips, random cut out, rain, snow and color jittering, ensuring that the neural network could generalize well to various real-world scenarios. The augmented dataset provided diverse training examples, which is crucial for the model's performance in different environmental conditions.

The training of the models was conducted using the initial 80% of the dataset, ensuring that the neural network had ample data to learn from diverse scenarios. The split and augmentation strategies aimed to maximize the model's ability to accurately detect and navigate crop rows under varying conditions.

5.3.2 Architecture

The model architecture is depicted in Figure 5.3. The input images are resized to 512x512 pixels and then processed through a backbone network. Among the various backbones we experimented with, the EfficientNet backbone[?] yielded the best results, likely due to its use of Bi-FPNs (Bidirectional Feature Pyramid Networks). Additionally, other backbones such

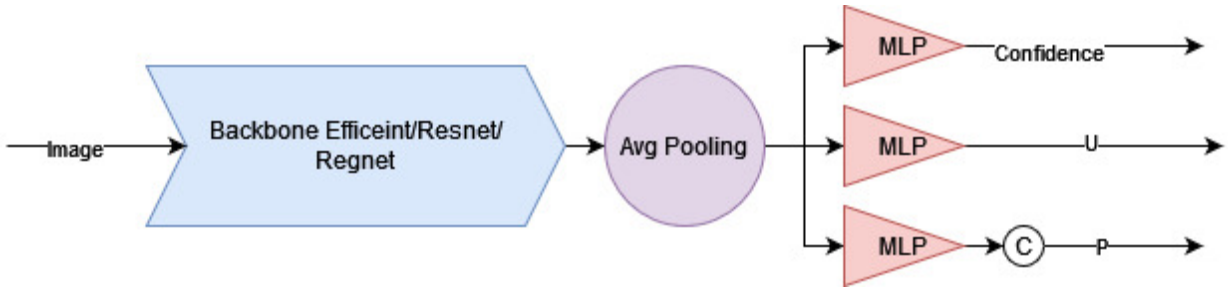


Figure 5.3: Model Architecture

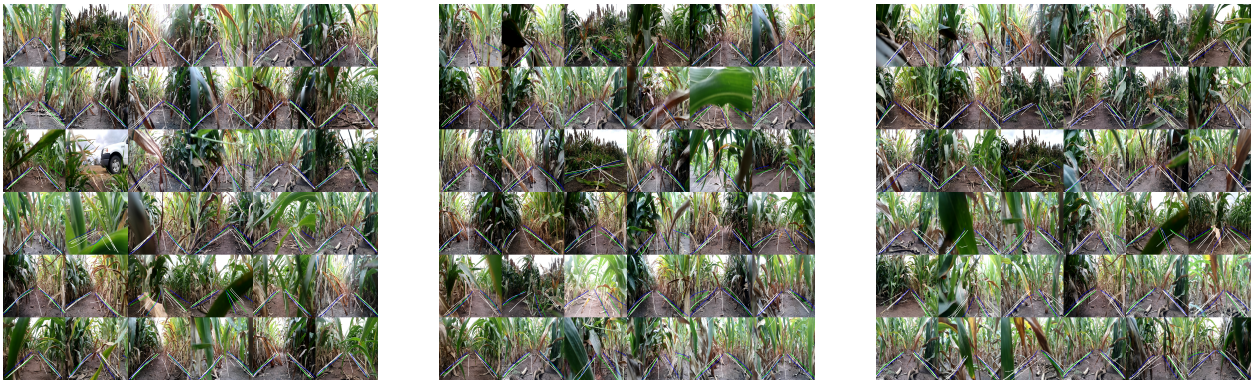
as ResNet-18[?] and RegNet[?] were also evaluated using the same configuration.

After passing through the backbone, a global average pooling (GAP) layer is applied. The resulting features are then fed into three separate Multi-Layer Perceptrons (MLPs), each comprising two layers. These MLPs are responsible for predicting the following:

- Confidence scores: This MLP uses a sigmoid activation function to output the confidence levels for the detected polynomials.
- Polynomial coefficients for U: This MLP, responsible for predicting the coefficients of 10 polynomials representing U, uses an Exponential Linear Unit (ELU) activation function in the first layer and a linear activation function in the final layer.
- Polynomial coefficients for P: Similar to the U MLP, this MLP predicts the coefficients for 10 polynomials representing P, also utilizing an ELU activation function in the first layer and a linear activation function in the final layer followed by an activation which statiesfies constraints on P which will be discussed below.

Predictions

Ideally, it is best to predict polynomials (x, y, z) directly in the camera frame, where z represents depth and y represents height from the image. However, if its is assumed that the camera is positioned on a plane and the height is known, the prediction can be simplified by focusing on (u, v) pixel coordinates and then using calibration parameters to project and predict $\left(\frac{x}{z}, \frac{y}{z}\right)$. Given the assumptions, we can express the coordinates as $(U(\lambda), \frac{1}{P(\lambda)})$, where



(a) EfficientNet (b) RegNet (c) ResNet

Figure 5.4: Samples from test set for the three backbones from Batch 1 (MSE Loss)

λ parameterizes the polynomial. Since y is a known constant height h, we can derive the coordinates in the camera frame as follows:

$$(x, y, z) = \left(\frac{U(\lambda)P(\lambda)}{h}, h, \frac{P(\lambda)}{h} \right)$$

This approach leverages the known height to project the (u, v) coordinates and predict the spatial (x, y, z) coordinates, simplifying the calculation while maintaining accuracy.

Constraints on $P(\lambda)$:

Lemma 1. *If $p(\lambda) = a\lambda^3 + b\lambda^2 + c\lambda + d$ is a real-valued polynomial function such that $0 < \frac{1}{p(\lambda)} < 1$ for all $0 \leq \lambda \leq 1$, then $b^2 > 3a^2 - 2ac$ and $d > 1$.*

Proof. We notice that $0 < \frac{1}{p(\lambda)} < 1$ if and only if $p(\lambda) - 1 > 0$ for all $0 \leq \lambda \leq 1$. Thus, the cubic polynomial $q(\lambda) = p(\lambda) - 1$ has no roots in the interval $[0, 1]$. If $q(\lambda) = a(\lambda - \alpha)(\lambda - \beta)(\lambda - \overline{\beta})$, where β is allowed to be complex and α is assumed to be real (since every cubic polynomial has at least one real root), then $|\alpha|, |\beta| > 1$. We have the following equations:

$$b = -a(\alpha + \beta + \overline{\beta})$$

$$c = a(\alpha\beta + \beta\overline{\beta} + \overline{\beta}\alpha)$$

$$d = -a\alpha\beta\overline{\beta}$$

This gives us the following:

$$\begin{aligned}\alpha^2 + \beta^2 + \overline{\beta}^2 &= (\alpha + \beta + \overline{\beta})^2 - 2(\alpha\beta + \beta\overline{\beta} + \overline{\beta}\alpha) \\ &= \frac{b^2}{a^2} + \frac{2c}{a} \\ &> 3\end{aligned}$$

The above inequality gives us that $b^2 > 3a^2 - 2ac$.

It is trivial to see that $d > 1$ as $p(0) > 1$.

Activation Function: the activation function from the predictions which satisfies the constraints is given by the following:

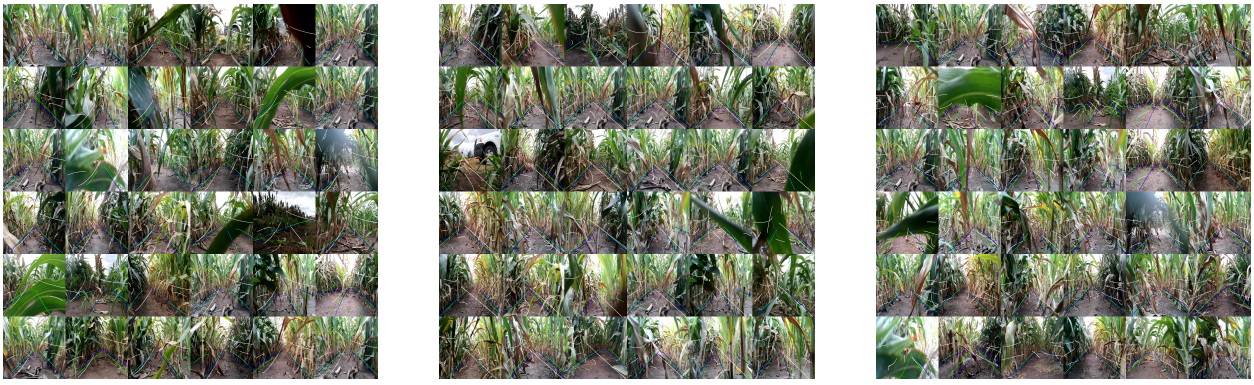
$$a = \hat{a}$$

$$b = 3\hat{a}^2 - 2\hat{a}\hat{c} + e^{\hat{b}}$$

$$c = \hat{c}$$

$$d = 1 + e^{\hat{d}}$$

Here, a, b, c, d are the polynomial coefficients and $\hat{a}, \hat{b}, \hat{c}, \hat{d}$ are the outputs from the linear layer. ■



(a) EfficientNet (b) RegNet (c) ResNet

Figure 5.5: Samples from test set for the three backbones from Batch 2 (PolyLoss)

5.3.3 Learning

The neural network is trained on the following loss function

$$\mathcal{L} = \mathcal{L}_{\mathcal{U}} + \mathcal{L}_{\mathcal{V}} + \mathcal{L}_{\mathcal{C}} \tag{5.1}$$

Where $\mathcal{L}_{\mathcal{C}}$ is the Binary Cross-Entropy (BCE) Loss between the predictions and the targets. The targets are selected by the Hungarian matcher, which assigns polynomials in a way that minimizes the total loss, inspired by the work of **?**. This approach removes the need for Non-Maximum Suppression (NMS).

The loss function for the polynomial predictions U defined as PolyLoss can be defined as:

$$\mathcal{L}_U = \int_0^1 (U(\lambda) - \hat{U}(\lambda))^2 \, d\lambda \tag{5.2}$$

where $\hat{U}(\lambda)$ is the predicted polynomial and $U(\lambda)$ is the target.

The loss function defined as CircleLoss for the polynomial predictions P can be defined as:

$$\mathcal{L}_V = \int_0^1 \left(V(\lambda)\hat{P}(\lambda) - 1 \right)^2 d\lambda \tag{5.3}$$

Where $V(\lambda)$ is the target and $\hat{P}(\lambda)$ is the polynomial This approach is necessary since there is no closed-form solution for polynomials of degree greater than five. Hence, it is not possible to find the solution analytically.

5.4 Experiments

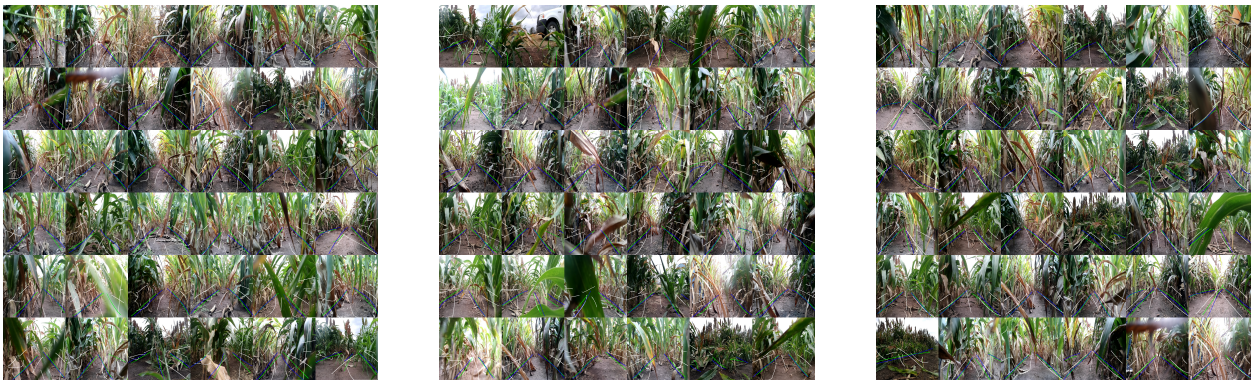
5.4.1 Objective

The objective of these experiments is to evaluate the accuracy of predictions (U, V) using the loss functions described in Equations 5.1, 5.3, and 5.2.

Three batches of models were trained with different configurations as shown in Table 5.1.

Table 5.1: Experimentation Batches

Batch No	Loss Function	Predictions
1	MSE between coefficients	U, V
2	PolyLoss 5.2, PolyLoss 5.2	U, V
3	PolyLoss 5.2, CircleLoss 5.3	U, P



(a) EfficientNet (b) RegNet (c) ResNet

Figure 5.6: Samples from test set for the three backbones from Batch 3 (PolyLoss, CircleLoss)

5.4.2 Experimental Setup

The training was conducted on a PC equipped with 2 RTX 4090 GPUs, an AMD Ryzen Threadripper PRO 5955WX 16-Core processor, and an Ubuntu 22.04 operating system, using Python 3.11. The models utilized backbones EfficientNet_lite0, ResNet18, and RegNetX.

5.4.3 Evaluation Metrics

The models were evaluated by using the following metrics.

- **Mean Poly Distance (MPD):** This is a metric similar to Intersection over Union (IoU). It calculates the trapezoidal integral distance between the predicted polynomials and the targets after matching the predictions and targets with their minimum distance. If there is an extra prediction or a missing target, its distance to 0 is calculated and this distance is minimized.

$$MPD = \sum_{\lambda=0}^{1} \left[(U(\lambda) - \hat{U}(\lambda))^2 + (V(\lambda) - \hat{V}(\lambda))^2 \right] \times 0.01 \tag{5.4}$$

Here, λ is incremented in steps of 0.01. In the case of Batch 3, $\hat{V}(\lambda) = \frac{1}{\hat{P}(\lambda)}$.

- **Accuracy Percent (AP):** This is the accuracy of the confidence MLP (multi-layer perceptron). Ideally, the MLP should output the polynomials with minimal distance with high confidence. If it predicts one with higher distance than the minimal, this is chosen as negative, and only those with minimal distance are selected as positive.

$$AP = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \tag{5.5}$$

5.5 Discussion

Table 5.2: Performance Metrics

Back Bone	Batch	MPD	AP
EfficientNet	1	0.3862	0.73496
EfficientNet	2	0.088	0.9835
EfficientNet	3	0.087	0.9758
RegNet	1	0.400	0.7678
RegNet	2	0.097	0.9783
RegNet	3	0.104	0.966
ResNet	1	0.769	0.7573
ResNet	2	0.1289	0.9657
ResNet	3	0.1077	0.9716

From Table 5.2, it can be understood that the accuracy of the models using MSE is significantly below par. This indicates that the models struggle to generalize well when using MSE as the loss function.

In Figure 5.4, it is evident that the predictions tend to memorize the polynomial coefficient predictions over the horizon. This memorization issue is exacerbated by human bias in labeling, which causes the predictions to perform poorly. Human labeling biases introduce inconsistencies that the model fails to correct, leading to suboptimal performance.

Moreover, the predictions from this batch fail when the canopy is almost parallel to the u-axis, as seen in the third row and third column of Figure 5.4b. In such cases, the predictions

are highly inaccurate. Additionally, when there are multiple rows in the image, this batch of models, regardless of the backbone, becomes confused and performs poorly.

EfficientNet outperforms all the other backbones for this particular task, which can be attributed to the BiFPN (Bidirectional Feature Pyramid Network) architecture. The BiFPN architecture enables the model to effectively communicate across different levels of the feature pyramid, thereby improving its ability to estimate rows even when they are partially occluded. This is demonstrated in Figure 5.5a, specifically in the last but one row, first image in the columns.

Furthermore, human bias or errors in labeling are evident in Figure 5.1. In the last but one training sample, the left row label inaccurately passes through the plants, whereas the model's predictions fit to the edges, accurately bisecting the navigable trajectory. This discrepancy can be attributed to the fact that human labels are often offset by a certain distance, either inward or outward. The neural network, aiming to minimize error, predicts the mean path through the navigable area, which results in better performance than relying on biased human labels.

To further understand these issues, additional ablation studies are necessary. These studies would help dissect the contributions of various components and reveal the underlying reasons for the observed performance trends. Specifically, they would shed light on how different loss functions, backbones, and labeling biases impact the model's ability to generalize and accurately predict navigable paths.

5.5.1 Conclusion

From the results of Batch 3, where Circle Loss is employed, we observe that the performance metrics, AP and MPD, are quite close, indicating the effectiveness of the loss scheme. Although there is a marginal 1 percent difference in accuracy between the batches across all models, the significant reduction in post-processing offered by this method when deployed is commendable. Furthermore, we deployed ResNet18 on a Jetson platform to evaluate its performance. Given that ResNet18 has the largest number of parameters (11.8M) com-

pared to EfficientNet (5M) and RegNet (3M), it was noteworthy that we observed only a 1-millisecond latency in the network. This result indicates that achieving a near 1ms latency for an end-to-end Row Detection Network is feasible. This highlights the potential for optimizing real-time applications where fast processing is crucial.

5.5.2 Limitations

One of the key drawbacks of this method is that not a lot of research has been done in polynomial detection, unlike the various other methods described in the introduction section. Hence, there might be chances of this not being very feasible. Moreover, further studies have to be done to ensure the predictions of the neural networks using the aforementioned loss functions are very generalizable. Even though for this study, data was collected from two different fields for test sets and training sets, it was collected in the same period on similar crops, hence raising the question of whether this method works on all types and stages of the crop lifecycle. We are currently collecting data during this planting period to verify this.

Chapter 6

Conclusion & Future Work

Some of the key contributions of this thesis are as follows:

1. **Detection and Estimation of Seed Spacing Using Neural Networks**: This study demonstrates the efficacy of neural networks in automating the detection and estimation of seed spacing, significantly reducing the labor-intensive task of manually walking across fields to gather seeding distance data. The automation system implemented in this research can achieve this task in just 67 seconds, compared to the 2 hours required for manual data collection.

   The results highlight that small neural networks, such as YoloR-P6 or YOLOR-CSPX, perform comparably to larger networks for the given dataset. This efficiency is attributed to the homogeneity in seed sizes, which reduces the necessity for higher levels in the Feature Pyramid Network (FPN). Consequently, smaller neural networks can be optimized for this specific application.

   Furthermore, the dataset utilized in this study was recorded with a GPS system lacking Real-Time Kinematic (RTK) corrections. Integrating RTK with neural networks deployed on edge processing systems, such as NVIDIA Jetson[?] , could provide real-time seed placement information. This advancement would facilitate easier evaluation of tractors' performance and enable the creation of real-time maps of the plant environment.

2. **Development and implementation of multi-robot-task allocation algorithm for pesticide spraying using neural networks :** The newly designed architecture, similar to Cross Transformer, effectively addresses the challenges associated with multi-robotic pesticide delivery tasks. The proposed model performs nearly as well as the expert model, with only a 0.5% performance difference in the best cases and a 50% difference when considering the mean performance. Additionally, the model demonstrates increased robustness to out-of-distribution data.

   Given that the success rate of the model matches that of the expert model in this scenario, it shows promise for deployment in real-world applications. Once the technology matures to the point where pest detection and pesticide spraying can be integrated into a single robot, this model could significantly reduce the time required for pesticide spraying.

   The performance of this neural network can be further enhanced through real-world deployment insights, particularly by analyzing infectivity as a function of location. This data can be used to re-train the model, thereby improving its overall performance.

   When compared to current state-of-the-art methods for multi-robot task allocation, framing the problem as a Partially Observable Markov Decision Process (POMDP) enables our approach to solve it within a significantly shorter time frame. This is particularly noteworthy given that previous methods were computationally infeasible for scenarios with high infectivity or a large number of rows.

3. **Row detection techniques for under-the-canopy navigation using neural networks :** In this study a new loss function was introduced to increase the efficiency of the row detection task. Usually this is done using a unet or yolo model to detect divable path. This study introduces a new loss function that is able to detect the path more accurately and efficiently. The results show that the new loss function is able to detect the path with 1ms latency on edge devices this is primarly due to the fact that the introduction of slack varibales namely λ and parameterizing the curves with the polynomials. There is no requiremetn for deconvolution layers, further more, there is

also no requirement of an FPN as the object sizes are pretty constant through out the row navigation task.

In summary, this thesis has explored the applications of neural networks in modern agriculture, focusing on several key areas to enhance automation and efficiency. One of the primary areas of investigation is the automation of seed spacing estimation. By employing advanced object detection methods and utilizing Intersection Over Union (IOU) filters, the research has demonstrated how neural networks can accurately predict the distance between seeds, reducing the time and labor required for manual data collection. This advancement has significant implications for precision agriculture, enabling farmers to optimize planting practices and improve crop yields.

Another significant contribution of this thesis is the development of a novel method for solving multi-robot pesticide spraying tasks. By leveraging the capabilities of neural networks, the proposed method enables coordinated and efficient pesticide application, reducing the processing time currently required by traditional methods. This approach highlights the potential for neural networks to enhance decision making in agriculture.

Additionally, this thesis presents an innovative technique for detecting row paths for under-the-canopy navigation. Utilizing a polynomial detection method, the research addresses the challenges of navigating through dense crop fields with minimal computational resources. This advancement is crucial for the deployment of robots in agricultural settings, ensuring reliable and efficient operation in complex environments.

The results of these studies collectively demonstrate the transformative potential of neural networks in revolutionizing traditional farming practices. By improving the accuracy, efficiency, and sustainability of agricultural operations, neural networks contribute to making farming more profitable and environmentally friendly. These advancements pave the way for further innovations in agricultural technology, promoting the adoption of intelligent systems that can adapt to the dynamic challenges of modern agriculture.

6.1 Future Work

The research presented in this thesis opens up several avenues for future exploration and development. Some potential areas for further investigation include:

- **Real-Time Seed Spacing Detection:** The seed spacing method can be enhanced to work in real-time by developing a more efficient neural network and deploying it on an edge device. This improvement will not only reduce the time taken to detect the seeds but also provide real-time information to the farmers. Additionally, the map generated by the system can be utilized to create the graph required for the multi-robot task allocation algorithm, thereby streamlining the integration of different agricultural processes.

- **Optimized Multi-Robot Task Allocation:** The multi-robot task allocation algorithm can be further optimized by incorporating real-time data on pest infestation levels. By integrating pest detection capabilities into the model, the algorithm can dynamically adjust the allocation of tasks based on the current infestation status. This real-time adaptability will improve the overall efficiency and effectiveness of pesticide spraying operations, ensuring targeted and precise application of pesticides.

- **Robot Localization and Navigation:** The map generated by the seed spacing detection algorithm can be combined with the row detection algorithm to localize the robot with respect to the map using Adaptive Monte Carlo Localization (AMCL) or other localization algorithms. This integration will enable the robot to navigate through the field with minimal human intervention. To achieve this, the behaviors of the robot can be defined using a behavior tree, allowing the robot to autonomously navigate through the field. This approach can be seamlessly integrated with the current navigation stack of the robot, enhancing its autonomous capabilities and operational efficiency.