Yield prediction using Spatial and Temporal Deep Learning Algorithms and Data Fusion

by

Bhavesh Bisht

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
Master degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

The world's population is expected to grow to 9.6 billion by 2050 [26]. This exponential growth imposes a significant challenge on food security making the development of efficient crop production a growing concern. The traditional methods of analyzing soil and crop yield rely on manual field surveys and the use of expensive instruments. This process is not only time-consuming but also requires a team of specialists making this method of prediction expensive. Prediction of yield is an integral part of smart farming as it enables farmers to make timely informed decisions and maximize productivity while minimizing waste. Traditional statistical approaches fall short in optimizing yield prediction due to the multitude of diverse variables that influence crop production. Additionally, the interactions between these variables are non-linear which these methods fail to capture. Recent approaches in machine learning and data-driven models are better suited for handling the complexity and variability of crop yield prediction. Maize, also known as corn, is a staple crop in many countries and is used in a variety of food products, including bread, cereal, and animal feed. In 2021-2022, the total production of corn was around 1.2 billion tonnes [9] superseding that of wheat or rice, making it an essential element of food production. With the advent of remote sensing, Unmanned aerial vehicles or UAVs are widely used to capture high-quality field images making it possible to capture minute details for better analysis of the crops. By combining spatial features, such as topography and soil type, with crop growth information, it is possible to develop a robust and accurate system for predicting crop yield. Convolutional Neural Networks (CNNs) are a type of deep neural network that has shown remarkable success in computer vision tasks, achieving state-of-the-art performance. Their ability to automatically extract features and patterns from data sets makes them highly effective in analyzing complex and high-dimensional datasets, such as drone imagery. In this research, we aim to build an effective crop yield predictor using data fusion and deep learning. We propose several Deep CNN architectures that can accurately predict corn yield before the end of the harvesting season which can aid farmers by providing them with valuable information about potential harvest outcomes, enabling them to make informed decisions regarding resource allocation. UAVs equipped with RGB (Red Green Blue) and multi-spectral cameras were scheduled to capture high-resolution images for the entire growth period of 2021 of 3 fields located in Ottawa, Ontario, where primarily corn was grown. Whereas, the ground yield data was acquired at the time of harvesting using a yield monitoring device mounted on the harvester. Several data processing techniques were employed to remove erroneous measurements and the processed data was fed to different CNN architectures, and several analyses were done on the models to highlight the best techniques/methods that lead to the most optimal performance. The final best-performing model was a 3-dimensional CNN model that can predict yield utilizing the images from the Early(June) and Mid(July) growing stages with a Mean Absolute Percentage error of 15.18% and a Root Mean Squared Error of 17.63 (Bushels Per Acre). The model trained on data from Field 1 demonstrated an average Correlation Coefficient of 0.57 between the True and Predicted yield values from Field 2 and Field 3. This research provides a direction for developing an end-to-end yield prediction model. Additionally, by leveraging the results from the experiments presented in this research, image acquisition, and computation costs can be brought down.

Acknowledgements

I would like to express my sincere gratitude to everyone who has supported me throughout my thesis journey. Firstly, I am deeply grateful to my supervisor Dr. Iluju Kiringa and my thesis co-supervisor Dr. Tet Yeap for their guidance and constructive feedback. I would also like to thank Patrick Killeen for his invaluable assistance and support from the very beginning, which has significantly contributed to the progress and success of this thesis. Without their invaluable support, this thesis would not have been possible.

I extend my gratitude to my family for their unwavering encouragement and belief in me. Their constant support has kept me motivated during difficult times. At last, I want to thank the University of Ottawa, for providing me with the necessary resources and facilities to conduct my research.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Predicting yield has become an integral part of food security and decision-making. It can improve crop agronomic management which leads to sustainable systems. An accurate crop yield prediction model can help farmers decide on what crops to produce and when to plant them [166]. Moreover, it helps to understand the correlation between environmental factors (i.e., hydrological, meteorological, soil factors, etc.) with the final yield values that will maximize crop production. Traditional methods of predicting these values often show low performance as they are unable to capture non-linear spatial and temporal dependencies.

The development of sensor technology in recent years has significantly promoted the application of Unmanned Aerial Vehicles (UAVs) for data acquisition due to their improved spatial, spectral, and temporal resolution compared to airborne and satellite platforms [113]. Deep learning, on the other hand, has emerged as a powerful tool for analyzing the large amounts of data collected by UAVs and other platforms equipped with advanced sensor technology, enabling more accurate and efficient extraction of valuable information for various remote sensing applications. It is a field of machine learning that simulates the neural network of the human brain and is efficient in capturing intricate non-linear relations with high accuracy.

The current research proposes an end-to-end life cycle of yield prediction right from data capturing, post-processing, data fusion, model development, optimization, and evaluation. Several analyses, such as correlation analysis, were conducted to examine the relationship between various phenological stages of crop growth and the final yield that can be used to determine the optimal days for capturing UAV images, leading to better performance and cost savings associated with UAV-based data collection. The study was carried out on six demonstration fields in Ottawa, Canada, where corn is produced. These fields are located in the same geographic area and therefore exhibit similar characteristics. We experimented with several non-parametric Deep Convolutional Neural Network Models that can efficiently capture spatial and temporal relationships in the field data to predict the final yield. The baseline was created using a two-dimensional Convolutional Network model where the training data was done on RGB imagery (Red Green and Blue) from a single phenological stage. Several regression evaluation metrics including Root Mean Squared Error, Correlation Coefficient, and Mean Absolute Percentage Error were used

in conjunction to evaluate the model performance on the unseen data. The results show that the combination of remote sensing and deep learning shows promising results in developing a robust yield prediction model that can help farmers to optimize their farming practices, improve crop productivity, and increase profitability, ultimately contributing to more sustainable and efficient agricultural practices.

1.1 Motivation

With a production volume of around 1.2 billion metric tons, corn has become the global leader among the most important grains produced worldwide [148]. It is Canada's third most valuable crop [157]. Therefore, ensuring a sufficient yield is vital for food security, emphasizing the need to understand and increase agricultural productivity. The quantity can be deciphered through several analyses, which can be used to identify strengths and weaknesses in farming practices and make timely decisions including Seed Selection, Moisture Control, Pest Management, Irrigation Schedule, Fertilizer usage, etc.

Remote sensing is a term used for identifying and collecting information without having physical contact with the object of study; more specifically, it refers to information gathered by devices that detect electromagnetic radiation, visible light, infrared light, and near-infrared light. [71] Remote sensing provides three major resolutions in farms, namely Spatial resolution, Spectral resolution, and Temporal resolution. Where Spatial resolution provides the overview imagery of the field using traits like size, distance, height, width, weather, and pests related to the crops that can be identified. Whereas, Spectral resolution helps to understand health, moisture, nitrogen level, etc., based on the different spectral wavelengths reflected. The Temporal resolution is related to time and details on the crop's phenological growth stages. The existing research relied mainly on satellite [145] or low-quality drone imagery for predicting the yield. This study utilized high-quality images, with a resolution ranging from 12.0 to 33.2 MP(Mega Pixels) and a Spatial resolution of 0.7 to 1.3 cm, which were stitched together to create 8K orthomosaics. Orthomosaics or mosaics are high-resolution geo-referenced aerial images created by stitching together multiple overlapping drone images. The high-resolution images enabled the extraction of fine-grained features from the field, ultimately leading to improved prediction performance. Data collection for this research was carried out by IndroRobotics mainly using dual-payload DJI M210 drones equipped with both RGB and multi-spectral cameras to capture images throughout the entire growing stages (Early Mid and Later). Incorporating temporal resolution into the analysis helped to gain a deeper understanding of the relationship between final yield values and various stages of crop growth.

The accurate prediction of crop yields is a critical challenge that the agricultural industry faces. Yield prediction can aid farmers in decision-making related to crop management, resource allocation, and risk management, ultimately leading to more sustainable and efficient agricultural practices. While traditional methods of yield prediction have been utilized in the past, recent advancements in deep learning have shown promise in improving the accuracy of yield prediction models [166]. The traditional Linear regression models are unable to capture the spatial and temporal nonlinear relationships because the

interaction between the different explanatory variables depends on spatial structure in a nonlinear fashion. Machine and Deep learning have gained a lot of attention recently, as they are able to produce high-quality results in various fields including medical analysis, self-driving cars, price prediction, etc [152]. While machine learning models such as Random Forest [70] have been successful in accurately predicting yield during the early stages of crop growth by effectively learning spatial features, they fall short with a huge volume of data and in capturing temporal relationships that are critical to understanding crop growth over time. The ability of Convolutional Neural Networks(CNNs) to capture spatial and temporal relationships between different input variables, their capacity to automatically learn and extract features from images, and their ability to handle huge volumes of data make them a highly suitable choice for this task.

The volume of data used in this research is substantial, as we have utilized 8K orthomosaics consisting of high-resolution images captured over multiple phenological stages of crop growth for 3 fields at AreaX.0. Such a large volume of data requires significant computational power for efficient processing and analysis. Fortunately, with the advancements in graphics processing units (GPUs), it is now possible to efficiently train complex deep learning models on large datasets. GPUs expedite computations and facilitate efficient training of large deep-learning models, leading to faster experimentation and enhanced performance. They offer increased memory capacity, and parallel processing capabilities, and have broad support from deep learning frameworks and libraries like Pytorch. Models like 3D CNNs that are well-suited to handle spatiotemporal data used in this research were trained on GPUs.

1.2 Problem - Challenge

Crop forecasting holds great importance in planning the market, farming practices, food security, harvest, and nutrient management. However, an infinite number of heterogeneous factors that influence the yield measurement like weather, moisture, soil quality, genetics of seed, pest, and crop health make it a complicated task. Therefore, there is a need for a robust solution that can extract these highly nonlinear combinations. Another challenge is the limited spatial and temporal resolution which directly affects the accuracy of the predicted yield. Traditional approaches rely on low-resolution satellite imagery that fails to capture the fine-grained features that are required to assess crop growth and health. Additionally, these method provides a snapshot of the field at a particular point in time and fails to capture the changes over time.

Unmanned aerial vehicle (UAV) technology bridges the gap among space-borne, airborne, and ground-based remote sensing data.[179] It has been widely used to capture high-quality aerial data of the fields that can be used for further analysis. However, handling and operating this technology requires expert specialized skills like setting up the Ground Control Points (GCPs) that reduce the margin of error in mapping large area maps. Furthermore, capturing high-quality images of large farms demands significant resources due to challenges such as limited power supply and adverse weather conditions that obstruct drone flights. Mounting high-quality multi-spectral RGB cameras and other

sensors on high-end drones combined with experts makes the data generation process an expensive task. Ground yield data acquisition is often made by mounting a yield monitoring device on the harvester. Achieving consistent and accurate grain yield measurements is a challenging process, especially over the large dynamic range of flow rates that stem from spatial yield variability [37]. This gap can be due to management-related processes or measurement errors due to the yield monitoring process itself. Therefore, an optimized plan and post-processing pipeline incorporating multiple techniques are required to eliminate disruptions from the yield data, which can then be used as a reliable source of ground truth. After data capture and processing, analyzing the resulting images requires significant computational power. Visualization and processing typically rely on software such as Pix4D for generating orthomosaics and QGIS. In our research, a single orthomosaic of the field was an average of around 2 GB, making it challenging to load the entire image in the memory. In addition, the modeling process itself can be computationally expensive, especially for more complex models that require iterative optimization procedures. This can lead to long processing times, which can be a hindrance in time-sensitive agricultural decision-making.

To address these challenges a multidisciplinary approach is required that incorporates expertise in agronomy, computer science, and other fields and statistics. An end-to-end pipeline that consists of several steps and techniques to develop an accurate and practical yield prediction model addressing all these challenges is proposed in this research that can help farmers and policymakers make informed decisions and address the challenges of food security and agricultural sustainability.

1.3 Solution Outline

This research proposes an end-to-end yield prediction workflow comprising data acquisition, raw data processing, data fusion, optimized modeling, evaluation, experimentations, and error analysis. There were two types of data sources available: (a) Orthomosaics obtained from UAV image data captured throughout the 2021 growing season, and (b) Shapefiles (geospatial vector data format) containing yield measurements obtained at the harvesting stage through a yield monitoring device mounted to the harvesting machine. The first step was to preprocess the raw data, where different techniques were employed in the form of a pipeline as shown in Fig. 3.26 . During the initial stages of this data processing workflow, various statistical techniques such as the Z-score, and backward and forward threshold-based methods were used to clean the yield data obtained from the harvester [110]. However, later in the pipeline, more advanced machine learning techniques for outlier detection, such as Isolation Forest and Elliptic Envelope Algorithm, were utilized. Similarly, several methods were opted for processing the big raw field orthomosaics. As the average field orthomosaic was approximately 2GB in size, they were segmented into smaller 4-5 slices per field by pre-defining the boundaries using QGIS software [20]. The smaller slices could be efficiently loaded into the memory. These slices were further cropped into 9 X 9 m images, where each tile represents a particular part of the field as shown in Fig. 3.36. The image tiles and processed yield were fused together based on the spatial

location on the field, such that each tile has a single yield measurement. The image tile in this pair serves as the training data (X) and yield data represent the label (Y). The test set was created using the image tiles and yield measurements from other fields that were not a part of the training set to evaluate model performance on the unseen data. Proposed deep learning networks are a combination of convolution and fully connected layers with varying kernel/filter sizes that can efficiently capture the spatial and temporal relationship to predict yield. The models were evaluated and compared based on popular regression metrics including Root Mean Squared Error, Mean Absolute Percentage Error, and Correlation Coefficient. Several experiments were performed to identify strategies that could improve the prediction performance such as Loss comparison, Regularization, Augmentation, Image resolution, etc. Moreover, the results from these analyses could reduce data acquisition and computation costs.

In the end, model hyperparameters were tweaked to further optimize the performance. The best-performing was a 3D CNN model that utilized images from the Early and Mid-growing stages from Field 1 for training to predict yield with a MAPE of 15.18%, an RMSE of 17.63 (bu/ac), and an average Correlation value of 0.57 on the test set created using Field 2 and Field 3.

1.4 Methodology

This work is an experimental thesis, where we have designed and implemented controlled experiments to investigate and develop an end-to-end yield prediction model development workflow. It involves collecting real data from corn fields based in Ottawa, Ontario, utilizing deep learning models to develop a prediction model and finally experimenting on the model to identify sophistical techniques that can improve prediction performance and reduce costs. The developed model could predict the end of the season in advance which can help farmers make timely decisions to increase productivity. The first step was to do exhaustive research in the field of remote sensing, yield prediction, and deep learning to discover challenges, recent trends, and technology to design an optimized solution. The second step was to gather the data, which was done by professionals from third-party contractors specialized in their discipline. The field data was gathered with the help of InDro Robotics [10], which is a Canadian UAV research and development company. Throughout the growing season of 2021, drones were deployed to capture RGB and multi-spectral imagery of the fields from the air thus, capturing both the spatial and temporal dimensions. The yield data was gathered during the harvesting season by mounting Trimble's (Trimble is a US-based company that specializes in providing advanced technology solutions for various industries including agriculture) yield monitoring device on the harvester that provided continuous measurements over different parts of the fields in the form of shapefiles.

The consecutive steps involved implementing data preprocessing techniques, generating training data, building and experimenting with various deep learning architectures, and finally, emphasizing the key findings derived from experimentation to illustrate how these insights can bring benefits to farmers by reducing costs and improving overall efficiency.

The best-performing solutions were outlined and compared against each other based on several evaluation metrics.

1.5 Contributions and Limitations

1.5.1 Contributions

1. The current work gives a direction to build an end-to-end yield prediction model using deep learning and data fusion from data acquisition to model development based on high-quality real-world data acquired from AreaX.0
2. The proposed model predicts the end of the season yield utilizing the images from the Ealy and Mid-growing seasons, therefore it gives farmers early insights for proactive decision-making and optimizing resource allocation.
3. Contrary to current commercial predictors, our model makes extensive use of spatial and temporal dimensions of the field where the phenological stage was used as the temporal dimension to capture changes over time.

1.5.2 Limitations

a) The images were cropped into tiles of size 9 X 9 m, where the dimensions were calculated based on the harvester's speed (2.3 m/s), and a GPS error of 2m. The assumption was that the spatial correlation didn't exist outside of the 9 X 9 m tile.

b) Assigning a single yield value per tile may not be accurate as the tile can cover both empty and crop areas. Hence, it would be more reasonable to assign multiple values or a single value per pixel.

c) During this analysis, only drone imagery was used, and other factors such as weather, canopy area, and leaf area index were not considered. However, it is important to note that these factors can also impact the final yield.

d) While drone data provides a top view of the crops, analyzing the crops from a side view can provide additional insights into their growth such as canopy length, which can potentially enhance the accuracy of yield prediction.

e) Given the 2-meter GPS (Global Positioning System) error margin of drone imagery, a buffer zone of 2 meters was added around each tile to mitigate the error's impact on adjacent tiles. This buffer zone effectively decreases the error's influence but may increase the correlation between adjacent tiles.

f) Multi-spectral imagery was not taken into account in the analysis. Nonetheless, this type of imagery, which captures multiple bands of light beyond the visible spectrum, can offer valuable information about crop health and growth patterns, including changes in vegetation density, photosynthetic activity, and plant water stress.

g) The training and testing data in the study was based on a single image taken during the midpoint of each growing stage. However, it is important to note that multiple drone flights were conducted during each stage, which could provide additional data for future analysis.

h) The majority of data points come from medium or high-yield areas of the field. Incorporating more data points from the empty ground areas, where crops are missing, could have enhanced the model's performance by providing additional variability in the data.

i) The focus of this research was on utilizing CNNs to capture the spatial-temporal relationship. However, there are also other hybrid models, such as CNN-LSTM, that could have been explored for capturing these relationships.

1.6 Outline of Thesis

The thesis is structured as follows; Chapter 2 provides extensive background and a literature review in the area of yield prediction, remote sensing, and deep learning. It also outlines the current challenges and drawbacks of the existing research; Chapter 3 proposes an end-to-end pipeline that begins with data acquisition, data processing techniques, interpolation strategies, data fusion, and deep learning convolutional models that are effective in capturing the spatial-temporal relationship, model optimization techniques, and performance evaluation metrics; Chapter 4 shows the experimentation results, error analysis and lays out the conclusion of these experiments; Chapter 5 summarizes the conclusion of this study and the direct impact it can have on improving crop yield. In addition, it reviews the next steps to enhance our knowledge in this domain.

Chapter 2

Background and Related Work

2.1 Precision Agriculture and Smart Farming

Precision Agriculture and Smart Farming refer to the use of advanced technology in crop cultivation to enhance productivity and improve farming practices. The technology encompasses a variety of forms such as GPS, sensors installed in fields to measure moisture levels, and advanced machinery. This technique of growing crops ensures that the plants receive exactly what is required for their optimum health by considering features like weather, soil properties, etc. [27] [147]

A literature review of [39] [160] revealed that the first phase includes collecting the data which is done both in real-time and in schedules. The data extraction is performed using specialized equipment and sensors. Some of the most common data sources are moisture, precipitation, canopy width, [45] canopy area, leaf area index, field images, weather, soil nitrous oxide, fertilizer content, etc. The raw data that has been captured is transmitted to downstream algorithms and techniques that undertake data cleaning and processing as the second phase. The objective of this phase is to remove outliers, redundant data and generate meaningful data that can be consumed by the analytics software. [110] and [47] propose several data processing techniques that are also used in this research. Afterward, statistical methods such as predictive analytics and machine learning algorithms are employed to examine the data, thereby producing valuable insights to facilitate informed agricultural decision-making. For instance, [31] proposes the prediction of apple disease in the apple orchards of Kashmir Valley using data analytics and machine learning in the IoT system. Furthermore, [180] proposes a low-cost method to perform farmland topological analytics to achieve immediate improvement in the operation and path planning of large agricultural machines. The final generated insights are a quantized version of the best farming practices, that optimize resources and make sure that the soil and crop have the right amount of nutrition and health that reduces costs, and increases the overall productivity of the field which is the main aim of smart farming. Some of the other use cases of precision farming include agricultural mapping and field scouting, soil sampling and analysis, weather monitoring, labor management, and equipment management.

2.2 Internet Of Things

The Internet of Things (IoT) [11] is a network of connected physical objects or devices over the Internet, where they can communicate by sharing data. The main objective of the IoT ecosystem is to create a network of interconnected devices that are capable of automating tasks, optimizing processes, and providing insights into various aspects. The components of IoT are the connected hardware components as shown in the fig. 2.1 which are usually the physical devices, software applications that are needed to manage, and a network infrastructure that can provide a communication channel. The networking channel has moved to more advanced wireless protocols, such as Bluetooth, Wi-Fi, and Cellular networks. The software component of it collates the data, processes it, and analyzes it to draw insights and make informed decisions.

Due to the high volume of data collected and transmitted, there are concerns about the security network in place to preserve privacy and protection of sensitive information. However, data privacy and security are only one of several setbacks to IoT. Other challenges include data overload and interoperability issues. [58] [62] outlined five layers in the paradigm that work together including Abstraction, Communication, Network, Storage, Processing, and Application Layer.

2.2.1 Abstraction Layer

This layer resides between the hardware devices and the application layer. It provides a simplified interface for the end application developers to interact with the hardware devices. The presence of this layer makes it possible for the development of applications that work independently of the underlying hardware and networking infrastructure [62]. The utilization of provided APIs enables developers to seamlessly deploy their applications across various IoT devices. Device drivers, protocol translators, data management, and security services are some of the components of this layer. [128] In general, the IoT abstraction layer plays a crucial role in the development of IoT solutions by simplifying the process and facilitating the creation of scalable and interoperable IoT applications.

2.2.2 Application Layer

The application layer resides at the top of the IoT system and is used by the end users to interact with the IoT Platform. [168] It includes a user interface and services on top of the IoT network layer that streamlines user interaction. The developers interact with the application layer and develop tools or perform analysis by utilizing the IoT data that is offered at this layer. Automation systems, wearable health trackers, industrial automation systems, and smart city infrastructure are some of the IoT end applications. There are various communication protocols that are described under the Network Layer that are responsible for seamless communication with the lower layers. There are also security mechanisms that protect sensitive data and ensure user privacy. These security mechanisms include Authentication, Encryption, and Access Control.

2.2.3 Storage Layer

The storage layer is responsible for storing and managing the vast amount of data that is generated by IoT physical devices. This layer plays a critical role in ensuring that IoT data is collected, processed, and stored securely and efficiently. [112] There are the following sub-layers that are part of this layer:

1. Data Ingestion Layer: This layer is responsible for collecting the data from the IoT devices and storing it at the appropriate location. Data validation and filtering ensure that only the relevant data is stored.

2. Data storage layer: [168] This layer includes the actual database such as Relational database, No-SQL, and Distributed file system. It ensures that the incoming data is stored in a secure and scalable manner.

3. Data processing layer: This component is in charge of conducting real-time analytics on inbound IoT data, allowing for quick decision-making and action. This component can employ a variety of data processing techniques, such as stream processing and complicated event processing.

4. Data retrieval layer: This component offers a query and retrieval interface for the saved IoT data. This layer can make use of a variety of technologies, such as APIs, query languages, and visual analytics tools.

5. Data security layer: [128] This layer guarantees the security, accuracy, and accessibility of IoT data. It includes a number of security measures, such as encryption, access control, and monitoring.

2.2.4 Networking Layer

The networking layer is accountable for managing and handling communication protocols that ensure secure and efficient communication between IoT devices and systems. [121] It acts as a communication mediator between IoT devices, sensor nodes, gateways, and cloud platforms. Some of the protocols used in the network layer are:

1. MQTT (Message Queuing Telemetry Transport): This protocol [15] uses the publish-subscribe model for efficient transmission of data between IoT devices.

2. CoAP (Constrained Application Protocol): This [5] uses the UDP (User Datagram Protocol) and is used for machine-to-machine communication.

3. Zigbee: [60] This is a wireless mesh networking protocol commonly used for home automation and IoT applications

4. Wi-fi: It is one of the most common networking protocols that provides high-speed connectivity to IoT devices.

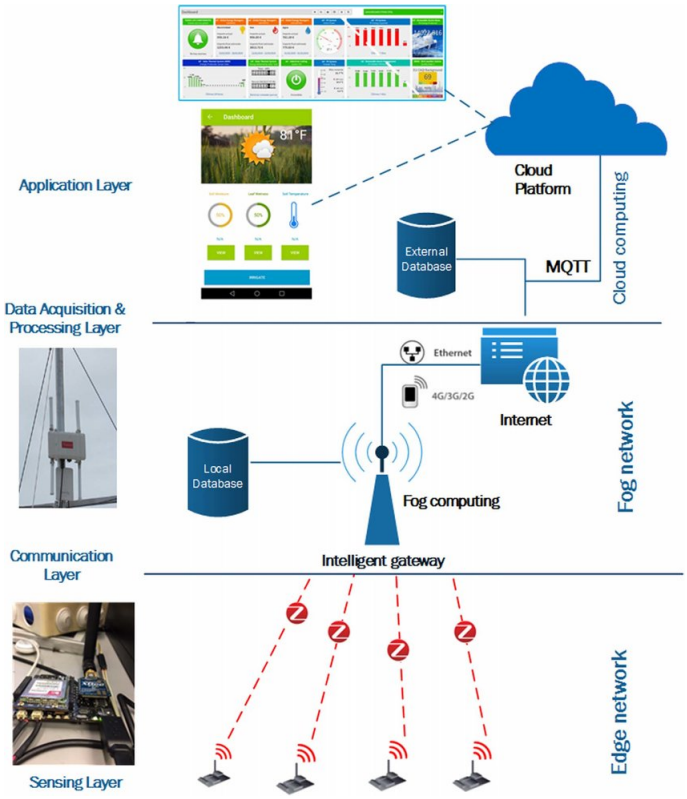

Figure 2.1: [117] The IoT architecture consists of several layers, where the data is primarily collected using sensors, and the captured data is analyzed through downstream applications in the application layer.

2.3 Smart Farming Solutions

2.3.1 FarmBeats

FarmBeats [167], developed by Microsoft, is an end-to-end platform that utilizes advanced technologies like Remote Sensing, Artificial Intelligence, and Machine Learning that enable precision agriculture. Some of the objectives of FarmBeats include reduced environmental impact, improved crop yields, lower cost, improved decision-making, and data-driven insights. It enables farmers to link various devices, such as soil moisture monitors, weather systems, and drones, in order to gather data on various parts of their crops and agricultural operations. The integration of Azure Cloud Platform in the suite offers vast computing resources that are utilized for the storage and analysis of data collected from physical devices. One of the key features of FarmBeats is that it uses a Gateway based design. The Gateway has two major objectives: (a) It performs local computation to convert into summaries that can be loaded to the cloud hence saving bandwidth and time for uploading the entire raw data (b) It consists of operations that can handle periods of network outage hence increasing the availability to the farmers. The FarmBeats contains the following main components [167] as shown in Fig. 2.2

1. IoT Base Station: The IoT base station gathers data from sensors and devices and safely sends it to the FarmBeats cloud platform. The data is then processed and evaluated using machine learning algorithms to provide insights and suggestions for maximizing agricultural harvests, decreasing water and fertilizer usage, and increasing farm efficiency overall. It serves as a gateway for data transmission between the sensors and the cloud. It uses a combination of long-range wireless communication technologies such as LoRaWAN (Long Range Wide Area Network), Zigbee, and Wi-Fi to connect sensors and devices on the farm to the cloud.

2. Sensors and Drones: They are critical components of FarmBeats. They are utilized to gather data from various aspects of farms, such as soil moisture, temperature, humidity, and crop health. Sensors are low-power devices that can be deployed across a farm to monitor different environmental conditions. They collect data such as soil moisture, temperature, humidity, and weather conditions, and transmit it wirelessly to an IoT base station or directly to the cloud. Drones, on the other hand, can be used to capture high-resolution images of a farm, which can be used for crop monitoring and mapping, soil analysis, irrigation management, livestock monitoring, and crop spraying.

3. Duty Cycling: Duty cycling is an important technique for conserving battery life and reducing energy consumption in wireless sensor networks (WSNs). Duty cycling involves two parameters: the duty cycle ratio and the duty cycle interval. The duty cycle ratio refers to the percentage of time that a device is in the active state, while the duty cycle interval refers to the duration of each active state. By adjusting these parameters, farmers can optimize the trade-off between energy consumption and data collection frequency. By using duty cycling, FarmBeats can significantly extend the battery life of sensors and other IoT devices.

4. UAV Flight Planning: It involves determining the most efficient routes for drones to follow when collecting data. This helps to optimize the use of drone resources and ensure that data is collected in a timely and comprehensive manner. The algorithm uses geographic information to generate a path that minimizes the distance traveled by the drone while maximizing the amount of data collected. Once the path is generated, the drone can be deployed to collect data along the predetermined route. The data collected by the drone is then transmitted wirelessly to an IoT base station or directly to the cloud for processing and analysis.

5. Precision Maps: Precision maps are created using machine learning algorithms that analyze data from multiple sources to generate a detailed view of the farm's fields and crops. This data can include information such as soil moisture, temperature, humidity, crop health, and nutrient levels. Precision maps generated provide farmers with a detailed understanding of their fields and crops, enabling them to make informed decisions about how to optimize their farm operations. For example, a precision map may identify areas of the field that are experiencing stress or disease, allowing farmers to take action to address these issues before they impact crop yields.

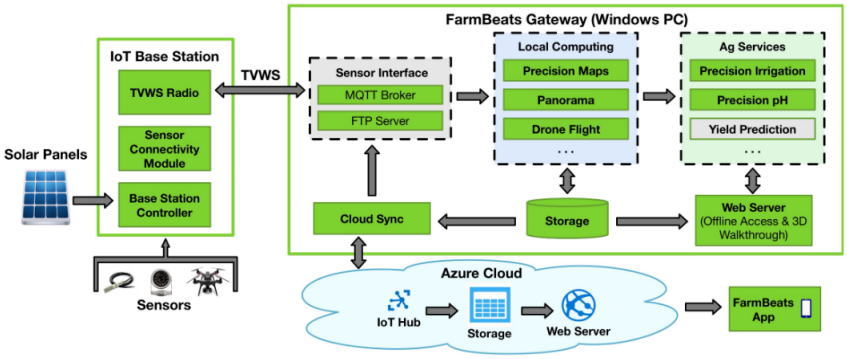

Figure 2.2: [167] The overview of FarmBeats System

2.3.2 FarmCommand

FarmersEdge [7] is a Canadian agricultural technology company that specializes in offering digital solutions to farmers in order to optimize their operations and maximize yields. FarmCommand is a platform developed by FarmersEdge that serves as a centralized hub for farmers to collect, monitor, visualize, and analyze data to manage their agricultural operations effectively. It is a suite of many tools as shown in Fig 2.3 that provide an end-to-end platform for increasing crop production and mitigating risk. It also offers an intuitive user-interface that supports real-time insights on various devices. Following are the key features and functionalities of FarmCommand:

1. Site-specific weather monitoring: It integrates weather data from various sources to deliver hyper-local weather information specific to each field. This feature helps farmers make informed decisions about irrigation, planting, and other field activities based on current and forecasted weather conditions.

2. Satellite Imagery: With satellite imagery capabilities, farmers can access high-resolution images of their fields from space. This feature allows for comprehensive crop monitoring and assessment of field conditions, including crop health, growth patterns, and potential issues such as pest infestations or nutrient deficiencies.

3. Equipment Tracking and Maintenance: It helps farmers to manage their machinery and equipment so that they can track the location, status, and usage of their agricultural equipment in real-time. It also provides maintenance reminders and alerts based on equipment usage and in case of any failures.

4. Predictive Modelling: It generates forecasts and insights by analyzing historical and real-time data. It predicts crop growth, yield potential, and field conditions that help farmers make informed decisions, optimize resource allocation, and proactively address issues such as pests, diseases, and nutrient requirements.

5. Scouting: It gives the ability to farmers to document and monitor their field scouting activities, collecting data on pests, diseases, and weeds. The platform supports and

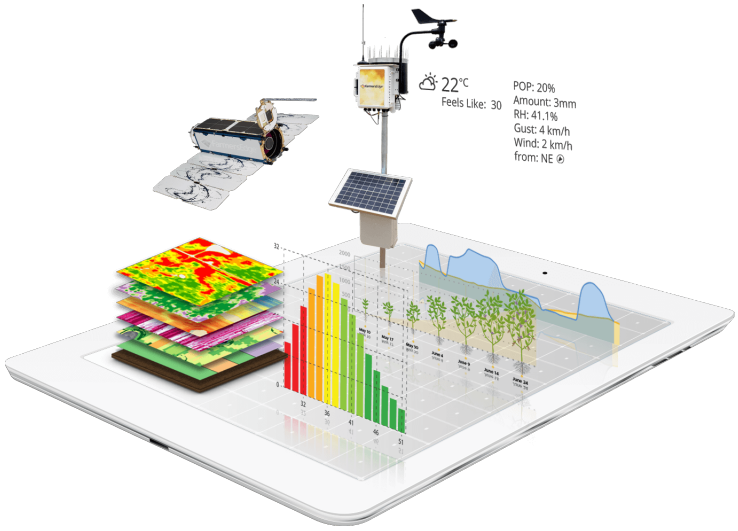

Figure 2.3: Farmcommand [7] is an end-to-end smart farming platform offered by FarmersEdge.

facilitates geo-referencing and collaboration, hence enabling scouts to share their observations and recommendations.

2.3.3 AgExpert Field

AgExpert [1] is a suite of farm management software products developed by Farm Credit Canada, which is a financial institution that provides loans and other finance-related services to the agriculture and agribusiness sectors in Canada. AgExpert has two major components AgExpert Accounting and AgExpert Field. AgExpert Accounting aims to simplify farm financial management and improve profitability. It helps farmers track income and expenses, manage accounts, and generate financial reports. The software streamlines tax reporting and provides tools for financial planning. AgExpert Field [1] enables farmers to effectively monitor and manage various field tasks, including planting, fertilizing, spraying, and harvesting. The platform incorporates functionalities like field mapping, crop planning, input management, and yield monitoring. By using AgExpert Field, farmers can streamline their field management processes, improve efficiency, and make data-driven decisions. The platform provides tools for analyzing historical data, generating reports, and identifying trends to optimize farming practices. It provides an intuitive dashboard where farmers can see the inventory of different crops using several visualizations as shown in Fig. 2.4 The following are the main features:

1. Field Mapping: It helps to map fields by defining their boundaries and areas using the platform's mapping features. This allows for precise planning and documentation of field activities, such as planting and harvesting. By digitally mapping the fields, farmers can optimize resource allocation, track progress, and improve overall field management efficiency.

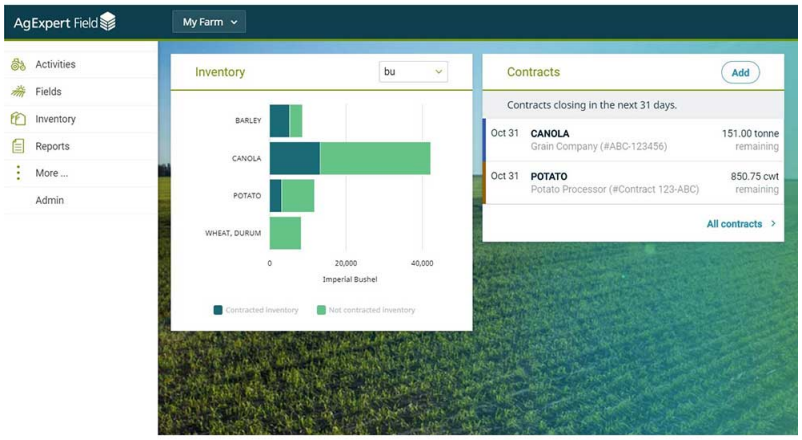

Figure 2.4: The figure [1] shows an example dashboard offered by AgExpert

2. Production Analysis: It allows farmers to evaluate the costs associated with their agricultural operations. By inputting data on input expenses, labor costs, and equipment usage, the platform generates insights and calculates the production costs. This analysis enables informed decision-making, optimization of resources, and assessment of profitability.

3. Record Activities: It allows farmers to document tasks such as planting, fertilizing, spraying, and harvesting. By entering relevant information like dates, quantities, and materials used, it helps farmers maintain accurate records of their activities. These recorded activities serve as valuable references for compliance, crop insurance, and financial planning purposes.

4. Inventory Management: This feature enables efficient inventory management for agricultural operations by providing tools for recording inventory levels, tracking usage, and generating reports. Using the recorded data, farmers can optimize inventory control, ensure timely resupply, and streamline field operations.

5. Planning tool: It offers a comprehensive planning tool for agricultural operations. It gives the capability to farmers to create and manage crop plans, including field layouts, planting schedules, and input requirements. It assists in optimizing resource allocation, tracking progress, and ensuring timely execution of tasks. With the planning tool, farmers can enhance productivity, streamline operations, and achieve their production goals.

2.4 Remote Sensing and Unnamed Aerial Vehicles

Remote sensing refers to the technique of analyzing and monitoring a specific area from a distance by studying the reflected radiation. There are two types of remote sensors, one type responds to external stimuli. For instance, radiation is captured by the reflected

sunlight. Whereas other types have built-in systems to capture data, such as a laser reflected by the sensor on the ground.

The application of remote sensing has been in agriculture since 1972 when it started with the first launch of the Landsat Multi-spectral Scanner System (MSS) satellite. Between 2015 and 2019, studies using satellites, UAVs, and manned aircraft collecting data using remote sensing for agricultural applications increased by 23% [103]. Due to the rise of remote sensing technologies that can capture various forms of data from any location on Earth, aerial imaging has emerged as a prevalent method for gathering data in the precision farming industry. Remote sensing devices such as RGB and Multi-spectral cameras are mounted on these aerial devices called UAVs (Unnamed Aerial Vehicles) that can capture a wide range of features from the fields. UAVs are also sometimes referred to as drones. They come in various sizes and can be used for various purposes, including surveillance, mapping, and photography. [163] outlines five major categories for UAV design including Fixed-wing, Rotary-wing, Blimps, Flapping wing, and Parafoil-wing.

UAVs are increasingly being used in farming for tasks such as crop scouting, mapping, and crop dusting. They can provide farmers with detailed information about the health and growth of their crops, allowing them to identify and address issues such as pests, diseases, and nutrient deficiencies. For instance, [99] used a time series method for the analysis of environmental data for weather forecasting, soil, and moisture evaluation. [36] used the data captured from Sentinel-1 to classify various kinds of crops. The data obtained from UAVs has been utilized for identifying crop diseases [115], managing animal husbandry [144], and applying herbicides and other chemicals to crops, which has proven to be more precise and efficient than conventional methods.

Using UAVs for farming can have a number of benefits over satellite imagery. For example, they can cover large areas quickly and at a lower cost than manned aircraft. They can also operate in areas that may be difficult for humans to access, such as steep or rugged terrain. Furthermore, UAVs are capable of collecting data at a higher frequency and resolution compared to satellite imagery. This capability is particularly useful for monitoring crop growth and can lead to the production of clearer images with minimal atmospheric interference. Several types of research have been done, to further optimize the UAV performance, for instance, [98] proposes a multiple autonomous UAV network for increasing data capturing efficiency and reducing time, [155] proposes a prototype for connecting UAVs in joint survey missions which are adaptively configured plan due to disruptive events.

2.4.1 Challenges

The application of remote sensing in agriculture faces certain challenges and limitations as outlined below [93] [91]:

a) Limited flight time: Most drones have a limited flight time due to battery limitations. The average UAV has a relatively short flight duration of 20 minutes to 1 hour [163].

As a result, large farmlands require multiple drone flights or multiple devices to scope the area making it a time-consuming and expensive task.

b) Weather conditions: The use of drones in farming can be limited due to their sensitivity to adverse weather conditions such as wind, rain, and extreme temperatures. In such conditions, drones may be unable to fly or may be at risk of crashing, which can hinder their operation and effectiveness in data collection and monitoring tasks.

c) Navigation and obstacle avoidance: Expertise is required for accurately navigating them to avoid obstacles such as trees, buildings, and power lines. Advanced technology like GPS and obstacle avoidance sensors are necessary for this, but just like any other technology, they can malfunction if proper maintenance and precautions are not taken. They are not infallible, and accidents can occur.

d) Data processing: Drones have the capacity to gather extensive data, such as images and sensor data, which can be valuable for farmers. Nonetheless, in order to obtain helpful insights from this data, it must be processed and analyzed, which can be a time-consuming process and necessitates the use of sophisticated software and hardware.

e) Regulatory issues: The use of drones in farming is subject to regulations and restrictions. Farmers need to be aware of these regulations and obtain the necessary permits and licenses before using drones. Failure to do so can result in fines and legal issues.

f) Cost: UAVs can be expensive to purchase and maintain. Additionally, the cost of training personnel to operate and maintain the drones can add to the overall cost. This can be a challenge for small-scale farmers who may not have the resources to invest in this technology.

The data UAV data captured as a part of this research was done by InDro Robotics. The above challenges were tackled using advanced drone technology and expertise in aerial imaging. The company utilizes state-of-the-art drones equipped with high-resolution cameras and sensors to capture detailed images of farm fields and crops from various angles and altitudes. Therefore, farmers save on the high costs of purchasing and maintaining drone equipment, as well as the expenses associated with hiring and training skilled operators. Moreover, outsourcing to a professional drone service provider can ensure that the drone is operated safely and in compliance with local regulations, which can help to prevent any legal issues that may arise from improper drone operation.

2.4.2 Geospatial data

Geospatial data is the data about the different objects that are present on the earth's surface that have a geographic component in it. It combines the location characteristics of the object (latitude and longitude) and other attributes associated with the object such

as reflectance, etc. Geospatial data can be collected through various methods, including satellite imagery, aerial photography, and field surveys. The article referenced in [54] examines the management of geospatial data over the past decade and how it has served to connect the fields of information technology and geoscience.

Geospatial data is often stored and managed using Geographic Information Systems or GIS, which allows users to analyze, visualize, and manipulate the data in order to gain insights and make informed decisions. Some common types of geospatial data include maps, aerial and satellite images, terrain models, and demographic data. GIS is a technology that interprets geographical features as tabular data. It is a comprehensive system that enables the creation, management, analysis, and mapping of various types of data. By linking data to a map, GIS integrates location-based information (i.e., where things are) with a wide range of descriptive data (i.e., what those things are). This provides a foundation for mapping and analysis that is used in science and almost every industry. It helps users understand patterns, relationships, and geographic context. [68] outlines the involvement of geospatial groups to collect 3D data using laser to develop 3D GIS models of building envelopes and city spaces. The benefits include improved communication and efficiency as well as better management and decision-making.

There are two types of Geospatial data:

a) Raster data: A raster is a collection of matrices of cells or pixels that are organized into rows and columns as shown in Fig 2.5 where each pixel represents the reflectance value which is reflected by the elements on the field. The use of raster images is highly suitable for representing the surface of a field, especially when there are continuous changes occurring across landscapes. These types of maps are often referred to as surface maps. In addition to their suitability for surface maps, there are other reasons why maps may be stored as raster images. For instance, rasters have the ability to store polygons and lines, which can be analyzed spatially and temporally with high efficiency. As part of this research project, the orthomosaics, and cropped tiles were saved in raster format.

b) Vector data: Rather than representing geographic information as pixels in a raster format, it is possible to store this information using points, lines, and polygons. This approach is known as a vector format, where points represent discrete locations, lines represent linear features, and polygons represent areas. Vector data, as depicted in Fig. 2.6. It consists of individual points that are stored as pairs of coordinates representing physical locations on the earth's surface. These points can be connected using lines, and enclosed areas can be created by forming polygons. The use of points, lines, and polygons makes it possible to represent the features of interest in greater detail, allowing for more precise spatial analysis. Compared to the raster data, the vector data represents the information, which is sharp, clean, scalable, and efficient to store. As part of this research, the tiles were generated by drawing polygons on the orthomosaics using the location of a single yield point as a reference.

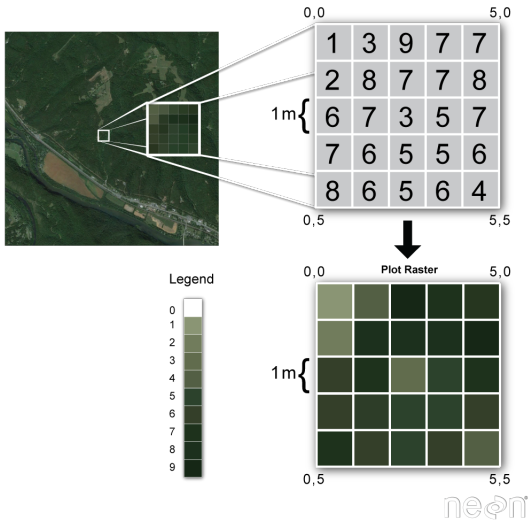

Figure 2.5: [21] Geospatial data in a raster format is organized into a grid of pixels, where each pixel represents a specific location on the earth's surface.

2.4.3 RGB and Multi-spectral Imagery

The RGB (Red, green, and blue) is a model to display colors in digital images. The three colors can be combined to create a wide range of colors that can effectively represent the geographical characteristics of a point on the surface of the earth. RGB colors are often used as features in satellite imagery, aerial photographs, and other types of raster data. RGB imagery is commonly used for mapping and monitoring land cover, vegetation, and urban areas. It can also be used to detect changes in the landscape, such as deforestation or urban expansion. It can be used for a variety of use cases in smart farming, such as [85] investigated linear relationships between Vegetation indices calculated from RGB images from UAVs and ground leaf chlorophyll content. [131] proposes an end-to-end deep learning framework for generating a digital elevation model from RGB imagery where low elevation can be represented by light green shades and higher can be represented by dark brown or black color. RGB features can be combined with demographic data to study population density in geospatial analytics.

Multi-spectral imagery is a type of remote sensing data where the sensors are sensitive to specific wavelengths of light. Unlike traditional RGB images, which are typically collected using sensors that are sensitive to the three primary colors of light RGB (Red, Green, and Blue), Multi-Spectral (MS) imagery includes data from a wider range of wavelengths, including the visible and Near-Infrared (NIR) parts of the electromagnetic spectrum. The additional wavelengths can provide additional features that the RGB wavelengths fail to capture such as vegetation, mineral content, and moisture. Healthy vegetation typically reflects more light in the NIR part of the spectrum than in the visible part of the spectrum, so MS imagery can be used to detect and map the distribution of healthy vegetation. [171] performed a correlation analysis between the vegetation indices derived from MS images and crop density of a canola field in western Australia. [32] provided high-resolution esti-

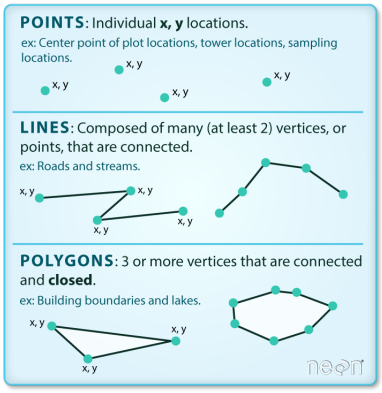

Figure 2.6: [24] Vector geospatial data is represented using individual points, lines, and polygons instead of pixels.

mates of surface soil moisture, evapotranspiration rates, and plant nitrogen and chlorophyll content using different bands. Some more use cases of MS imagery are in forestry, mineral exploration, and land use mapping. For this research, the high-quality field imagery was captured by InDro Robotics using both RGB and MS cameras mounted on the drones.

2.4.4 Vegetation Indices

A vegetation index (VI) is a computational formula that utilizes measurements of reflected light across various segments of the electromagnetic spectrum to deduce the condition, density, and physical characteristics of vegetation. The indices are used in remote sensing and agriculture to monitor vegetation growth, detect changes in vegetation health, and identify areas of drought or disease. They are typically calculated using measurements from satellite or aircraft-borne sensors. Photosynthesis requires plants to absorb a certain amount of light. However, with the help of VIs, the light that is reflected instead of absorbed by plants is analyzed. The insights derived from this analysis are then utilized to evaluate the health and condition of the plants. Vegetation indices primarily rely on light spectra within specific ranges, including Ultraviolet (10 to 380 m), Visible (comprising Blue, Green, and Red with wavelengths of 450-495 nm, 495-570 nm, and 620-750 nm respectively), as well as Near and Mid-Infrared (ranging from 850-1700 nm) [49] [61]. Observations have revealed that plants with high chlorophyll content, indicating good health, exhibit a lighter reflection in the Near-Infrared (NIR) and Greenlight spectrums compared to plants that are in poor health and have withering leaves. [165] used VI derived from UAV-acquired datasets, to accurately estimate the leaf chlorophyll content. According to [175] the identification of regions of interest and extraction of simple vegetation indices becomes quite challenging when the vegetation being analyzed has multiple vegetation indices due to spatial variability. Additionally, vegetation indices corresponding to other types of vegetation, such as weeds or cover crops, can be indistinguishable from those of the vegetation of interest, adding to the difficulty of analysis.

Some of the vegetation indices widely used in yield prediction are below:

a) **Normalized Difference Vegetation Index**: NDVI is one of the most widely used indices derived from multi-spectral data as a normalized ratio between the red and near-infrared bands [141] [100]. For instance, [73] used NDVIs as features for predicting Sugarcane yield in Brazil. [30] used NDVIs for estimating the yield of spring maize, [33] modeled Grassland biomass in Ireland using NDVIs extracted from Multi-temporal remote sensing data, [169] used deep learning techniques for predicting crop yield in Argentina using NDVIs and [51] uses NDVIs combined with images and crop information as parameters and proposes Spiking neural networks for the spatial-temporal analysis of image time series making use of low power-consuming hardware platforms.

b) **Green Normalized Difference Vegetation Index**: GNDVI [82] measures the greenness of the plant. The chlorophyll index is employed during the later stages of development as it reaches saturation later than NDVI. It is beneficial for assessing crop health in the initial stages and can be applied in crops with high canopy density or during more advanced growth stages. It is one of the most popular vegetation indices for calculating crop canopy water and nitrogen uptake. It uses near-infrared (NIR) and green bands (GREEN) for calculation. [145] used GNDVI for field-scale crop production using WorldView-3 and PlanetScope satellite imagery. [72] used it as a feature derived from the sensor data to predict wheat yield using machine learning. [119] evaluated a comparison study using Green, Red, and Near Infrared Bands for Predicting Winter Wheat Biomass, Nitrogen Uptake, and Final Grain Yield.

c) **Enhanced Vegetation Index**: EVI is a useful tool for evaluating vegetation greenness [90], which is comparable to NDVI. However, it is particularly effective in areas of dense vegetation because it accounts for the noise produced by the canopy background. Additionally, EVI includes the "B" blue band, "L" parameters for the canopy background, and "C" coefficients for atmospheric resistance, which enable more accurate calculations between red and near-infrared wavelengths while minimizing the impact of atmospheric noise, background, and saturation. It has been widely used in precision farming, for instance, [145] used EVI as a combined feature with other vegetation indices to predict crop yield using satellite imagery, [156] performed a comparison study of using EVI and NDVI for large-scale rice yield estimation and [75] performed a study on the variability of EVI in the dry season of a tropical forest.

2.4.5 Orthomosaics

Orthomosaic is a combination of multiple smaller images, that has been stitched together to form one big image. The picture is similar to Google Earth but offers superior resolution and provides detailed information as if captured by an aerial photograph. The stitched images are corrected for lens distortion, camera tilt, perspective, and topographic relief. It is used to create a detailed map of large areas, such as cities, forests, and fields. The generated maps can be used to perform spatial and temporal analysis to track changes in the landscape over time. As per [83] accurately georeferencing the Orthomosaic is one of the significant challenges. There are two methods for georeferencing, direct and

indirect. [169] discussed a direct method called Real-time Kinematics that can be utilized during UAV flight. Whereas [114] introduced a swath-based mosaicking approach and compared it with the standard blending modes. They are being widely used in agriculture, as they provide the overall view of the field wherein the Multi-Spectral orthomosaics can be used to understand the vegetation health of the field by using vegetation indicators. For instance, [169] used deep transfer learning on orthomosaics for crop yield prediction. The raw imagery used in this study was field Orthomosaics generated by InDro Robotics using Pix4D software.

2.5 Yield prediction

Crop yield prediction is an important task for decision-making in agriculture. Understanding the yield can lead to better farming practices, that can answer the questions of what to grow and when to grow. Historically, farmers have relied on their personal understanding of their fields and the prevailing climatic conditions to make yield predictions. However, this approach is heavily reliant on their experience and can be subject to inaccuracies and inconsistencies. This traditional method of yield prediction often lacks the reliability and precision necessary to optimize crop production and ensure food security. Population explosion, variability in natural weather due to issues like global warming, soil loss, and climate-changing demands are some factors that impact the yield, and understanding these factors can lead to optimal crop growth and timely production [89]. Thus, by understanding the yield patterns of different crops, governments can also identify potential areas for improvement in agricultural practices and invest in research and development to increase crop productivity. [132] discusses that the accurate prediction of crop yield in large areas over low cost is critical for grain policy-making and food security. The early estimation of crop yield, particularly when combined with spatial mapping of within-field variations, is a crucial factor in enabling site-specific management practices such as fertilization, irrigation, and pesticide application [120].

The use of IoT devices has led to the implementation of various sensors in fields to gather data. This data can then be analyzed to better understand the characteristics of the field and their relationship to crop yield. Overall, the yield prediction can be seen as a statistical regression problem. [52] advocated that spatial data usage in regression is treated differently from non-spatial data. During regression analysis, the non-spatial data loses its spatial relationship, which is crucial in fields, and leads to underestimation of true prediction error. Few published studies on field-scale or farm-scale yield prediction have been done in crop yield prediction studies, which have previously explored yield prediction models for regional or national scales [149]. An extensive comparison of the different algorithms for yield prediction used in the background research is presented in Table 2.1. A variety of yield prediction approaches has been presented in literature, such as [164], which used spectral data gathered using remote sensing to grain yield variation, and analysis of spatial and temporal yield using crop models. [44] used neural networks [143], massive crop yield prediction using machine learning [84], [29] data mining techniques, etc. However, they come with some challenges and drawbacks.

2.5.1 Challenges

[173] Outlines that yield prediction is one of the most challenging problems in precision agriculture as the crop yield depends on a lot of heterogeneous external factors such as soil quality, weather, landscape, soil fertilizer, pest infestations, quality and accessibility of water, seed variety and so on, which makes it a complicated task [87]. Achieving a balance between accuracy and practicality is another challenge in yield prediction. While high levels of accuracy can be attained with a significant amount of data [132], the process of collecting and analyzing such data can be resource-intensive, both in terms of time and cost. As a result, it is important to strike a balance between the desired level of accuracy and the available resources for data collection and analysis. Internal statistical dependency [154] is another challenge related to the large range of factors that can impact crop yield. [172] highlights that yield processes and approaches are time-specific and nonlinear. The traditional statistical methods fail to capture this highly nonlinear relationship. [142] outlines the usage of regression models for non-spatial data like regression trees, SVM, etc., and raises the issue of assuming statistical independence during the cross-validation approach. The research proposes a novel cross-validation technique to overcome the classical approach of yield prediction tasks.

This research overcomes these challenges by (a) using high-quality orthomosaics taken over the entire growth stage by experts from IndroRobotics (b) utilizing advanced post-processing methods to overcome the computation issues of handling large volumes of image data (c) proposing deep learning methods like 3D CNN which have been proven to capture highly nonlinear relationships in unstructured data like images (d) experimenting with a variety of model optimization techniques to improve the performance on unseen data.

2.5.2 Machine Learning

Machine learning (ML) based yield prediction is a topic that is being studied more and more globally and has been used in agriculture for a while. Machine learning is a method of teaching computers to learn from data, without being explicitly programmed. It involves using algorithms and statistical models to analyze and understand patterns in large sets of data and then using that understanding to make predictions or decisions without human intervention. ML can be used for a wide range of tasks, such as image and speech recognition [63], natural language processing [101], and predictive modeling [122]. Machine learning can be used in yield prediction to analyze large amounts of data from various sources, such as weather data, soil data, and historical yield data, and make predictions about future crop yields. There are several approaches to using machine learning for yield prediction, depending on the type and amount of data available, and can be broadly categorized into three main types: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning can be used when historical yield data is available, along with other relevant data such as weather and soil data. A model can be trained on this data to predict future yields based on the relationship between the input data and the output (yield). Unsupervised learning can be used when only historical yield data is available, and the goal is to find patterns and trends in the data. Clustering algorithms can be used to group similar

yield patterns together, and then a model can be trained to predict future yields based on these patterns. Our research involves utilizing supervised ML, where the continuous yield values obtained from the harvester were used as the ground truth labels. These values were captured in shapefiles, where each point in the field is assigned a single yield value based on its coordinates.

To tackle the task of yield prediction, it is imperative to choose the appropriate algorithms that can capture the relationship between the training data(images) and the training label(yield). In addition, the algorithms and the supporting platforms must be able to handle the volume of data. Machine learning has been widely used in yield prediction, for instance, [97] used multiple linear regression models to forecast crop yield in Canadian Prairies, [16] utilized, models such as support vector machine, regression trees, and k nearest neighbor, for massive cross yield prediction, [133] analyzed the soil behavior, and prediction of crop yield using Naïve Bayes and k nearest neighbor algorithms, [102] used high resolution, remotely sensed data and used machine learning algorithms including random forest, and support vector machine for spatial prediction of soil properties and corn yield, [76] used data mining techniques combined with machine learning algorithms like decision trees, and logistic regression, k-nearest neighbor for predicting rice yield in Humid Subtropical Climatic Zone and [159] studied classification techniques for crop yield forecasting.

Machine learning has been widely used for yield prediction, but it still faces several challenges, including limited data availability, high variability, complex interactions, and scalability issues. Yield prediction requires historical data on weather, soil, and crop management practices, which can be challenging to collect and may not always be available. This can lead to insufficient training data for ML models, resulting in lower accuracy and reliability in yield prediction. Another challenge is the high variability of crop yields, which can be influenced by factors such as weather, pests, and diseases. The complex interaction between these factors makes it difficult to accurately predict future yields using machine learning models. Scalability is another challenge in machine learning for yield prediction. Yield prediction models need to be scalable to handle large-scale farming operations, which can be challenging to achieve with traditional machine learning methods. Some of the widely used machine learning algorithms for crop yield prediction are described below:

2.5.2.1 Random forest

Random Forest (RF) [48] is a popular ensemble learning method for classification and regression tasks in machine learning. It is a collection of decision trees, where each tree is grown independently from a random subset of the data. The final output of the random forest is the average or majority vote of the outputs from all the decision trees as shown in Fig. 2.7. A decision tree is a flowchart-like tree structure, where each internal node represents a feature (or attribute), each branch represents a decision based on that feature, and each leaf node represents the outcome. The basic idea behind the decision tree is to identify the feature that provides the most information gained about the target variable. Random Forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

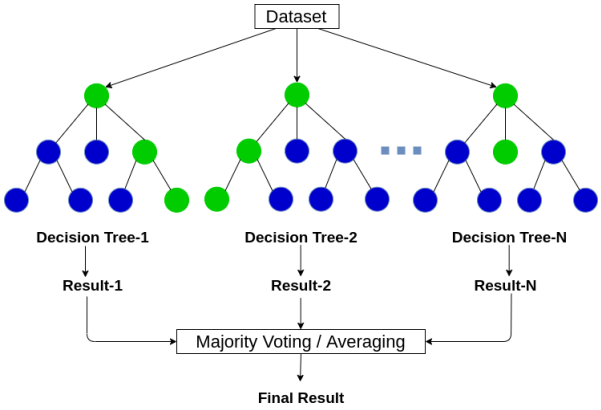

Figure 2.7: [24] Random Forest Algorithm

The main advantage of Random Forest is that it reduces overfitting, which is a common problem in decision tree models. Overfitting occurs when a model is too complex and fits the training data too well, but performs poorly on new, unseen data. Random Forest addresses this problem by averaging the predictions of multiple decision trees, which reduces the variance and increases the overall accuracy of the model. The random forest has been widely used in crop yield prediction. For instance, [70] used the random forest for predicting what sugar can yield whereas, [95] used RF for predicting yield from various data sources including wheat grain potato tuber and maize silage yield. [166] highlighted that the random forest was the second most used machine learning algorithm for predicting yield.

2.5.2.2 Support Vector Machines

Support Vector Machines (SVM) [95] is a type of supervised machine learning algorithm that can be used for classification or regression tasks. The algorithm finds the best boundary (or "hyperplane") that separates the different classes in the data. SVMs are based on the idea of finding a hyperplane that maximally separates the different classes in the data. The distance between the hyperplane and the closest data points from either class is known as the margin. The goal of an SVM is to find the hyperplane with the largest margin, as this will lead to the most robust classifier. The classification using hyperplane using maximum margin is shown in Fig. 2.8. SVMs can also be used for non-linearly separable data by introducing a technique called the kernel trick. The kernel trick maps the input data into a higher-dimensional space, where a linear boundary can be found to separate the classes. Common examples of kernel functions include the polynomial kernel and the Radial Basis Function (RBF) kernel.

SVMs are widely used in many applications such as Image Classification, Text Classification, and Bio-informatics. One of the main advantages of SVM is that it works well even with Unstructured and Semi-structured data like text, images, and trees. SVMs are also robust to overfitting and work well with small datasets. Contrary, SVMs can be memory-intensive and computationally expensive when working with large datasets. [166]

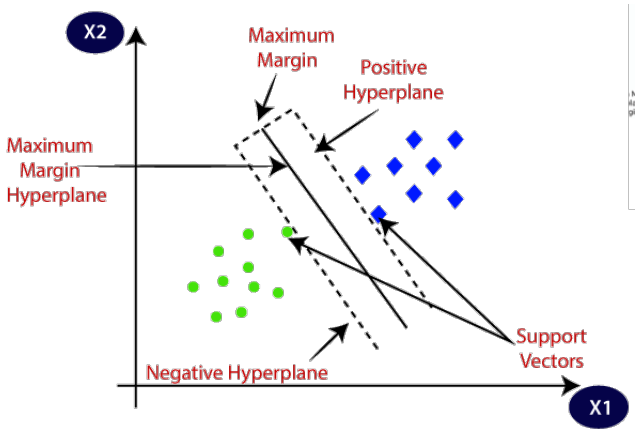

Figure 2.8: [23] Support Vector Machine

highlights that SVM was the third most used machine learning algorithm for yield prediction. [77] used SVM to predict rice yield in the subtropical climate of India. [176] proposes a SVM-based open crop model (SBOCM) for predicting rice yield in China. [79] proposes an approach to increase the net yield rate of crops, based on rainfall and uses different machine learning algorithms including KNN, and decision trees.

2.5.2.3 Artificial Neural Networks

Artificial neural networks (ANNs) [94] are a type of machine learning model that is inspired by the structure and function of the human brain. They consist of layers of interconnected "neurons" that process and transmit information. ANNs are used for a variety of tasks, such as image recognition, natural language processing, and decision-making. They can be trained using large amounts of data and can improve their performance over time through a process called "learning". There are many different types of ANNs, but the most common is the feed-forward neural network, also known as a Multi-Layer Perceptron (MLP). This type of network consists of an input layer, one or more hidden layers, and an output layer as shown in Fig. 2.9. The input layer receives the input data, the hidden layers process it, and the output layer produces the result. The processing units in each layer are called Neurons, which are connected to each other by pathways called Edges or Connections. Each connection has a weight, which determines the strength of the connection. In the training phase, these weights are adjusted to minimize the error between the predicted output and the true output.

[166] highlighted that ANN was the most used algorithm for crop prediction. Due to the fully connected neurons and non linear activation functions like Sigmoid, ANNs are highly capable of learning highly nonlinear relationships. For instance, [43] used ANNs to predict paddy yield at 3 different locations based on 10 years of historical dataset, [181] used meteorological data including temperature and rainfall records to predict wheat yield in Turkey. Another study was conducted by [78] to predict rice crop yield which is a major crop in India using a massive yield dataset from 27 districts. [57] did an early season apple yield analysis using a hybrid model based on image analysis and neural networks.

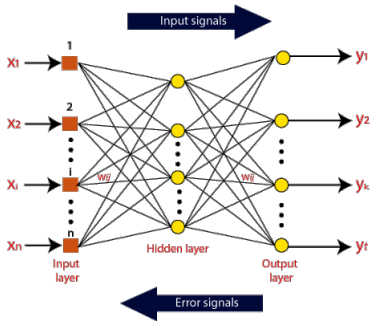

Figure 2.9: [3] Architecture of a single layer Artificial Neural Network that consists of an input layer, hidden layer, and output layer

2.5.3 Deep Learning

Deep learning or DL is a sub-field of machine learning that utilizes neural networks with multiple layers to learn and model patterns in data. These networks are trained using large amounts of labeled data and are capable of learning complex, nonlinear relationships. Deep learning techniques have been used to achieve state-of-the-art performance in tasks such as image recognition, natural language processing, and speech recognition using architectures such as convolutional neural networks (CNNs) [130] and recurrent neural networks (RNNs) [108]. Deep learning models are trained using large datasets and powerful hardware, such as GPUs. The process of training a deep learning model involves feeding the model a large dataset of input-output pairs and adjusting the model's parameters so that the model's predictions match the outputs in the training data.

The development of Deep Learning (DL) algorithms in recent years has provided reliable and clever ways to enhance the mapping of the Earth's surface. In recent years, as computer processing and labeled instances (i.e., samples) became more accessible, Deep Neural Networks (DNN) performance in image-processing applications has significantly improved. [129] mentioned that DL is a fresh method for processing infrared thermal images in several fields, particularly when using data from satellites. In some of the research where DL was combined with remote sensing data, [35] used DL methods with the terrestrial imagery to detect potholes on roads, [170] detected land surface temperatures combined with multi-spectral observations from orbital platforms, [170] determined sea surface temperatures from orbital imagery to identify.

DL has shown promising results in overcoming some of the challenges in machine learning for yield prediction. The models are designed to learn hierarchical representations of data, allowing them to capture complex patterns and relationships between variables. This makes them well-suited for yield prediction, where there are often complex interactions between weather, soil, and crop management practices. Moreover, DL models can handle large amounts of data and scale effectively, making them suitable for large-scale farming operations. Existing researchers have tried different novel DL architectures to achieve different goals in smart farming, such as [55] used two deep networks, one for analyzing the spatial data and the other one for LiDAR data, [86] proposed a deep stacking model for Hyper-spectral Imagery classification that includes nonlinear activation in the hidden lay-

ers without the use of SGD for training and [81] developed a deep convolution auto-encoder to classify high-resolution SAR images.

2.5.3.1 Challenges

Some of the challenges related to deep learning are as follows [41]:

a) Inadequate data sets: DL algorithms require a lot of data to train. [56] pointed out that many existing remote sensing datasets lack image variation, and diversity and have a smaller number of classes.

b) Human-understandable solutions for modeling physical phenomena: For various remote sensing applications, simply arriving at a conclusion is inadequate because users require an understanding of the system's decision-making process to gauge its dependability. Nonetheless, deep learning (DL) systems are extensive and intricate, making them difficult to evaluate. [118] and [67] pointed out that one of the major disadvantages of DL algorithms is to understand what they are doing and often remains a black box to the end user.

c) Big Data: Previously, DL algorithms were commonly used on small-scale RGB and gray-scale images. However, in the case of data acquired via remote sensing, there are specific challenges that must be taken into account. The reason for this is the existence of noise and complexity in data obtained through remote sensing, which is caused by a high number of channels ranging from four to hundreds. [41] highlighted that most remote sensing suffers from a lack of good quality training data.

d) Non-traditional heterogeneous data sources: Data from various sources like social media is fused with remote sensing data to achieve better performance, for instance, [74] enhanced RS data for flood assessments using information gleaned from social media images. Since a lot of factors could influence performance, the challenge of optimal data fusion arises. Homogeneous data such as data captured from sensors have semantic variation but a little-to-no semantic variation. However, fusion becomes more syntactically and semantically challenging if human information (such as linguistic or text) or algorithmic outputs (such as binary decisions, labels or symbols, probabilities, etc.) are involved.

e) Optimal architecture for spectral, spatial, and temporal data: The existing DL components and architecture may not be effective and adequate to deal with RS data. [34] has highlighted the uncertainty in the overall network as more layers are added in the case of remote sensing data. An additional challenge is determining the optimal training method for training the network. This includes the selection of appropriate loss functions, optimizers, and learning rates.

The proposed study utilizes drone-captured 3-channel RGB data of high quality, captured by experts, for three distinct fields throughout the growth period. This approach

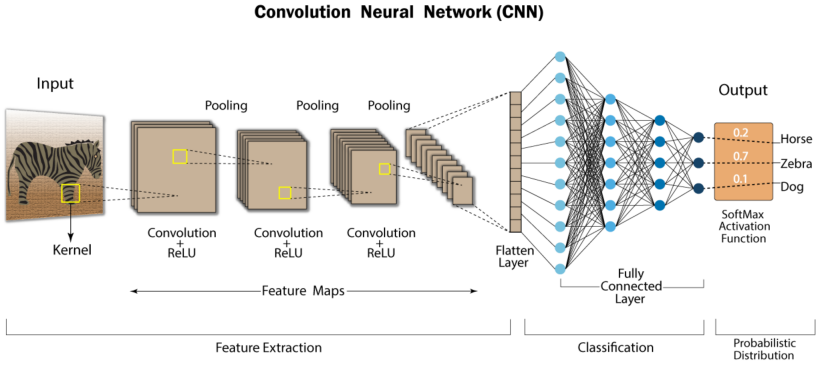

Figure 2.10: [3] Architecture of a Convolutional Neural Network with different layers that classify an input image into three classes

effectively addresses challenges related to data accuracy, diversity, and volume. A series of rigorous experiments were conducted, exploring various model architectures and optimization techniques, in order to identify the most accurate model capable of capturing spatial and temporal relationships between the input field images and yield values.

2.5.3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) [130] are a specific type of neural network that is designed to process data that has a grid-like topology, such as an image. CNNs are particularly useful for image and video recognition tasks, as well as natural language processing tasks. CNNs are composed of an input layer, several hidden layers (typically made up of convolutional and pooling layers), and an output layer as shown in Fig. 2.10. The convolutional layer in CNN is responsible for detecting features in the input data, such as edges, textures, and patterns. This layer uses a set of filters, which are applied to small regions of the input data, to extract features from the image. The pooling layers in a CNN are used to down-sample the data, reducing the dimensional of the input and allowing the network to focus on the most important features. Once the features have been extracted by the convolutional and pooling layers, they are passed to the fully connected layers, where they are used to classify the input image into one of several possible classes. The final output of the CNN is a probability distribution over the classes, indicating the likelihood that the input image belongs to each class.

Crop yields can be predicted by utilizing CNNs to analyze crop images captured throughout the growing season. The images provide crucial features such as crop size, shape, color, and overall plant health. These features are then utilized to anticipate the final crop yield. A CNN can be trained to identify patterns in the images that correspond to high or low yields. For instance, a CNN may learn to recognize that healthy and fully grown plants usually result in high yields, whereas in empty areas unhealthy plants tend to result in low yields.

[166] highlighted that CNNs were the most widely used deep learning algorithm for

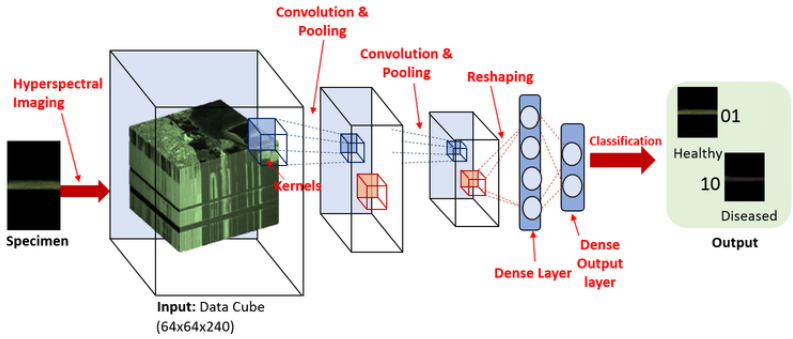

Figure 2.11: [3] Architecture of 3D Convolutional Neural Network that consists of 3D operations between different convolution and pooling layers. The feature map is flattened before feeding to the dense layers which is similar to 2D CNN

yield prediction. The process of training a CNN for yield prediction typically involves feeding it a large dataset of images of crops that will serve as the Training data (X) with the corresponding yield data that will serve as Training Labels (Y). The CNN then learns to recognize patterns in the images that are associated with high or low yields. After training, the CNN can predict the yield before the season ends by processing the images captured from the field, following the same input approach as during training. Such yield forecasts can significantly aid farmers in making timely informed farming decisions.

It has been widely used in yield prediction, for instance, [28] used CNNs to predict cotton yield from RGB images acquired through a mobile device and achieved an error rate of 17.86%. when predicting yield using 205 images from the testing dataset, [124] combined weed detection, biomass evaluation, and yield prediction using CNNs on NDVI and RGB data acquired from UAVs and [137] proposes a novel deep CNN algorithm for yield prediction and automatic counting of fruits and vegetables using UAV images.

2.5.3.3 3D Convolutional Neural Networks

A 3D CNN [138] (Three-Dimensional Convolutional Neural Networks) is a type of CNN algorithm that is specifically designed to process 3-dimensional data, such as videos or 3D medical imaging. It works by applying a set of filters that operate on three dimensions (height, width, and time or depth) of a video or image sequence. This means that 3D CNNs can capture the dynamics of a scene over time, making them useful for tasks such as action recognition, where the motion of objects is critical in identifying the action being performed. These features are then passed through multiple layers of the network, where they are further processed and classified. 3D CNNs are commonly used in computer vision tasks such as action recognition, object detection, and segmentation.

A 3D CNN consists of multiple layers that collaborate to evaluate the input data and provide a forecast. The key layers in a 3D CNN are shown in Fig. 2.11 and described below:

1. Convolutional Layer: This is the main layer where the filters are applied to the input data to extract features. The filters are typically small and move in three dimensions of the image (height, width, and time or depth) as compared to 2D CNN where filters move in height and width only.

2. Pooling Layer: This layer is used to down-sample the feature maps, reducing the spatial dimensions of the data and increasing the robustness of the network. This works the same as 2D pooling layers.

3. Fully Connected Layer: In order to analyze the features extracted by the preceding layers and make predictions, one or more fully connected layers are utilized. It consists of interconnected neurons, where an activation function is applied to introduce non-linearity.

4. Output Layer: This layer produces the final prediction. It typically contains a single neuron that outputs a continuous value as the prediction.

In agriculture, 3D CNNs can be used to analyze 3D data, such as images or point clouds captured from drones or other aerial platforms. This can be used for tasks such as crop counting and crop health monitoring. For example, 3D CNNs can be trained to detect plant diseases or pests [107] by analyzing images or point clouds captured by drones. This can help farmers identify areas of their fields affected by a particular disease or pest and take appropriate action to address the problem. 3D CNNs can also be used to estimate crop yields by analyzing images of plants captured by drones through different phenological stages as it is effective in capturing changes over time. This can help farmers to get early predictions of the crop yield from models trained on historical data and make more informed decisions about planting, fertilization, and harvesting. 3D CNNs have been widely used in smart farming, for instance, [161] used NASA's MODIS satellite data to predict yield in the USA utilizing 3D CNNs to extract spatiotemporal features, [125] investigated the Performance of 3D CNNs to classify tree species in the boreal forest, which was focused on pine, spruce, and birch trees by using RGB and Hyperspectral data and [174] combined 2D and 3D CNNs to determine the land cover discrepancy in the assigned land category of cadastral map parcels.

2.5.4 Evaluation

Yield prediction is usually approached as a regression problem in which the network generates continuous numerical yield values. This form of supervised learning involves learning a mapping between input features and continuous target values. In this context, the input features consist of farm Orthomosaics captured by UAV and divided into smaller tiles measuring 9 x 9 meters, while the target values correspond to the yield values collected by the harvester. Regression evaluation metrics are used to assess and measure the accuracy of a regression model's predictions. The evaluation metric chosen is determined by the specific requirements of the solved problem. The regression evaluation metrics commonly used in the literature are listed below:

2.5.4.1 Mean Absolute Percentage Error (MAPE)

It is one of the most popular regression evaluation metrics because of its simple interpretation in terms of relative error [14]. It computes the average absolute percentage difference between the actual and predicted target values 2.1. It is often used when the target variable has a wide range of values and when it is important to understand the magnitude of the error in percentage terms. Despite its usefulness, MAPE has some limitations that need to be taken into account. For instance, it can be sensitive to extreme values and large outliers, which can distort the results. Additionally, MAPE can be difficult to handle when the actual values are zero, which can lead to division by zero errors. Therefore, it is important to use MAPE along with other metrics to get a more comprehensive understanding of model performance. From a statistical standpoint, the regression models try to find a measurable function g(x) or gMape(x) such that g(X) is as close to the Y (Y is the ground truth). Mathematically it is shown below:

$$MAPE = \frac{100\%}{n} \sum_{i=1}^{n} |\frac{x_i - y_i}{y_i}| \tag{2.1}$$

It has been used widely as an evaluation metric in yield prediction such as,[126] noticed a decreasing trend in MAPE and ME values when the neighborhood size increases during performance evaluation of spatial-temporal Multi-task Learning, in predicting yield whereas [116] combined correlation coefficient with MAPE and evaluated different Machine Learning model performance on several vegetation indices including NDVI, NDRE, etc.

2.5.4.2 Mean Absolute Error (MAE)

Mean Absolute Error (MAE) is a measure of the average magnitude of the errors in a set of predictions, without considering their direction. It is the average of the absolute differences between the predicted values and the actual values 2.2. MAE [13] is commonly used in regression problems to evaluate the performance of a model. Like other evaluation metrics, a lower MAE indicates a better fit of the model to the data, meaning that the predicted values are closer to the actual values. MAE is a robust metric, meaning that it is not affected by extreme values in the data, as opposed to metrics like Mean Squared Error (MSE) which can be heavily influenced by outliers. However, the MAE metric does not indicate the direction of the error (i.e., whether the predictions are overestimating or underestimating the actual values), so it may be less useful for certain types of analysis where the direction of the error is important. Mathematically it is represented below, where n represents the number of data points being compared, and y_i and $\hat{y}_i$ represent the true and the predicted values respectively.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{2.2}$$

[124] used MAE as their primary metric to compare different yield prediction models, where the best model was able to get an MAE value of 484 kg/ha using RGB images combined with thermal temperature as input.

Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE) are both measures of the accuracy of predictions in regression problems [12]. However, they differ in how they scale the errors and how they handle extreme values. One advantage of using MAPE is that it provides a relative measure of accuracy, which is useful when comparing the performance of models in different scales or with different units. Therefore MAPE was preferred to display the experimentation results in this research.

2.5.4.3 Root Mean Square Error (RMSE)

Root Mean Squared Error [22] (RMSE) is a commonly used measure of the difference between the actual and predicted values in a typical regression problem. It is the square root of the average of the squared differences between the predicted and actual values. The RMSE is a measure of the variability of the residuals, and it gives an idea of how far off the predictions are from the true values. The smaller the RMSE, the better the fit of the model to the data. Mathematically, RMSE is defined as below 2.3, where N is the number of data points, y(i) is the i-th measurement and $\hat{y}$ (i) is its corresponding prediction. Mathematically it is represented below :

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\Big(\frac{y(i) - \hat{y}(i)}{N}\Big)^2} \tag{2.3}$$

In the context of yield prediction, the RMSE can be used as a measure of the accuracy of a predictive model. In yield prediction, the goal is to predict the yield of crops, livestock, or other agricultural products, based on various factors such as weather conditions, soil characteristics, and management practices. The RMSE can be used to evaluate how well a predictive model is performing by comparing the predicted yield values to the actual yield values. [166] reviewed that RMSE was the most used evaluation metric for evaluating the predicted yield, followed by R-squared and MAE. [113] used RMSE and RMSE% as their evaluation metric in grain yield prediction and achieved an RMSE value of 688.2 using Support vector machines on data captured from RGB sensors whereas [42] used RMSE for comparing convolutional neural networks with Support vector machines, Random Forest regressor and Linear Regression models on nine corn fields.

2.5.4.4 Correlation Coefficient (CORR)

The Correlation Coefficient in yield prediction was used in this research as an evaluation metric to measure the strength and direction of the relationship between the predicted and actual yield values. The value ranges from -1 and 1, where 1 indicates a perfect positive correlation, -1 indicates a perfect negative correlation, and a value of 0 indicates no correlation between the two variables. Mathematically 2.4, it's expressed below where n

represents the total number of data points, x_i, and y_i are the values of the i-th data point in the actual and predicted yield dataset respectively, and $\bar{x}$ and $\bar{y}$ represent the means:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.4}$$

Ref	Algo	SD	TD	PD	SP	Advantages	Limitations
[97]	LR	Y	Y	Y	N	Simplicity and ease of implementation, faster training times, interpretable results, and feature significance	Asumption of linear relationship, limited flexibility in capturing complex patterns, sensitivity to outliers and constrained capability to handle high-dimensional data
[30]	SVM	Y	N	N	N	Ability to handle both categorical and numerical features, tweaking performance using different kernel functions, robustness to overfitting and can capture complex nonlinear relationships	Sensitivity to noise and outliers, limited scalability, limited capturing of nonlinear relationships and limited ability to handle large feature spaces
[69]	FSR	Y	Y	N	N	Iterative feature and model selection, flexibility in incorporating different base learners, faster training time and easy interpretation of the final model	Prone to local optima, limited handling of collinearity, vulnerable to noise and outliers, computational complex and lack of feature evaluation
[70]	RF	Y	Y	N	N	Handling high dimensional data, capture non-linear relationships and is robust to outliers.	Overfitting with noisy data, computational complexity, dimensionality issues
[84]	M5-Prime	N	Y	N	N	Nonlinear modeling capability, flexibility in handling missing data, no assumption of data distribution required, robustness to outliers, missing data and interpretable results	Limited in handling image noise, curse of dimensionality and computational complex
[40]	RF, SVM	Y	N	Y	Y	Capturing non-linear relationships, effective feature selection, ensemble learning, ability to handle missing data and performance can be tweaked by fine-tuning hyperparameters	Poor Interpretability, overfitting, computational complex and curse of dimensionality

[78]	ANN	Y	Y	N	N	Capturing complex nonlinear relationships and patterns, can handle high-dimensional data, scalability to larger and more complex networks, and continuous improvement with fine-tuning	A large amount of training data requirement, sensitivity to feature scaling and normalization, lack of interpretability, computational complexity and large training time
[113]	DNN	Y	Y	N	Y	Capable of capturing complex non-linear relationships and patterns, can handle high-dimensional data, scalability to larger and more complex networks and continuous improvement with fine-tuning	Overfitting issues, large data volume requirements, sensitivity to feature scaling and normalization and heavy computational resource requirements
[28]	CNN	Y	N	Y	Y	Automated feature extraction, effective in capturing spatial patterns, can capture both low-level image features and high-level semantic information, scalability to larger datasets, translation invariance and robustness to variations and State-of-the-art performance in image analysis tasks	Lack of interpretability, computationally intensive, prone to overfitting, sensitivity to hyperparameters and they cannot handle variable sized images as input.
[178]	LSTM	Y	Y	N	N	Ability to capture long-term dependencies in sequential data, robustness to vanishing gradient, can capture complex temporal patterns, and state-of-the-art performance in time series analysis	lack of interpretability, computationally intensive, large volume of data for training, limited long-term memory capacity and difficulty in handling variable-length sequences.

[80]	Conv-LSTM	Y	Y	N	Y	Ability to capture both spatial and temporal dependencies simultaneously, automatic feature extraction, scalability to larger datasets, parallelizable computations and state-of-the-art performance in spatiotemporal data analysis.	Computationally intensive, large volume of data for training, sensitivity to hyperparameters, prone to overfitting and complex model architecture
[123]	3D-CNN	Y	Y	Y	Y	Ability to handle volumetric data such as 3D images, scalability to larger datasets, ability to handle both spatial and temporal information, robustness to noise and ability to handle missing data, and State-of-the-art performance in spatiotemporal data analysis.	Computationally intensive, trade-off between model complexity and performance, increased model parameters, training time, and lack of interpretability

Table 2.1: Comparison of different algorithms for yield prediction in the literature. The acronyms used in the table LR (Linear Regression), SVM (Support Vector Machine), FSR (Forward stagewise algorithm), RF (Random Forest), M5-Prime (M5-Prime regression trees), ANN (Artificial Neural Network), DNN (Deep Neural Network), CNN (Convolutional Neural Network), and LSTM (Long Short-Term Memory), represent different machine learning algorithms (Algo) employed for yield estimation. Whereas the columns SD (Spatial Data), TD (Temporal Data), PD (Phenological Data), and SP (Subplots) represent different data considered in the respective research.

Chapter 3

Approach - Modelling Yield Prediction

This chapter presents the methodology that was used to develop the proposed deep learning model that can effectively capture the spatial and temporal relationships in corn fields. The developed model can provide early yield predictions for the season, which can help farmers make timely critical farming decisions. The process depicted in Fig 3.1 involves multiple stages, beginning with the acquisition of data and continuing with data preprocessing and fusion. Finally, the high-performing models undergo optimization, hyperparameter tuning and the most accurate model is chosen for deployment.

3.1 Data Acquisition

In this section, we describe the data collection process and how different sensors were networked in the IoT paradigm. To give an overview, [112] Internet of Things (IoT) refers to the network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, and connectivity which enables these objects to connect and exchange data, as described in the Section 2.2. IoT allows these physical objects to be sensed or controlled remotely across existing network infrastructure, creating opportunities for more direct integration between the physical world and computer-based systems and resulting in improved efficiency, accuracy, and economic benefit. IoT has played a significant role in developing precision farming by providing farmers with real-time data and analysis to improve their decision-making process. [64] The incorporation of IoT in precision farming has the potential to enhance crop yields substantially, decrease wastage, and enhance the sustainability and efficiency of agriculture. Nevertheless, it also brings up important concerns, including data security, data ownership, and the necessity for standardization and interoperability of devices and systems [112].

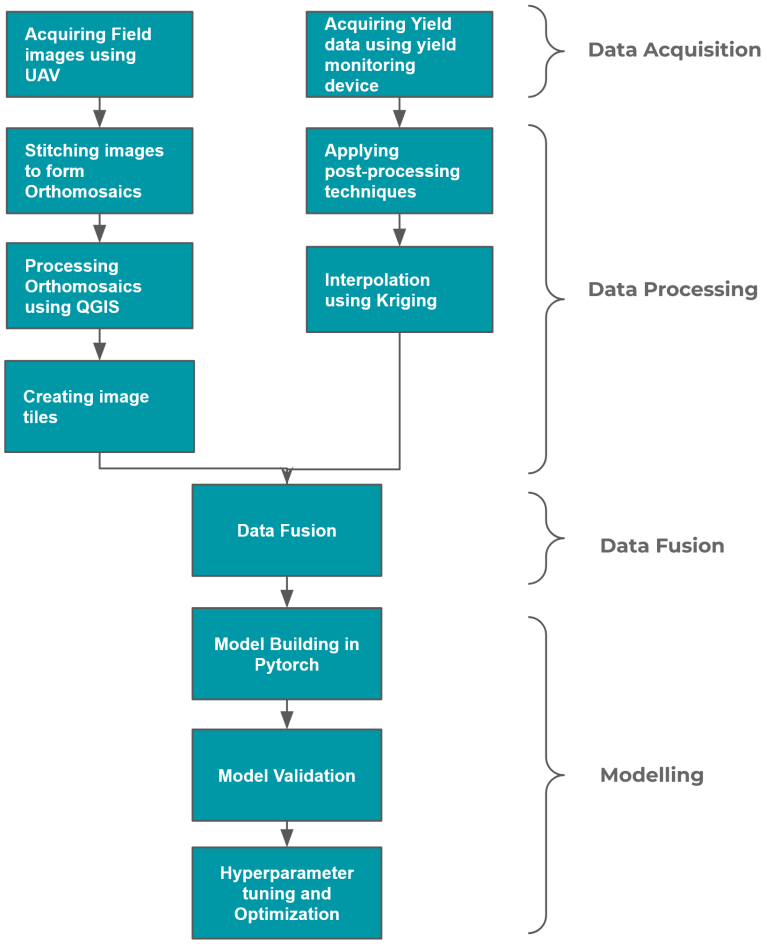

Figure 3.1: End to End life-cycle of the yield prediction model development

3.1.1 Area X.0

The data was acquired from Area X.0 which is a 1,866-acre agricultural facility operated by Invest Ottawa to manage a private test area for networked and autonomous vehicles. In this facility, corn is the main crop grown, with planting usually commencing in June and harvest taking place in September. It is fully equipped with advanced technology such as Wi-Fi, 4G/LTE to 5G networking infrastructure, GPS, and dedicated short-range devices. For this research, there were two data sources, (a) Yield data, a which is continuous measurement of the final yield acquired at the end of the season (b) Field orthomosaics which is a combination of many images stitched together to show a larger area captured by a drone during multiple flights.

Established in 2019 as a part of Area X.0, the Ottawa Smart Farm (OSF) was designed to demonstrate the advantages of IoT and data technology to farmers and encourage its adoption as a means of increasing efficiency. The OSF has formed partnerships with local collaborators as well as global agricultural technology enterprises such as Microsoft, Trimble, and FarmersEdge. The objective of this facility is to demonstrate the advantages

Field	Area in acres
1	3.38
2	4.67
3	3.55
4	2.07
9	1.43
11	48.08

Table 3.1: Field areas measured in acres.

of smart farming to the farmers and encourage them to adopt these technologies to optimize costs, increase production, and make critical agriculture decisions at the right time. At Area X.0, the OSF producer planted corn in six demonstration fields in 2021. The area (in acres) of these demo fields is shown in Table 3.1. However, due to a large volume of data only three fields, Field 1, 2, and 3 were considered for this study as shown in Fig 3.2.

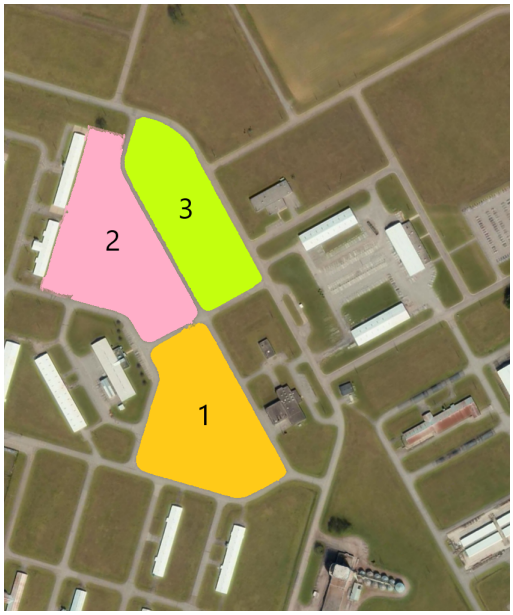

Figure 3.2: The image shows three demo fields where corn was grown (Field 1, 2, 3 and 9) highlighted on satellite imagery at AreaX.0

3.1.2 Yield Data

The yield data acquisition at the time of harvesting was done by Trimble which is an American hardware and software services company that provides solutions in various domains including agriculture. It provides several solutions including Land preparation, Planting and seeding, Crop protection and Spraying, Nutrient management, Water management, and harvesting. Data acquisition for farm management is done using several sensors and several tools are offered for offline analysis. It also supports data ingestion to its cloud

platform where a variety of other downstream services are offered. As a part of this research, the Trimble yield monitoring device was fitted on the harvester, to collect crop yields and moisture levels with some metadata including swath width and harvester speed, which were then used to generate precision maps and reports for further analysis. There are three layers in the Trimble suite:

1. Application layer: This is the topmost layer that offers a variety of tools for geospatial data analysis, and visualization of the fields, sensors, and equipment data using precision maps. The onboard tractor application is another functionality that it offers that helps the farmers track and visualize their tractor's field activity.

2. Middleware layer: It offers storage and streamlined access to raw data. There are inbuilt data preprocessing methods that support the precision map formation functionality.

3. Actuation and Perception layer: This is the physical layer that collects the data. Sensors that are network-enabled are attached to the tractors that send data to the Middleware layer for further processing. This data or farm-related measurements include the speed of the tractor, seed type and application rate, fertilizer type and application rate, field elevation, and Dry Yield. These measurements are combined with the GPS coordinates that help the analysis of some specific unit area within the field.

The measurements collected by Trimble were stored as shapefiles. A shapefile is a geospatial vector data format used in geographic information systems (GIS) software. Fig 3.4 shows the directory that contains 4 different files (a) the Shapefile DryYield.shp that contains the geometric data for the point features (b) associated attribute data (yield values), which are stored in DryYield.dbf (c) DryYield.shx is a positional index file that enables faster access to the data (d) DryYield.prj contains important projection information, including the coordinate system, units, and datum used in the data, which are necessary for proper rendering and analysis of the shapefile. For this research, Trimble captured five distinct measurements from Area X.0 fields including, Boundary, Dry Yield, Moisture, Speed, and Width, with each measurement represented as an independent entity as described below:

1. Dry Yield: The Yield is measured in bushels per acre (bu/ac). The yield data is gathered through the utilization of yield monitoring sensors, which are installed on the harvester. The weight of the harvested grains is calculated by the system, and these values are presented to the operator in real-time.

2. Moisture: The Moisture is measured in Percentage (%). The Trimble monitoring also provides inbuilt sensors that can measure the moisture from the crops. These sensors use various technologies to measure the moisture content, including capacitance, electrical resistance, and microwave.

3. Speed: The speed of the harvester is measured in Miles per hour (Mph). Trimble Yield Monitoring has inbuilt software and sensors that are able to record the speed of the harvester in real-time. Since the values of yield and moisture are displayed in real-time to the operator, they can adjust the speed accordingly. For example, if the crop is showing signs of high moisture content, the operator can reduce the speed of the harvester to allow the crop to dry out before harvesting. Guiding systems are also built in Trimble that can help the operator to navigate the field, in such a way that it reduces time and fuel for harvesting operations.

4. Width: The width refers to the width of the area being covered by a piece of agricultural equipment. The swath width is the distance between the outermost passes made by the equipment as it moves through the field. The swath width of a combined harvester is determined by the width of the header, which is the part of the harvester that cuts the crop. A wider header can allow the harvester to cover more ground in a shorter amount of time. The width of the harvester is measured in Feet (ft) with an average value of approx. 20 ft.

5. Boundary: The boundary in the shapefile represents the outline of the geographic area of the field as shown in Fig 3.3. It is defined by using coordinates that form a closed loop. A series of connected points or vertices are used to create lines and polygons, which are defined by their respective coordinates of latitude and longitude.

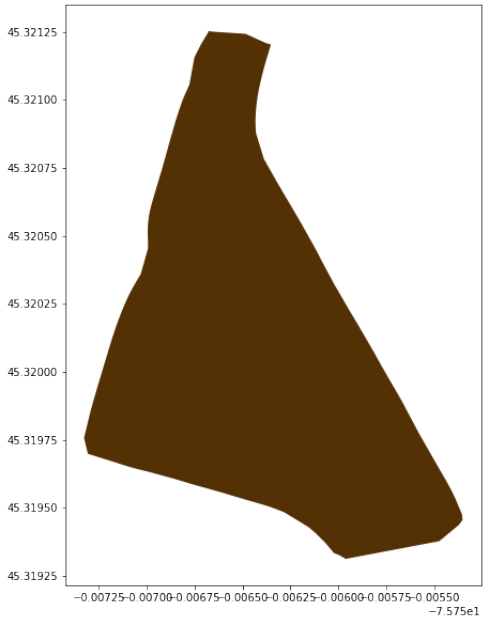

Figure 3.3: Boundary file plotted for Field 2, where x and y axis represent geometry in WGS 84.

For processing the shapefile and analysis in Python, the Geopandas [8] library was used. It extends the popular data analysis library, pandas, to enable geospatial data processing and analysis. Geopandas provides a convenient way to read, write, and manipulate

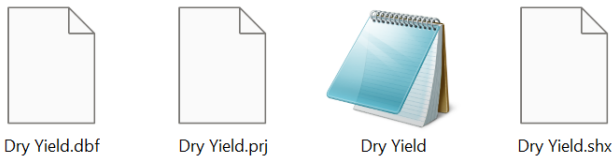

Figure 3.4: The directory housing the DryYield Shapefile contains several files essential for geospatial analysis.

geospatial data, such as shapefiles and other vector data formats, using Pandas dataframes. Another advantage is its integration with other popular Python geospatial libraries, such as Shapely, Fiona, and Pyproj that were also used in this research for analysis, therefore making it a powerful and flexible tool for geospatial analysis.

index	Name	Moisture	Units	Time	geometry
0	NaN	22.5	%	2021-11-06T23:46:13	POINT (-75.75538921365987 45.319463214439594)
1	NaN	22.5	%	2021-11-06T23:46:14	POINT (-75.75539227135475 45.3194676409676)
2	NaN	22.5	%	2021-11-06T23:46:15	POINT (-75.75539532904966 45.319472446911895)
3	NaN	22.5	%	2021-11-06T23:46:16	POINT (-75.75539928606658 45.3194781381612)
4	NaN	22.399999618530273	%	2021-11-06T23:46:17	POINT (-75.75540342294792 45.31948433529867)

Figure 3.5: Geopandas dataframe was created by reading the Moisture shapefile where each column represents an associated attribute

All the shapefiles (Width, Boundary, Dry Yield, Moisture, and Speed) were read independently using helper functions. For instance, Fig. 3.5 shows the dataframe formed after reading the Moisture shapefile where the 'Moisture' column shows the captured moisture values, 'Units' shows the units in which moisture is measured, 'Time' shows the date and time of acquisition and 'geometry' shows the point on the field using coordinates. All 5 dataframes by reading different measurements were then combined into a single dataframe for further analysis as shown in Fig 3.6.

CRS stands for (Coordinate Reference System) which is a framework used in geographic information systems (GIS) and cartography to specify locations on the Earth's surface using coordinates. Each point's geometry was transformed from Geographic 2D CRS-EPSG:4326 to Projected CRS-EPSG:32618 in order to present spatial data that depicts the Earth's surface on a two-dimensional plane. EPSG stands for "European Petroleum Survey Group" which is a standard set of codes used for identifying and defining spatial reference systems, such as coordinate systems and projections. The area of use of ESPG:32618 is between 78°W and 72°W northern hemisphere between the equator and 84 °N, onshore and offshore. This includes the region of Canada - Ontario, and Quebec. Conversion to a different EPSG is essential to ensure that geospatial data are properly aligned and can be analyzed and compared with other data sets that use different CRS. For instance, this makes the calculation of the distance between two points logical and accurate.

index	DryYield	Units	Time	geometry	Moisture	Speed	Width
0	9.692485809326172	bu/ac	2021-11-06 23:46:13	POINT (-75.75538921365987 45.319463214439594)	22.5	1.254545450210571	20.010000228881836
1	35.6683464050293	bu/ac	2021-11-06 23:46:14	POINT (-75.75539227135475 45.3194676409676)	22.5	1.227272748947144	20.010000228881836
2	80.89728546142578	bu/ac	2021-11-06 23:46:15	POINT (-75.75539532904966 45.319472446911895)	22.5	1.32272732257843	20.010000228881836
3	127.16541290283203	bu/ac	2021-11-06 23:46:16	POINT (-75.75539928606658 45.3194781381612)	22.5	1.568181872367859	20.010000228881836
4	164.94589233398438	bu/ac	2021-11-06 23:46:17	POINT (-75.7554034229479 2 45.31948433529867)	22.399999618530273	1.69772732257843	20.010000228881836

Figure 3.6: The combined dataframe where each record represents a single point in the field using geometry, and corresponding attributes including Yield, Moisture, Speed, and Width.

3.1.3 UAV Imagery

The utilization of Unmanned Aerial Systems (UAS) as platforms for sensing and/or communication represents a pioneering technology with vast potential in the field of precision agriculture. According to [177], it was primarily introduced as a low-cost alternative technique for environmental monitoring, high spatial and temporal resolution imagery acquisition, and image acquisition. As discussed in section 2.4.1, low spatial resolution, drone-related errors, limitation of covering large areas, UAV regulation, time-consuming data processing, image volume, and the absence of data from the phenological stages were some drawbacks of the existing research that are addressed and tackled in this research.

The drone imagery data generation for this research was conducted by Indro Robotics which is a leading Canadian UAV research and development company. They offer a complete package of UAV data capture that includes drones operated by experts, processing raw imagery, and stitching images to create orthomosaics. From May 2021 to October 2021, RGB and Multi-spectral Orthomosaics were captured for 6 demonstration fields, which were subsequently provided to Area X.O in external hard disk storage. There are two main layers of operation:

1. Actuation and Perception: The drone operator from InDro Robotics assists in capturing field data using a camera mounted on the drone. The imagery is initially stored within the drone's internal memory itself prior to being passed on to the subsequent layer for stitching.

2. Middleware: The drone imagery captured at Area X.O is sent to a third party for the purpose of generating Orthomosaics. Due to the substantial amount of data involved, the imagery is loaded onto hard drives that have a capacity of around 4TB each. In order to analyze the data, a manual process is required to load it into memory.

InDro Robotics captured the farm imagery during the 2021 growing season, from 26th May 2021 to 1st October 2021. The entire schedule of the drone flights is shown in the Table 3.2. The plan was for the drone to capture pictures of the corn's entire growth cycle across all the fields. The acquisition was done using battery-operated drones, and the collected data was saved on an SD card. Once the data was collected, it was sent to a third party for the creation of Orthomosaics which was done using Pix4D software. Autel EVO II drone, shown in Fig. 3.7 was mounted with an EVO 2 Gimbal camera (shown in Fig.3.8) was used to capture Red Blue Green (RGB) and Multi-spectral (MS) farm

Figure 3.7: [4]Autel Evo II Drone mounted with 8K RGB Camera

Figure 3.8: [6] Autel Evo II Gimble Camera that can record 8K video at 25 fps or 6K video at 30 fps with a data rate of up to 120 Mbps

imagery. The Multi-spectral imagery consists of 5 bands Blue, NI, Red Edge, Green, and Red. To accomplish this task, the Autel Robotics software, a mobile app, was utilized to manage the drone's movements.

Between May 26th and May 31st, 2021, the initial flight week was planned, during which data was collected on a daily basis for six days. For the first two days, only 4K images were taken, but on the third day, an upgrade was made to the EVO 2 camera so that it could capture 8K images. Following this phase of daily data gathering, the frequency was decreased to once per week starting on June 7, 2021. In the later part of the growth cycle, the Autel EVO 2 with EVO 2 gimbal camera was swapped out for a dual-payload DJI M210 drone that had two RGB cameras, namely the Mica Sense Red-Edge M and a Zenmuse X4S. The camera configuration was changed from 48MP to 20MP. Beginning on June 22, the DJI drone, equipped with both cameras, began capturing multispectral images as well as RGB images. Each flight captured imagery for approximately 45 acres of the farm at a height of 50 meters. Notably, Ground Control Points (GCPs) were not needed due to the presence of numerous recognizable structures in the AreaX.0 topology.

Time of season	Acquisition Period	Period	Image Resolution	Type	Camera Resolution (MP)
Early growing season	May 26th – May 31st	Daily	4K images	RGB	33.2, 12.0
Early growing season	June 7th – June 14th	Weekly	8K images	RGB	33.2
Early growing season	June 22nd – June 28th	Weekly	8K images	MS + RGB	17.7
Mid growing season	July 12th – July 18th	Weekly	8K images	MS + RGB	16.8
Mid growing season	July 22th – July 30th	Weekly	8K images	MS + RGB	16.8
Mid growing season	Aug 4th – Aug 10th	Weekly	8K images	MS + RGB	16.8
Mid growing season	Aug 11th – Aug 19th	Weekly	8K images	MS + RGB	16.8
Later growing season	Aug 25th – Aug 31th	Weekly	8K images	MS + RGB	16.8
Later growing season	Sep 2nd – Sep 9th	Weekly	8K images	MS + RGB	16.8
Later growing season	Sep 14th – Sep 17th	Weekly	8K images	MS + RGB	16.8
Later growing season	Sep 30th – Oct 1st	Weekly	8K images	MS + RGB	16.8

Table 3.2: The table shows the Drone Image acquisition schedule for the year 2021

3.1.4 Challenges during drone image acquisition

For some of the missions during the season, the drone ran out of charge. Due to the switching of batteries in the mid-run, the raw imagery was saved in different folders in parts. The large size of demo Field 11, made it impossible for the drones to capture the entire field in one charge. Only 50% - 80% of the field was captured with a fully charged drone. This challenge was solved by splitting the mission for Field 11 into overlapping runs and saving those into different rasters as shown in Fig. 3.9. Another challenge was ill-formed and noisy raw imagery. Some of the noisy images include (a) where a part of the drone comes into the image due to improper calibration of the camera as shown in Fig. 3.11 (b) inappropriate height of the drone when taking the pictures (c) missing capturing some part of the field, that results in a black spot of missing pixels in the orthomosaics as shown in Fig. 3.10 (d) improper alignment of the areas due to incorrect stitching by Pix4D.

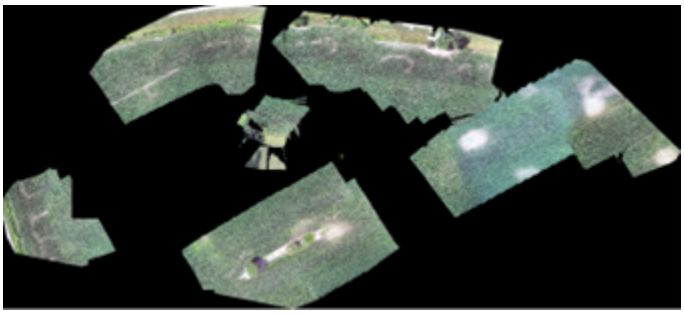

Figure 3.9: Figure shows the RGB Orthomosaic for field 11 is broken into smaller rasters

Figure 3.10: Example of an ill-formed mosaic from Field 2 that has missing pixel values

Figure 3.11: Example of an ill-formed RGB Image of Field 2 that has drone parts in it

3.2 Yield Data Processing

Once the raw data has been collected, the dry yield values must undergo post-processing before they can be used for modeling. Ensuring accurate and reliable data for decision-making in agriculture requires eliminating erroneous grain yield measurements. Achieving consistent and accurate grain measurements is a challenging task [38] due to the spatial variability and heterogeneity present in different areas of the field. Identifying outliers in grain yield measurements requires understanding the yield distribution, which is the first step. The shape of the distribution of raw yield measurements has been studied extensively, and early research suggested a normal distribution [151]. However, later studies have found that the distribution is non-normal, with both positive and negative distributions that are mostly negatively skewed, and have kurtosis values close to zero [96]. Another study conducted in Western Australia analyzed a dataset from 183 fields over a 10-year period and found that the majority of data points fell in the lower or higher ends of the yield spectrum [110]. Most research [111] on the post-processing of yield data has suggested removing errors rather than correcting them. The errors that can be caused by a variety of on-field factors are discussed below [162] [151] [110] that were removed using different techniques as a part of this study:

3.2.1 Harvest Lag time

The Harvest lag time refers to the delay between crop cutting and measurement by the yield monitor sensors. This lag time mainly varies with the model and make of the harvester [46], yield measuring sensor, and the GPS receivers [153]. One of the straight resolutions is to estimate this constant lag time from visual inspection [59] and subtract it from the measurements to get the actual value. The more advanced method involves, the use of statistical techniques like parametric modeling, surface area ratios, and segmentation. Another methodology discussed by [111] involves quantifying positional errors using the delay proposed by manufacturers and announced maximum and minimum values. This method minimizes errors by using a 20-25m error resolution for yield interpolation. The aim is to approximate the scale of grain mixing that occurs during harvesting before it reaches the measuring device. This approach reduces the variation resulting from the positional error and makes it less noticeable or significant. As part of this research, the lag time between harvesting and measurement was considered negligible and close to zero.

3.2.2 Continuous Measurement of Yield and Moisture

The continuous monitoring of moisture and yield can induce some errors and influence the accuracy of measurements. Grain flow per recording interval over a harvested area is used to calculate yield, where the harvested area is calculated as a function of the distance traveled by the harvester and the width of the swath. Impractical yield values can result from low travel distances or when a large quantity of crops moves through a system with short distances [111]. These impractical values are categorized as outliers and are removed upon analysis. Early research suggests that conventional yield cut-off filters were traditionally used to eliminate these values [150]. Interquartile ranges are a commonly used statistical method for eliminating yield outliers, but they have some limitations, such as their dependence on the distribution of the raw dataset. Advanced methods have been developed that can identify local extreme values, which do not rely on neighboring values.

A study conducted by [111], proposes a technique to detect extreme yield values based on the average of user-defined forward and backward observation count and threshold values. For instance, if the observation count value is set to 4 and the threshold to ±30% then the values that lie outside this range are marked as outliers. As part of this study, the backward and forward filter technique was employed to identify outliers in both moisture and yield values. The number of observation counts for both backward and forward filtering was set to 4 with a threshold of ±40%. In the Fig 3.12 the grey lines represent the Backward max and min values that were calculated by taking a rolling average of 4 backward values from the current value. A threshold of ±40% was used to calculate the maximum and minimum. The same strategy was applied for the forward filtering by taking the average of 4 forward values as shown in Fig 3.13. Mathematically, the Backward maximum, Backward minimum, Forward maximum, and Forward minimum threshold value for the current yield point is given by the following equations:

$$BackwardMaximumThresholdValue = 1.4 * \frac{Sum\ of\ last\ 4\ yield\ observations}{4}$$

$$BackwardMinimumThresholdValue = 0.6 * \frac{Sum\ of\ last\ 4\ yield\ observations}{4}$$

$$ForwardMaximumThresholdValue = 1.4 * \frac{Sum\ of\ next\ 4\ yield\ observations}{4}$$

$$ForwardMinimumThresholdValue = 0.6 * \frac{Sum\ of\ next\ 4\ yield\ observations}{4}$$

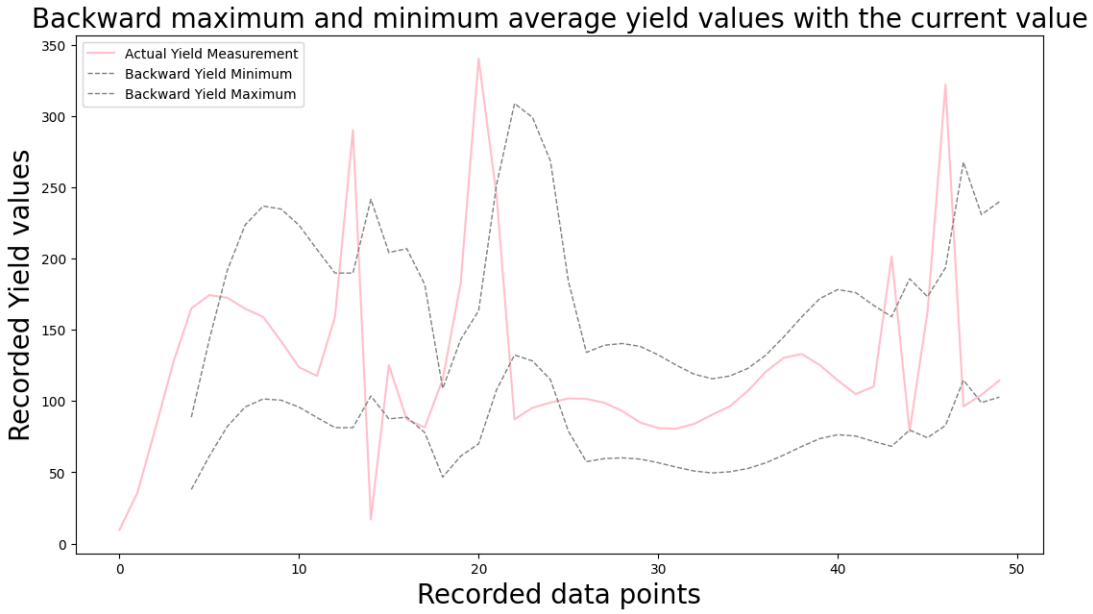

Figure 3.12: Shows fluctuations in the yield values for 50 data points from Field 2. The grey lines represent the minimum and maximum yield limits, which are based on the rolling average yields of 4 measurements taken backward from the current yield value, with a ±40% threshold. The current yield value is shown in red color.

As shown in Fig 3.13 the values between 21 and 41 meet the maximum and minimum criteria, as they fall within the gray boundaries. However, point 12 and point 13 do not meet the criteria. Point 12 has a lower yield value in the next four observations, while point 13 has a higher yield value in the same period. The same strategy is applied in the backward neighborhood technique to identify outliers. As shown in Fig 3.12 points 12 and 13 also fail the backward criteria. Such points are identified are appended to a list. Finally, an intersection of both techniques is taken to identify points that fail both the forward and backward criteria and are removed from the dataframe. There are several benefits to this algorithm, including (a) It is not influenced by GPS; (b) It uses trends of forward and backward values instead of relying on a predefined threshold value; (c) The exact same technique can be used for any different field or zone; (d) It works efficiently with large variation in yield values.

3.2.3 GPS Accuracy

GPS (Global Positioning System) is a useful tool for capturing yield data in agriculture. However, the accuracy of GPS in capturing yield data can vary depending on several factors

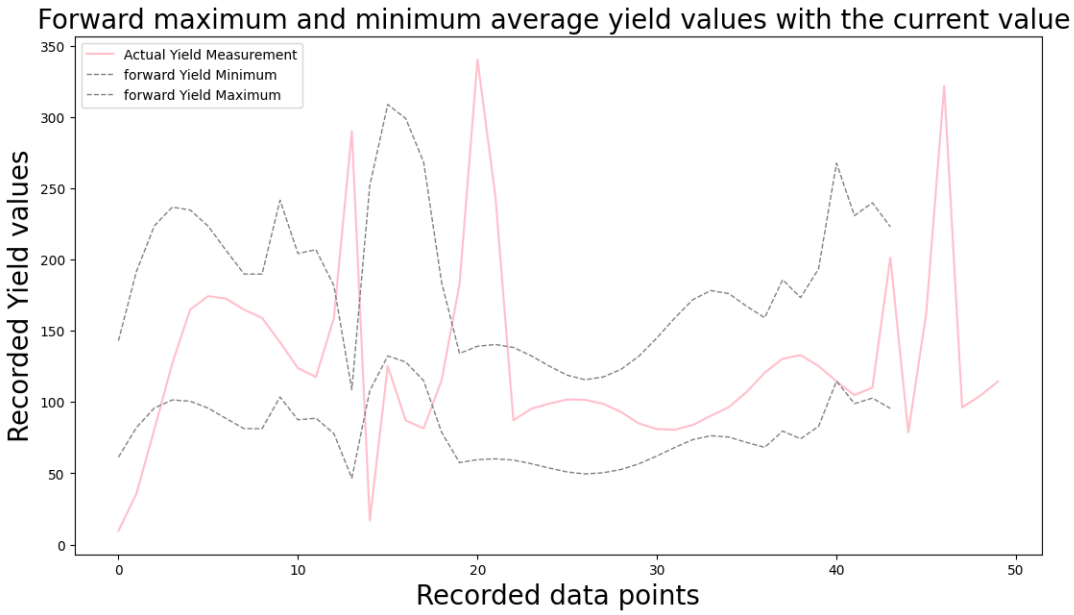

Figure 3.13: Shows fluctuations in the yield values for 50 data points from Field 2. The grey lines represent the minimum and maximum yield limits, which are based on the average yields of 4 measurements taken forward from the current yield measurement, with a ±40% threshold. The current yield value is shown in red color.

such as the quality of the GPS receiver, environmental conditions, and the type of crop being harvested. In general, GPS accuracy can range from a few centimeters to several meters. [50] outlines two types of GPS errors (a) Type 1: affects the entire dataset such that the recorded points may fall within an error threshold (b) Type 2: affects a smaller number of data points, where the harvest track is the wrong direction that gives a misleading depiction of the actual harvester path. Several methods were proposed to identify the wrong direction of the harvester. [153] proposes a routine to compare the position information between the current captured data point and the successive measurements by using the application of true north. The data captured by InDro Robotics for this research has a GPS error of approx 2m and is free from Type 2 error. Therefore, this value was considered while generating the image tiles for training.

3.2.4 Harvester Short Segments

The segment is a defined path the harvester goes through the field to collect yield as shown in Fig 3.14. [37] highlighted that the accuracy decreases with comparatively shorter segments. These short-duration segments are formed due to various reasons (a) the transport mechanism fills or empties at the start and end of every pass which is referred to as finish and fill mode error (b) turning of the harvester during a turn (c) erroneous swath width of the harvester. During these three phases, the distance traveled by the harvester is comparatively less however due to a constant flow of crops a higher yield value is recorded [111]. The duration between consecutive data points for this research was 1 second. Unique

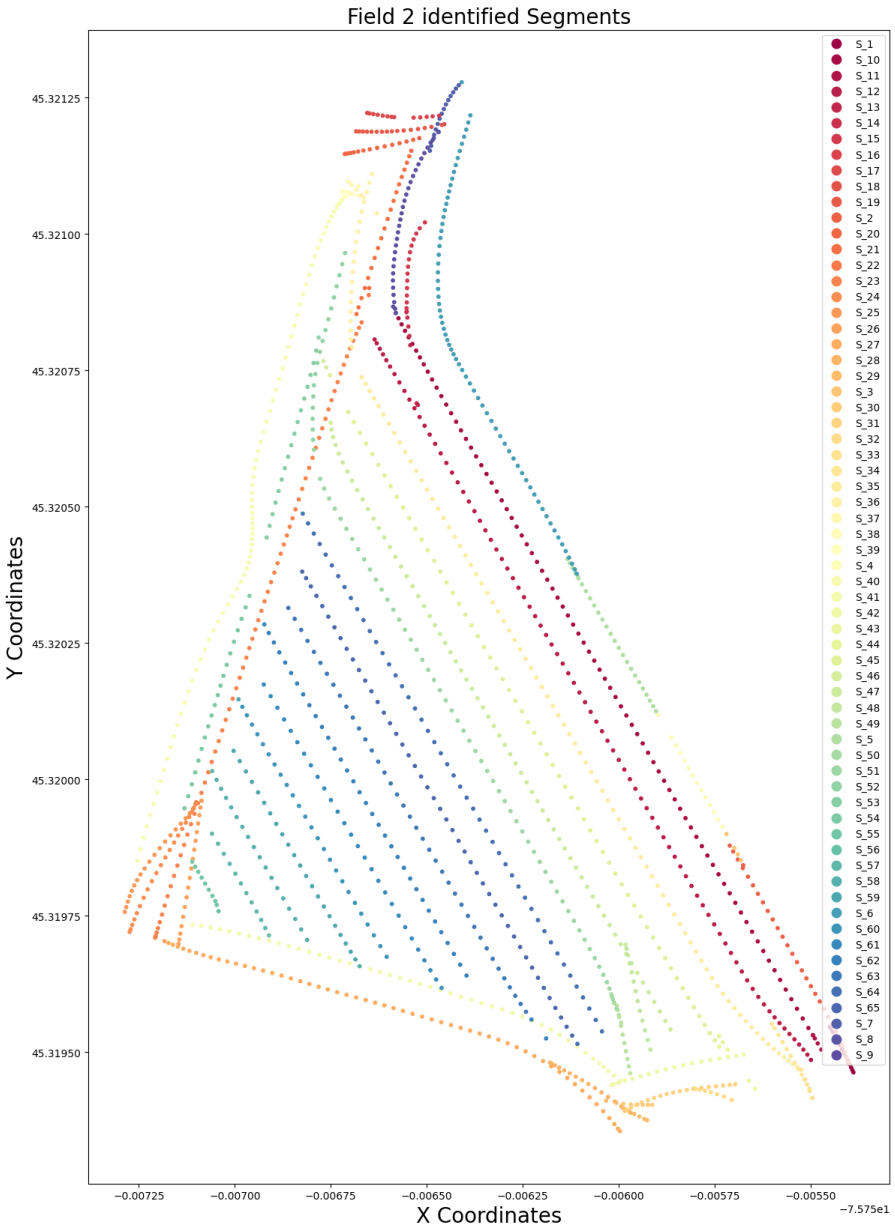

Figure 3.14: Identified segments for the demo field 2

Segments were identified based on this interval where the duration changes abruptly assuming that this is the point where the harvester changes its speed for taking a turn. The calculated distance between the starting point of each segment and the ending point was denoted as Segment length and each segment was denoted with a unique name. Then all the identified segments with their length were categorized as "Short" or "Normal" based on two different measures:

1. Distance Based: After identifying the segments, the length of each segment was calculated based on the distance between the starting and ending point of the segment. Erroneous or Short distances were identified using the Interquartile Range (IQR),

wherein values less than -1.5 IQR were identified as short segments. The identified short distance-based segments for Field 2 are shown in Fig. 3.15.

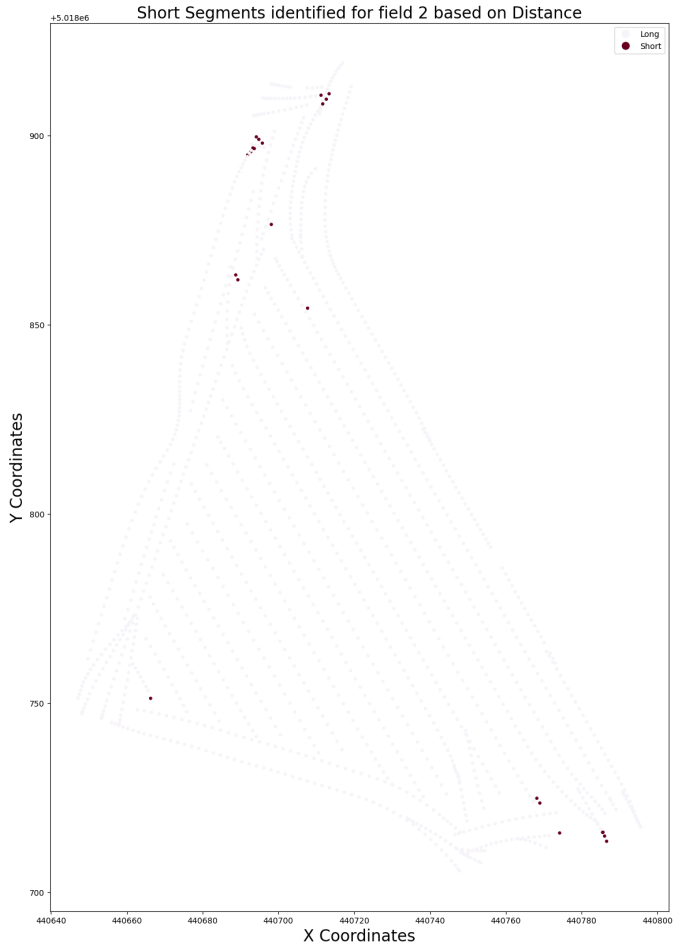

Figure 3.15: Field 2 identified long and short segments based on distance

2. Duration Based: The duration of each segment was calculated based on the time taken for the harvester to complete the segment. The total time for a particular segment was calculated by summing the identified durations between each consecutive point. Erroneous or Short duration segments were identified using Interquartile Range (IQR), wherein values less than -1.5 IQR were identified as short segments based on duration. Fig 3.16, shows identified short segments based on the duration for Field 2. 3.15.

3.2.5 Harvester Speed

Existing research [37] [135] suggests that the error percentage in yield measurement increases if the Speed of the harvester is varied. [111] highlighted that when the speed is reduced abruptly, a sudden increase in the yield value was noted. This low distance

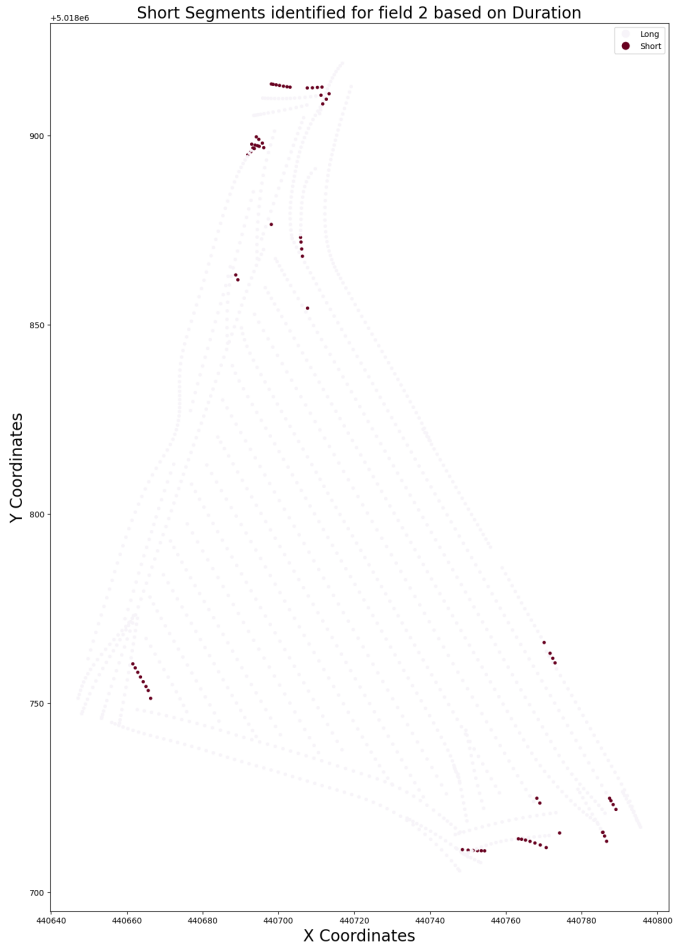

Figure 3.16: Field 2 identified long and short segments based on duration

traveled combined with a constant grain flow rate generates unexpectedly higher yield measurement. After performing a correlation analysis on the existing data, a high negative correlation value was found between the Yield value and Harvester Speed as shown in Fig 3.17.

Earlier research [47] relied on using speed-related filters, for instance, removing speed high (> 10 km/h) and low (< 1 km/h) values based on the data distribution. [111] proposed a method that identifies local yield fluctuations. This method involved the comparison of individual records on user-defined forward and backward observations with average speed values based on an arbitrary threshold value of $\pm 20\%$. This approach is similar to the backward and forward filter technique described in the section 3.2.2 to identify erroneous yield and moisture values. The observation count was set to 4, where this technique will take the average speed of forward and backward 4 speed values, and define the maximum and minimum on a threshold value of $\pm 20\%$. Mathematically the values are counted as follows:

$$BackwardMaximumSpeedThresholdValue = 1.2 * \frac{Sum\ of\ last\ 4\ speed\ observations}{4}$$

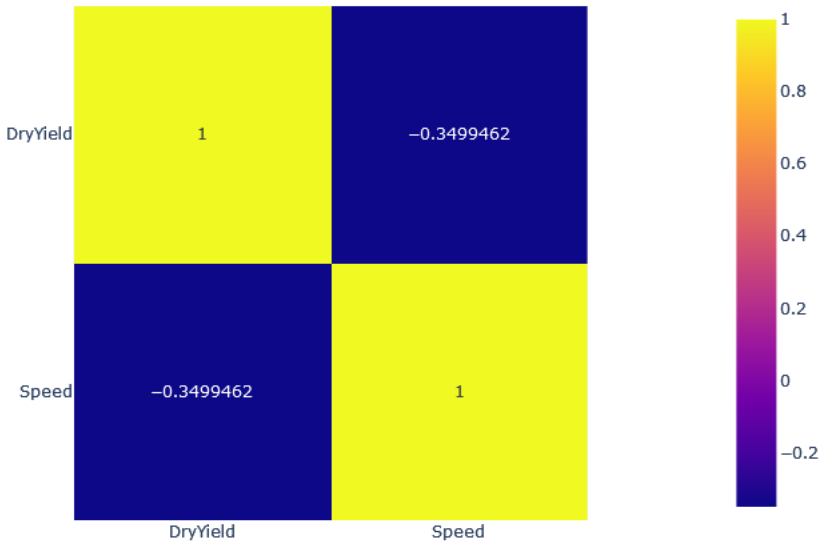

Figure 3.17: The correlation matrix between the captured Yield and Harvester Speed. A high negative correlation between the dry yield measurements and the speed of the harvester was noticed.

$$BackwardMinimumSpeedThresholdValue = 0.8 * \frac{Sum\ of\ last\ 4\ speed\ observations}{4}$$

$$ForwardMaximumSpeedThresholdValue = 1.2 * \frac{Sum\ of\ next\ 4\ speed\ observations}{4}$$

$$ForwardMinimumSpeedThresholdValue = 0.8 * \frac{Sum\ of\ next\ 4\ speed\ observations}{4}$$

Figs 3.19, 3.18, 3.20 shows speed backward and forward filters with the actual captured dry yield values respectively for 50 observations. The values that stay within the threshold values are marked as normal. For example, data points 21-40 fall within the defined threshold values for both forward and backward observations. However, at point 12, there is a sudden decreasing trend in the speed values accompanied by a drastic increase in the dry yield value. A similar trend is observed at point 42, where a sudden drop in the speed value resulted in a high dry yield measurement. These values are considered erroneous because they do not fall within the predefined threshold. In a practical scenario, achieving a uniform speed for the harvester is not possible, therefore this algorithm helps to flag sudden speed-changing scenarios that can be marked as erroneous.

3.2.6 Overlapping points

The harvester travels through several segments to harvest the field which involves cutting a designated strip of the crop in one pass. The width of this strip is determined by the

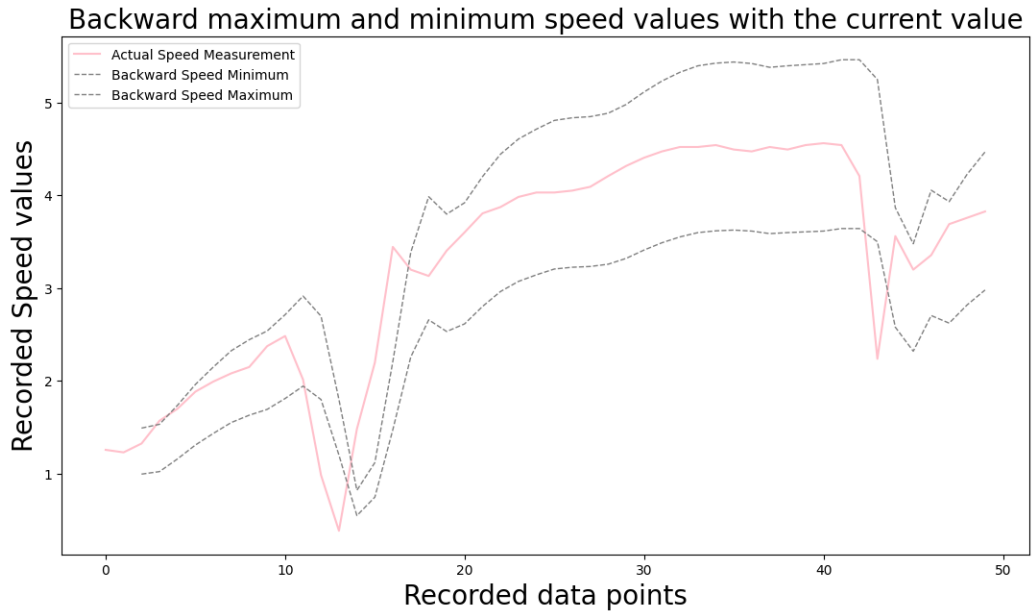

Figure 3.18: Shows fluctuations in the speed values for 50 data points. The grey lines represent the minimum and maximum speed limits, which are based on the average speed of 4 measurements taken backward from the current speed measurement, with a 20% threshold.

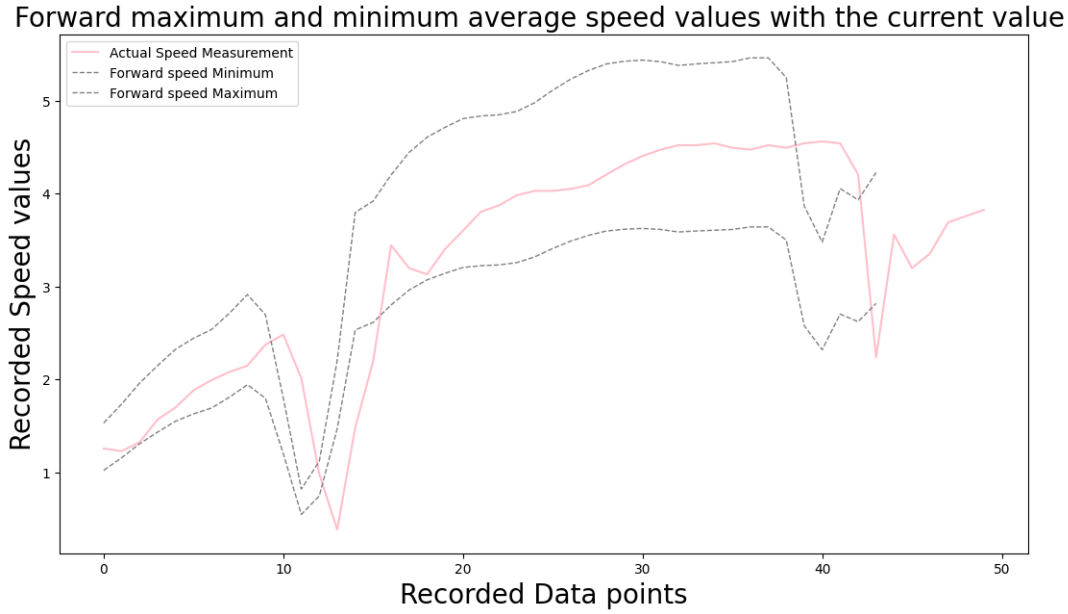

Figure 3.19: Shows fluctuations in the speed values for 50 data points. The grey lines represent the minimum and maximum speed limits, which are based on the average speed of 4 measurements taken forward from the current speed measurement, with a 20% threshold.

harvester's swath width, which is the distance between the outer edges of the harvester's

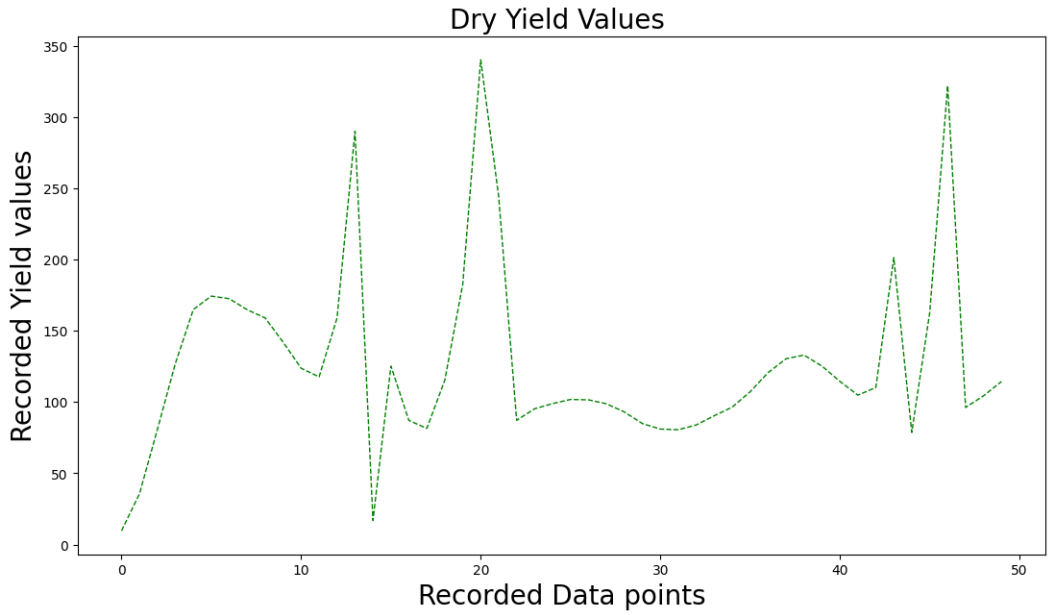

Figure 3.20: Shows fluctuations in the yield values for 50 data points. The green line represents the actual yield values

cutting mechanism. At every point, it captures the features such as location, swath width, dry yield, and moisture. After a fixed interval (1s in this research) it moves to the next location on the field and performs the same data capture. The area that the harvester already went through is marked as harvested. But in some cases, it has been seen that the harvester tries to capture another data point, in the area which already has been marked harvested. Since the same area is being harvested two times, the dry yield measurement at that location is erroneous.

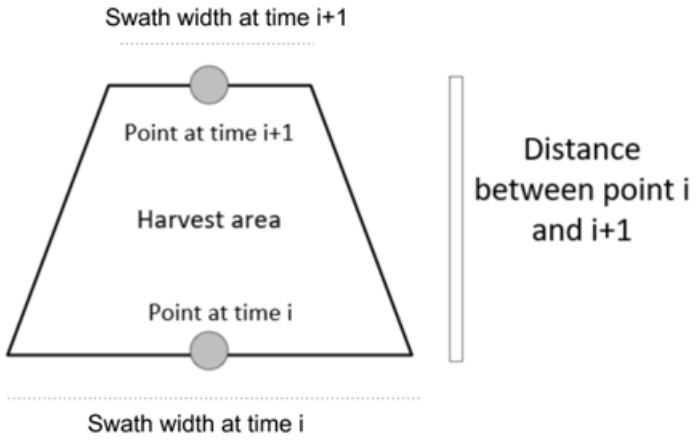

Figure 3.21: Shows harvested area in form of a trapezium, that is the area covered in a fixed interval by the swath machine between any arbitrary point i and i+1

Therefore, identifying these polygons are necessary to flag all the area that has already

been harvested, so that if a new point falls in an already harvested area it is marked as erroneous. A polygon was created around each consecutive point using geopy (python package) as shown in Fig 3.21. Two points were generated on the left and right of point "i" by forward and backward azimuth using the bearing angle and swath width as the total distance between them. Pyproj was used to calculate the bearing angle between two consecutive points. Another two points following the same strategy were created on the left and right of point "i+1". After defining these 4 points, a Geopandas polygon was created and stored against each data point in the Geopandas dataframe as shown in Fig 3.22. The polygons were generated in a sequential manner, and each new yield data point was checked to determine whether it fell within an area that had already been harvested. If a data point was found to be in a previously harvested area, it was excluded from the final dataset.

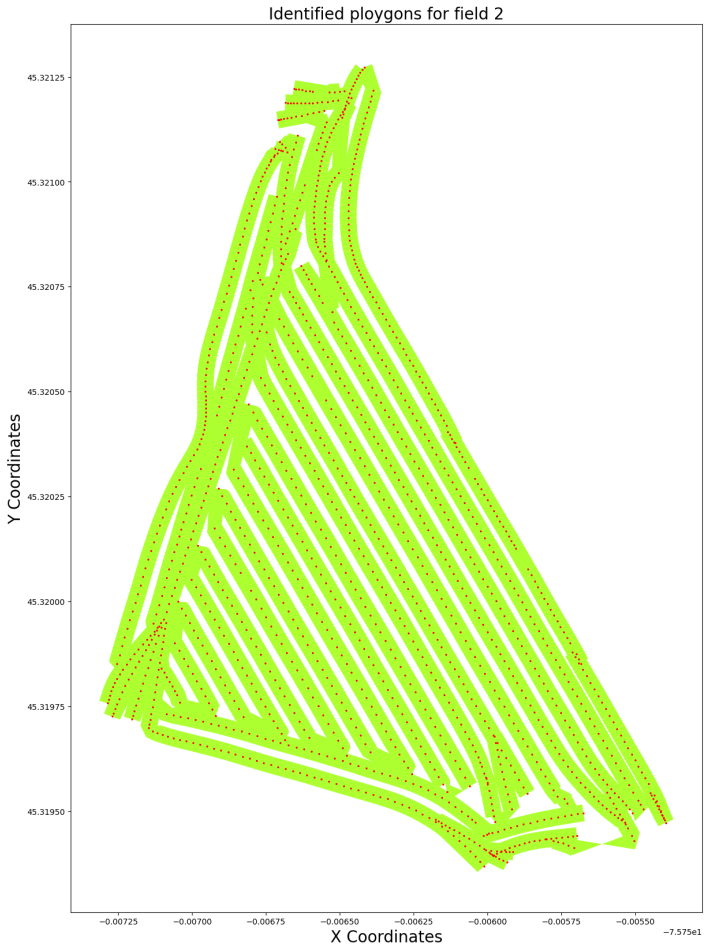

Figure 3.22: The green area represents the identified harvested area in form of a polygon and the red dot represents the actual data point that was captured by the harvester from Demo Field 2

3.2.7 Statistical Methods

The Z-score method is one of the most used methods [111] for detecting outliers in a dataset. It measures the number of standard deviations an observation is from the mean of the dataset. The Z-score method is a useful tool for identifying outliers in a dataset, but it's important to note that it can only be used for datasets that have a normal distribution. A data distribution analysis was done for the raw yield as a part of this research for all the fields as shown in Figs. 3.27 3.28 3.29. It was found that the initial raw data distribution for all the fields was not normal and showed a skewed distribution. Therefore, initially applying the Z-score method was not possible and it was applied at the end of the pipeline after the outliers were removed using the above-mentioned processing techniques.

3.2.8 Machine Learning Methods

Machine learning techniques can be used to yield outlier detection by training models to learn patterns in the data and identify those values that deviate significantly from the norm. The below machine learning-based techniques were applied to process the yield data:

3.2.8.1 The Elliptic Envelope algorithm

Elliptic Envelope algorithm or EEA It is an unsupervised machine-learning approach that is used to detect anomalies [88]. It is founded on the notion that anomalies are data points that are distant from the bulk of the data points in the feature space. Using the Mahalanobis distance metric, the technique fits an ellipse to the densest region of the data. The Mahalanobis distance calculates the distance between a point and a distribution while accounting for data covariance. The ellipse is then used to identify points that are far away from the center of the ellipse, which are considered to be anomalies. The approach requires normally distributed data and assumes normal points are dispersed in an elliptical form. Therefore, this algorithm for finding the outliers was applied at the end of the postprocessing pipeline. One of the hyperparameters supplied was contamination, which was set to 0.025. The contamination parameter governs the amount of contamination in the dataset, which is the percentage of outliers. The input dataframe consisted of two dimensions including measured Yield and Speed values to detect the combination where the speed was less but the yield was high or vice-versa. Therefore, based on these two features, the algorithm identifies the points that are anomalies. Outliers are data points that fall outside the envelope after the envelope has been fitted to the dataset. Fig. 3.23 shows identified outliers for field 2 in red color using EEA.

3.2.8.2 Isolation Forest

Isolation Forest [109] is an unsupervised machine learning system that detects outliers. The technique works by building random forests of decision trees to isolate abnormalities in the

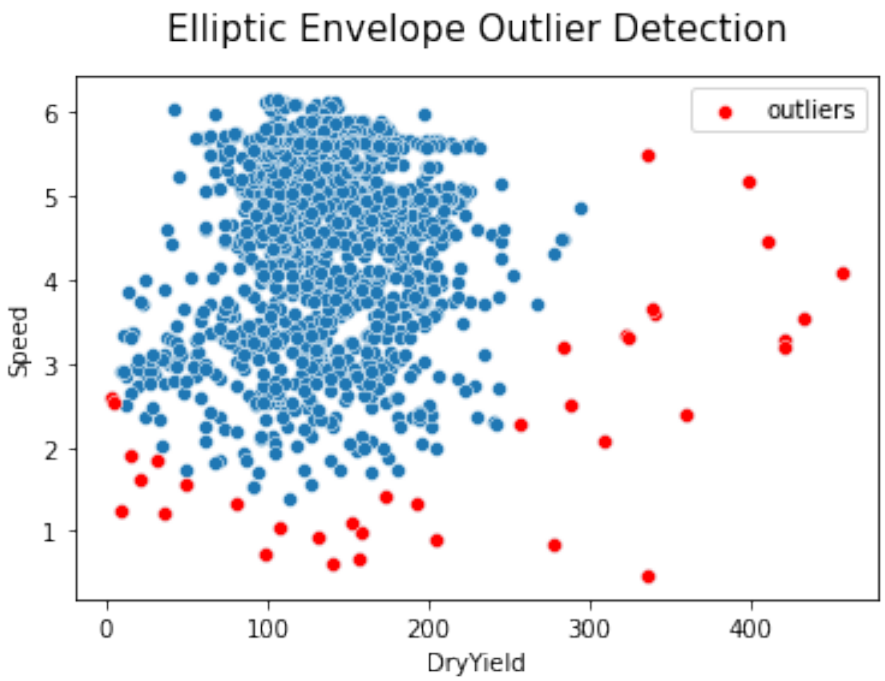

Figure 3.23: Scatter plot displaying detected outliers in red color by the Elliptic Envelope Algorithm on Field 2's data

dataset. The algorithm's main notion is that erroneous measurements are easier to identify than regular data points since they require fewer divisions in a decision tree to isolate them. The technique works by picking a random selection of data points from the dataset and then building a decision tree using each data point's random characteristic. The tree is then constructed by splitting the data recursively until all data points are isolated. The technique is continued until a forest of decision trees is formed. One of the benefits of this technique is that it can handle huge datasets with high-dimensional characteristics effectively. The parameters are similar to random forest including n_estimators, max_samples, max_features, etc. except for contamination. The contamination value was set to 0.025. Fig. 3.24 shows identified outliers for field 2 in red color detected using Isolation Forest.

3.2.8.3 Local Outlier Factor (LOF) Algorithm

The Local Outlier Factor or LOF [53] method is a density-based outlier identification tool. According to the core notion of the LOF method, outliers are data points that are separated from the rest of the dataset, i.e. they have a considerably lower density than their neighbors. The algorithm begins by computing each data point's local reachability density (LRD). A point's LRD is defined as the inverse of its k-nearest neighbors' average reachability distance. The greatest distance between two points is the distance between the points and the first point's k-nearest neighbor. The LOF of a location is then computed as the average ratio of its k-nearest neighbors' LRD to its own LRD. A location with a high LOF seems to be an outlier since its neighbors have a considerably higher density than it

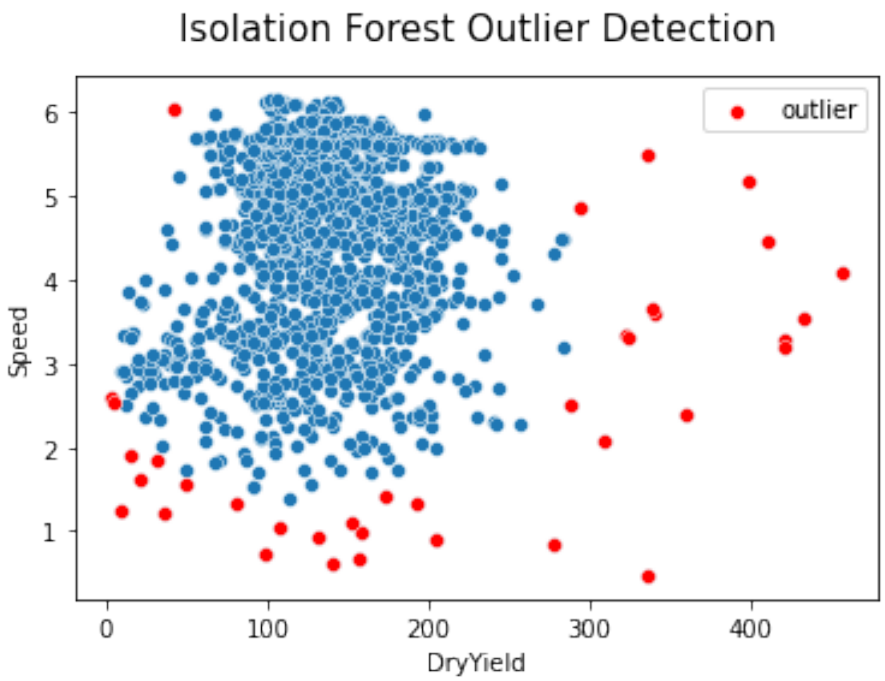

Figure 3.24: Scatter plot displaying detected outliers in red color by the Isolation Forest Algorithm on Field 2's data

does. The LOF algorithm is a robust algorithm that can handle noise and outliers in the data. It is not sensitive to the presence of small clusters or isolated data points, which can be problematic for other outlier detection algorithms. Some of the hyperparameters for this algorithm include n_neighbors, metric (distance), and contamination since this algorithm works based on the nearest neighbor principle. The contamination value was set to 0.025 which is similar to the rest of the two ML algorithms. Fig. 3.25 shows identified outliers for field 2 in red color detected using LOF.

3.2.9 Post Processing Pipeline

The entire post-processing pipeline involved several above-mentioned techniques that were applied sequentially. Several processing techniques that assumed normal distribution were applied at the end of the pipeline. Studies conducted by [151] [158], highlighted that the yield processing techniques should be carried out in a logical, sequential, and multi-level procedure with specified criteria for mistake elimination. Therefore, the processing techniques were applied in a logical order and sometimes took an intersection of the list of outliers that were flagged by different techniques such as ML algorithms.

Fig. 3.26 shows the end-to-end yield processing pipeline. The initial pipeline begins with removing the lag time, which was considered negligible as a part of this research. This is followed by applying the moisture and yield filter using the forward and backward threshold technique and then removing the GPS-related error. Next, short segments were

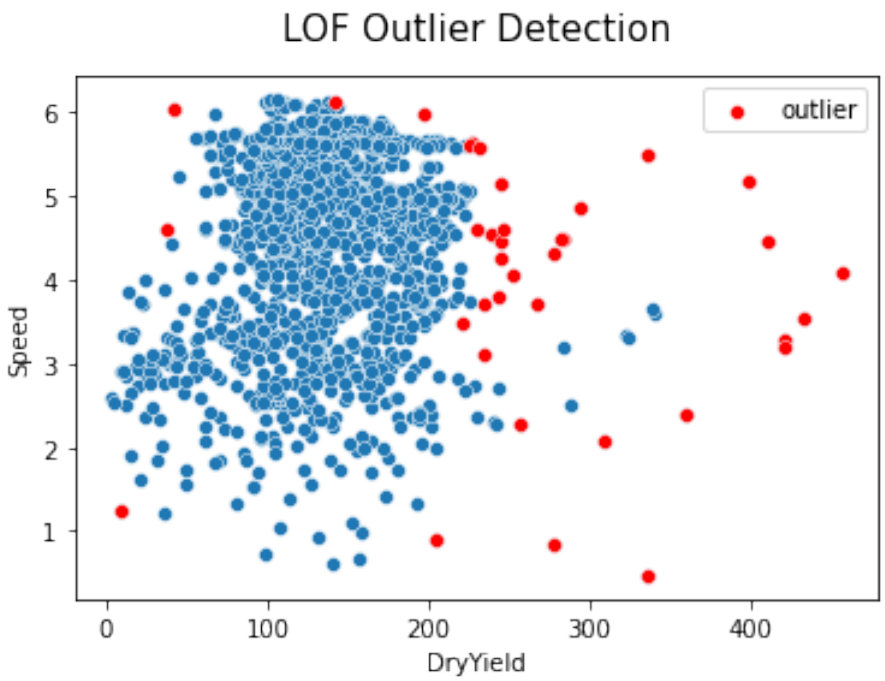

Figure 3.25: Scatter plot displaying detected outliers in red color by the LOF Algorithm on Field 2's data

identified from the field and categorized into short by duration and short by distance. The segments falling into both of these categories are flagged as outliers. Following this, the speed filter was applied using the forward and backward threshold technique with a threshold of ±20%. Next, the Harvested areas were identified sequentially, and overlapping points that fall in the area that is already marked harvested are flagged. Following this, a histogram analysis was conducted, revealing a normal distribution of yield across all fields. As a result, the statistical z-score method, which assumes a Gaussian distribution, was applied. Finally, three machine learning algorithms Elliptic Envelope, Isolation Forest, and LOF were applied and the outliers that were flagged by all these algorithms were removed. The final processed yield distribution for each field is shown in Figs 3.27, 3.28 and 3.29

3.2.10 Yield Interpolation

Yield interpolation refers to the process of estimating crop yields for specific areas or regions where crop yield data is not directly available. Crop yield interpolation is important for assessing the productivity of agricultural lands, predicting potential crop yields, and identifying areas that may require additional support or intervention. There are various methods used for crop yield interpolation, including spatial interpolation, regression analysis, and machine learning techniques. A study done by [92] highlighted that Kriging and Inverse Squared are the most commonly used interpolation methods. Kriging is a numerically superior method because it provides unbiased approximations with low variance, but

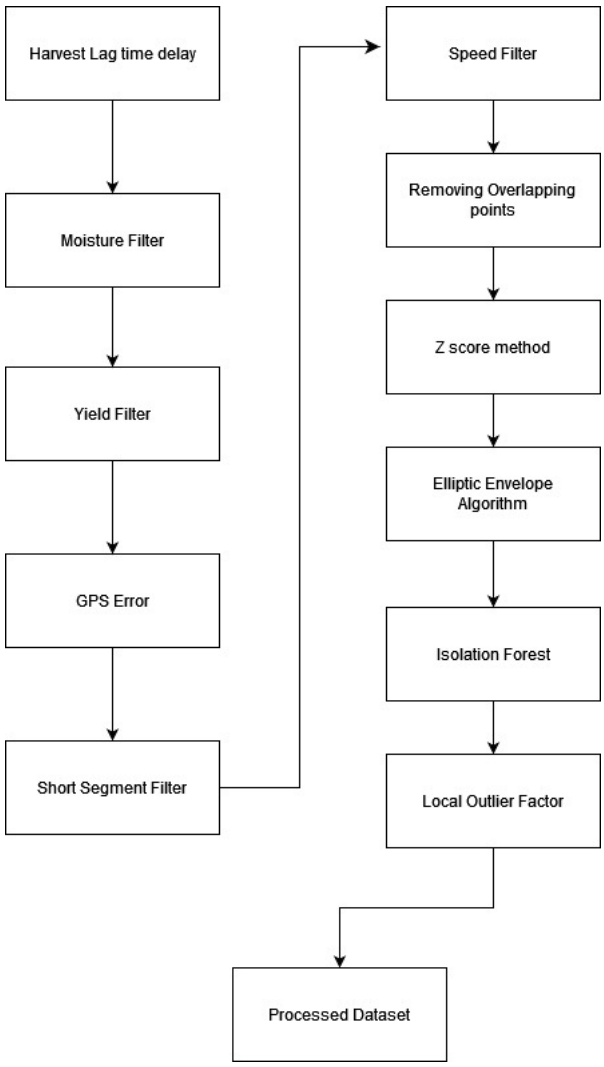

Figure 3.26: Shows the flow chart of different post-processing steps to remove erroneous yield values

it requires the creation of a semi-variogram to represent the spatial connections. Therefore, after background research on this, the kriging method was employed in this research for yield interpolation.

3.2.10.1 Kriging

Kriging is an interpolation technique in geostatistics that enables the estimation of an unknown variable's value at a specific location using nearby observations. This method proves particularly useful when there is a lack of measurements available. Kriging operates on the principle that the spatial correlation between the values of the variable being estimated decreases as the distance between locations increases. It [127] calculates the value of the variable at the desired position by taking a weighted average of the variable's values at nearby places, with the weights determined by the geographic correlation between the

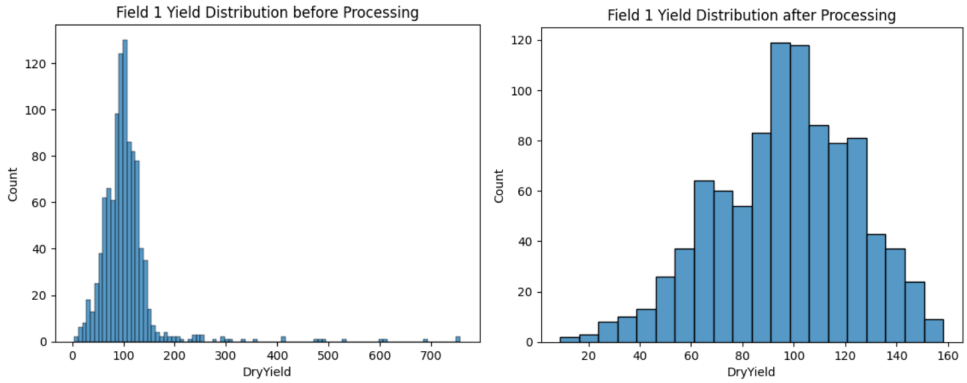

Figure 3.27: Shows the distribution of raw dry yield values before and after applying the processing techniques for Field 1 using a histogram where the X-axis represents yield values and the Y-axis represents the count

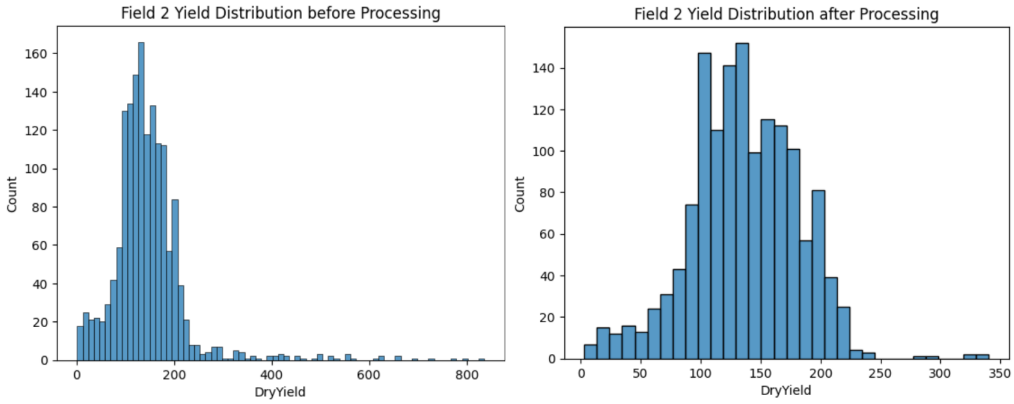

Figure 3.28: Shows the distribution of raw dry yield values before and after applying the processing techniques for Field 2 using a histogram where the X-axis represents Yield values and the Y-axis represents the count

values. Most geostatistical software requires two distinct stages for spatial interpolation: calculating and modeling/fitting the variogram for the entire area (data points), followed by kriging approximations for unsampled points in the area. Table 3.3 shows the yield data points from different fields before and after Interpolation.

3.2.10.2 Vesper

Vesper (Variogram Estimation and Spatial Prediction with Error) a Windows software for spatial interpolation was used in this study. Several researchers [134] [104] used Vesper 1.6 for creating yield maps and interpolation of digital elevations. Vesper utilizes the technique of kriging for spatial interpolation. It provides two variogram methods for conventional kriging:

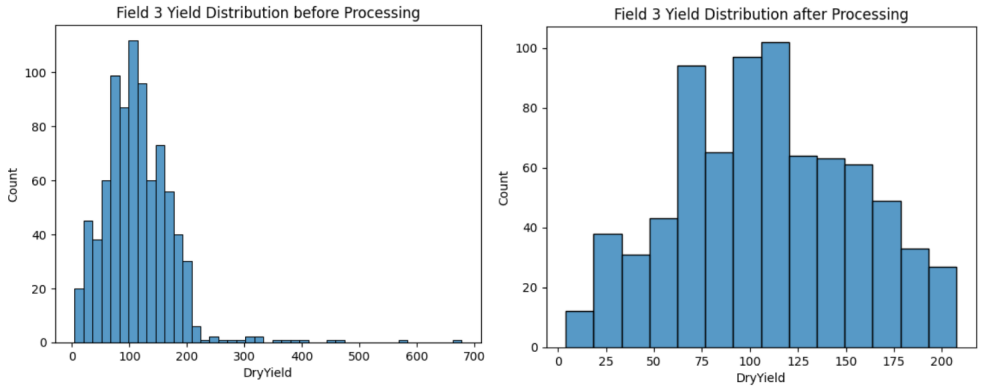

Figure 3.29: Shows the distribution of raw dry yield values before and after applying the processing techniques for Field 3 using a histogram where the X-axis represents Yield values and the Y-axis represents the count

Field	**Before Interpolation**	**After Interpolation**
Field 1	1038	2066
Field 2	1577	2313
Field 3	840	1402

Table 3.3: Number of yield measurements for fields before and after Kriging

1. Local Variograms: It provides a nonlinear least squares method to automatically fit the variograms to the data points [25]. The local variogram approach involves constructing variograms for small subsets of data points surrounding the location of interest. To use the local variogram method for yield interpolation, first, the yield data is geographically referred to using GPS or other location-based technologies. The yields at nearby sites are then used to compute a local variogram for each spot of interest. Based on the yields of nearby locations and their distances from the target location, the local variogram is applied to forecast the yield at the target location.

2. Global Variograms: Vesper provides global or whole area variograms and options to manually fit and adjust the structure. Unlike local variograms, which are constructed for each location of interest using nearby data points, a global variogram is calculated for the entire dataset, incorporating all available yield data [25]. First, it computes the semi-variance, which is a measure of the average difference between yield values at various distances apart. The semi-variance data is then fitted with a variogram model, which explains the geographic association between yield values as a function of distance. After fitting the variogram model, it can be used to interpolate yields at unsampled locations using kriging.

Another advantage of Vesper is its support for Punctual and Block Kriging. Punctual kriging, also known as ordinary kriging, is a technique that estimates the value of a variable

at an unsampled location using a statistical model based on the variable's values and their geographic association with adjacent sampled locations. Punctual kriging is based on the assumption that the variable of interest has a stable spatial distribution and that the spatial correlation structure is known and can be represented by a variogram [136]. In contrast, Block kriging is a technique used when the geographic variability of the variable of interest is non-stationary and varies across various regions or blocks of the research region. The study area is divided into blocks or sub-regions in block kriging, and each block is considered a distinct stationary domain. The variogram is computed individually for each block, and the variable of interest is predicted using Punctual kriging for each block [136]. As a part of this research, the output of the postprocessing pipeline was used as an input for kriging using Vesper. The Geopandas data frame was converted into a txt file with three columns X, Y (coordinates), and the yield value. Block kriging was selected with a block size of 9 X 9 m with 2.5 m as the distance between the interpolation. This distance of 2.5 m was calculated based on the average speed of the harvester and the average time between the captured data points. A boundary file in geostatistics is a file that specifies the boundary or extent of the research region for kriging analysis. It is used to exclude data points that are outside the field region, which can enhance kriging prediction accuracy and decrease computational time. Different variogram models including Gaussian, Exponential, Stable, Matern, Cauchy, and Stable were tried based on the field data. The exponential Variogram model showed the best RMSE value of 113.7 bu/ac as shown in Fig. 3.30 and was selected for kriging.

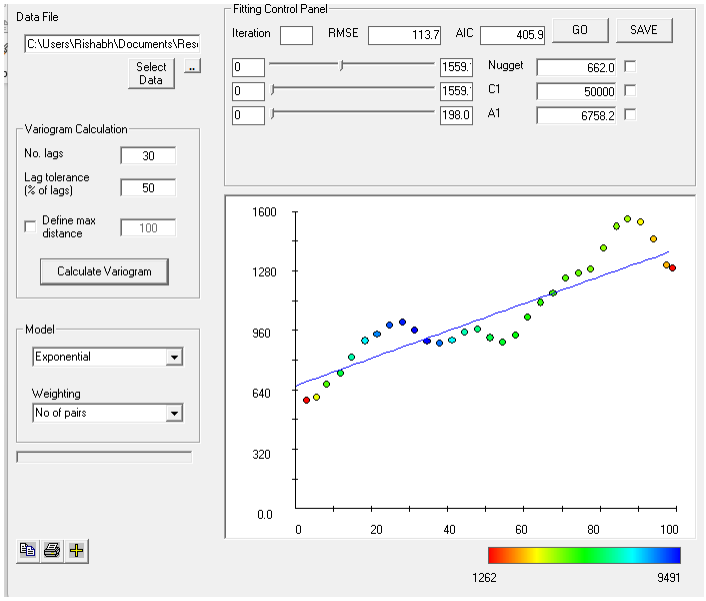

Figure 3.30: Shows calculated Exponential Variogram method for Kriging

After fitting the variogram, the kriging was initiated by the "Run Kriging Program" in Vesper. The Vesper window is shown in Fig. 3.31 shows predicted interpolation grid points using solid blue colored points based on the neighboring points. It also shows the calculated semi-variance and fitting variogram model.

The interpolated file was generated in the form of a text file. The text file was converted

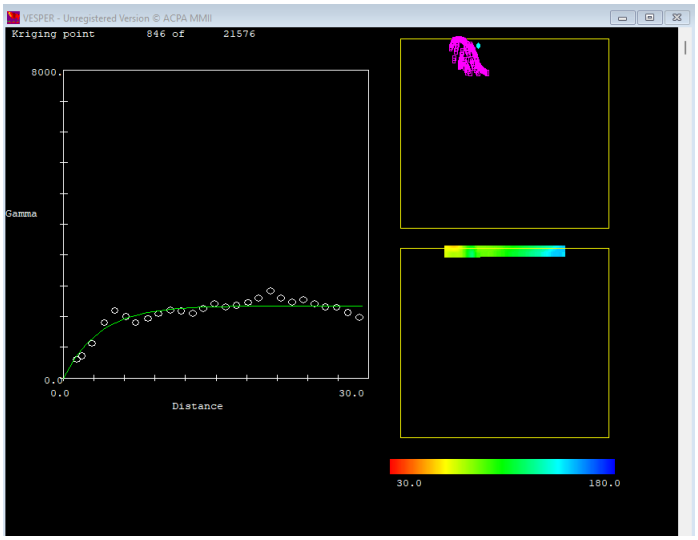

Figure 3.31: Depicts the final kriging interpolation process based on the fitted Variogram

back to Geopandas dataframe for further processing. Fig. 3.32 shows the raw yield points for Demo Field 2 and Fig. 3.33 shows interpolated yield points after kriging.

3.3 Drone Image Processing

3.3.1 Orthomosaics analysis

An orthomosaic is a detailed and accurate map-like image of the ground surface created by stitching together multiple aerial images together. The captured data consisted of approximately 255,000 aerial images in raw format and were converted to 330 Orthomosaics. The orthomosaics have file sizes ranging from 1GB to 4GB, making them difficult to open using standard image-viewing software. To overcome this, two Geographic Information System Software QGIS in Windows and GIMP software in Linux were used for viewing and analyzing the orthomosaics. TIFF (Tagged Image File Format) is a widely used file format for storing high-quality raster images. The captured orthomosaics (both RGB and MS) were stored in the Tag Image File Format (.TIFF), while the cropped RGB tiles were converted to JPG format. The pixel values for RGB TIFF files range between 0 and 255, whereas the index and reflectance files for multi-spectral orthomosaics have pixel values between 0 and 1.

3.3.2 Cropping Rasters

The average size of orthomosaics can range from 1GB – 4GB which makes it difficult to process and load into memory for further analysis. Therefore, each orthomosaic was cropped into smaller rasters that were used to extract smaller tiles that served as the training data.

Figure 3.32: Raw yield values for Demo Field 2 before interpolation

The cropping was done using a QGIS shell. QGIS (Quantum Geographic Information System) [20] is a free, open-source geographic information system (GIS) software package that allows one to create, edit, visualize, analyze, and publish geospatial data. QGIS provides a wide range of tools for working with geospatial data, including support for various data formats such as Shapefiles, GeoTIFF, and PostGIS. It also supports multiple projection systems and has built-in capabilities for spatial analysis, including distance and area calculations, buffer creation, and spatial join operations [20].

The coordinates for each slice were explicitly specified to create different sections. For example, in demo field 2, six slices were generated, as illustrated in Fig. 3.34, using the boundary coordinates listed below:

```
set f2Slice0BB=440596.7 5018969.75 440662.7 5018639.71
set f2Slice1BB=440629.7 5018969.75 440695.7 5018639.71
set f2Slice2BB=440662.7 5018969.75 440728.7 5018639.71
set f2Slice3BB=440695.7 5018969.75 440761.7 5018639.71
```

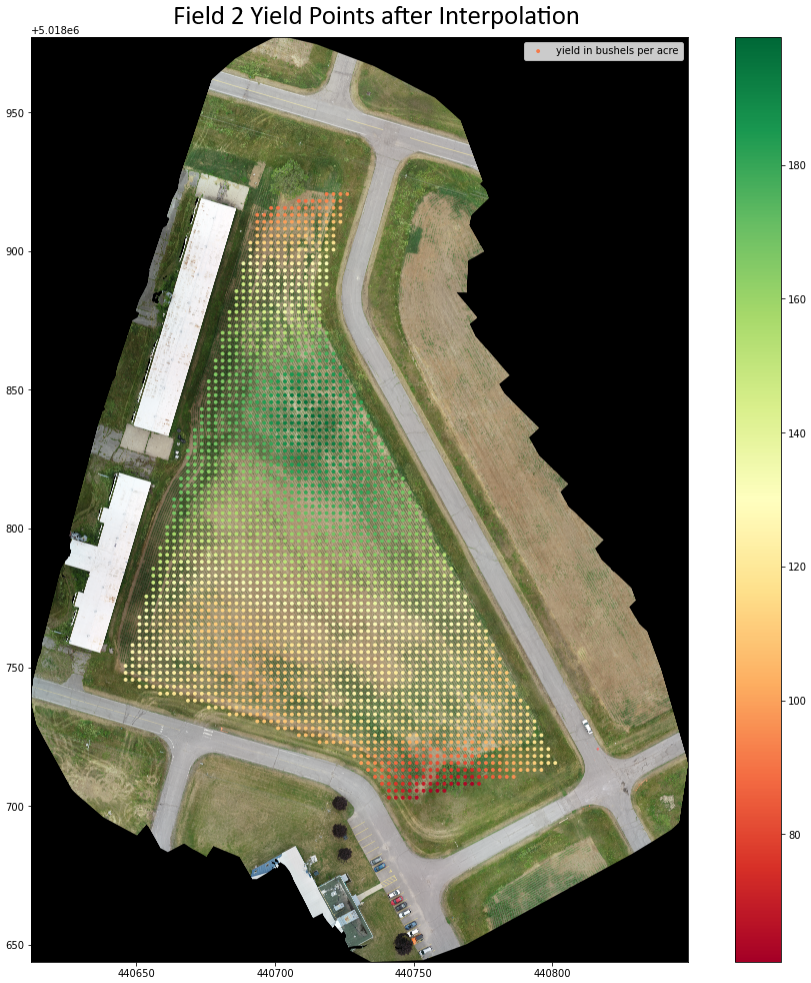

Figure 3.33: Interpolated yield values for Demo Field 2 after kriging

```
set f2Slice4BB=440728.7 5018969.75 440794.7 5018639.71
set f2Slice5BB=440761.7 5018969.75 440827.7 5018639.71
```

4 slices were created for demo field 3 specifying the coordinates as below:

```
set f3Slice0BB=440714.9 5018952 440781.2 5018650
set f3Slice1BB=440738.3 5018952 440829.1 5018670
set f3Slice2BB=440780.7 5018954 440865.1 5018667
set f3Slice3BB=440811.4 5018954 440899.8 5018663
```

`gdal_translate` a command-line tool in the Geospatial Data Abstraction Library (GDAL) that is used to translate or convert raster data from one format to another. It can be used to convert between various raster data formats, re-project raster data to different coordinate systems, and clip raster data to a specified extent. The following command was

used to crop the original orthomosaic into smaller slices by specifying the coordinates for each slice.

```
gdal_translate -projwin \%f1Slice0BB\% -of GTiff \%inputTifPath\%
    \%sliceOutDir\%0.tif
```

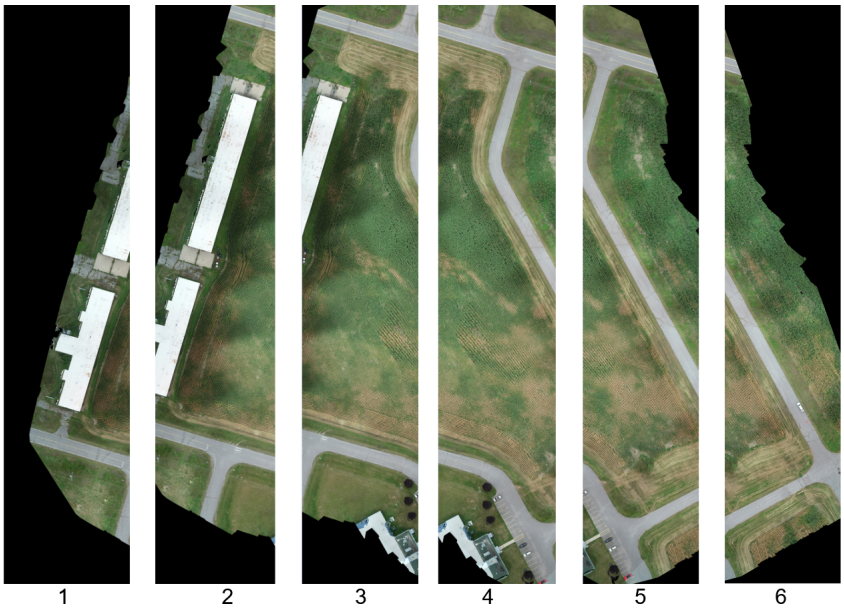

Figure 3.34: 6 Cropped slices from the field 2 orthomosaic makes the further processing computationally efficient

3.3.3 Tiles Generation

As described in Section 3.3.2, each yield data point was described by a coordinate pair. From each geospatial point, a distance of 4.5 m was chosen in all four directions to create a polygon tile of size 9 X 9 m. This also compensates for the GPS accuracy limitation that was offset by approx. 2 m [104]. The advantages of extracting tiles are listed below:

1. Spatial resolution: UAV images have a higher spatial resolution that allows for the extraction of more detailed information about the crops which is required for better yield prediction. The deep convolutional network can extract features from these tiles with the corresponding yield value as the ground truth.

2. Computationally efficient: Handling large orthomosaic images can be challenging due to the difficulties in loading multiple images into memory simultaneously. One solution to this problem is to crop the mosaic into smaller tiles, which not only helps to alleviate the memory issue but also ensures that the image resolution is not compromised. By breaking the larger mosaic into smaller parts, each tile can be treated independently and loaded into memory using Python generators.

3. Training Data generation: The tiles (independent images) serve as the training data (X), and the corresponding yield serves as the ground truth (Y) for training the model parameters as shown in the 3.35. Overall, these images were split into training, validation, and testing sets.

4. Better data management: Dividing UAV images into smaller tiles can help manage the large amount of data generated by these images in a more efficient manner. By breaking the image into smaller tiles, it becomes easier and quicker to process and analyze the data. This approach allows for more effective management of UAV imagery, which is critical for applications such as precision agriculture and crop monitoring.

Following are the steps followed to generate the tiles:

1. Conversion to the relevant coordinate reference system: All the yield data points coordinates were converted into EPSG:32618.

2. Polygon Generation: Using the geopy Python library, a polygon was generated by calculating four geopy points. Each point was determined by adding a distance of 4.5 meters in all four directions (North, East, South, and West) and an angle of 44.48 degrees, which was based on the harvester's relative position in the field during harvesting. The four points were then combined to form the polygon.

3. Identifying Relevant Rasters: To extract the tiles, the cropped rasters described in 3.3.2 were identified for all polygons. Compared to loading the entire orthomosaic in memory, loading cropped rasters is a more efficient approach. To find the respective cropped raster for each polygon, the bounds of the raster were checked. If the polygon falls within the bounds, the name of the raster was saved against the corresponding point in the geopandas dataframe.

4. Cropping: Points with the same rasters were grouped together, and the polygon was extracted from them using the Rasterio Python package. When saving the tiles, the corresponding CRS EPSG:32618 was passed as metadata.

3.4 Data Fusion

After data interpolation, it is important that the yield data and the drone images share the same spatial resolution. In the general context, data fusion involves combining data from multiple sources to achieve more accurate and cohesive information. Data fusion techniques have been applied in the context of smart farming to integrate information from remote sensing imagery and ground-based yield data, with the aim of enhancing crop management and improving yield prediction. By fusing these different sources of data, it is possible to obtain a more complete and accurate picture of crop health and growth, which can help farmers and agronomists make informed decisions about when to plant, fertilize,

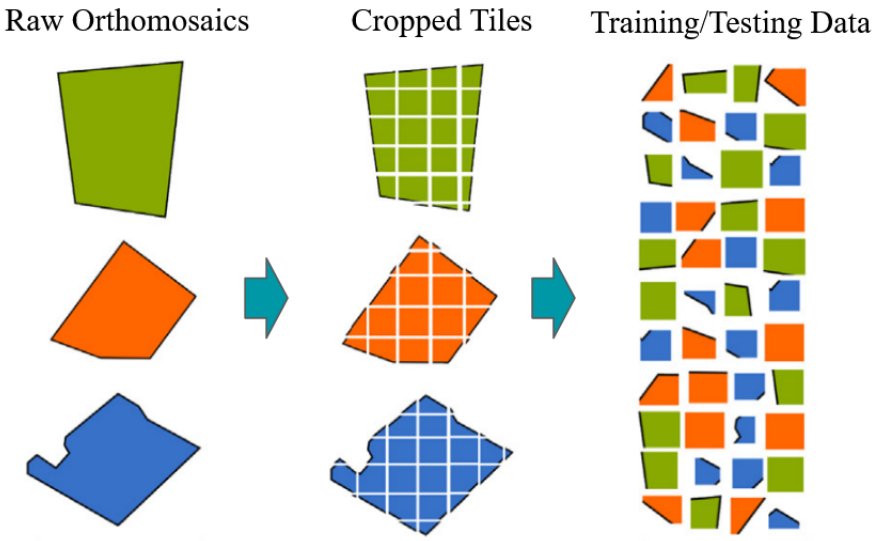

Figure 3.35: [124] As shows, each yield point on the field was converted into a polygon of size 9 X 9 m referred to as an image tile.

and harvest crops, as well as how to optimize crop yield and quality. Remote sensing imagery provides information about the health of crops, as well as the spatial distribution of vegetation and soil conditions. Ground-based yield data, on the other hand, provides direct measurements of crop yield at specific locations within a field. By fusing these two types of data, it is possible to obtain a more complete picture of the factors that affect crop growth and yield. Fusing imagery and yield data based on their spatial location can be an effective approach for creating a dataset to train machine learning models that can predict crop yield using the imagery. In this approach, the actual crop yield serves as the ground truth for calculating the loss, and the imagery is used as the training data. The ML algorithms can be applied to train models that can accurately predict crop yield based on this dataset. [104] [146] [70] used features extracted from satellite imagery, such as vegetation indices and soil moisture to predict yield. Another application of data fusion involves using spatial interpolation techniques to estimate yield at locations where no ground-based data is available based on the available yield data and remote sensing imagery. This can be useful for generating more accurate yield maps that can be used to guide crop management decisions [65] [140]. As a part of the research, these approaches were combined where yield was interpolated as the first step and then combined with the UAV imagery to have the same spatial orientation.

The Tiles extracted in section 3.3.3 were fused with the processed yield data based on the spatial coordinates. The spatial resolution refers to the coordinates of the polygon tiles, that were represented by the geometry column in the geopandas dataframe as shown in Fig 3.6. An example is shown in Fig 3.36 where the number of images generated was equivalent to the number of interpolated yield points required to cover the entire field. The fusion process was repeated for all the demo fields and the resulting tiles were saved in a folder with the corresponding field name and date. The processed dataframe consisted of

the path of the tile, with the yield data as shown in Fig 3.39. This dataframe was used as the training/testing dataset for the deep learning model.

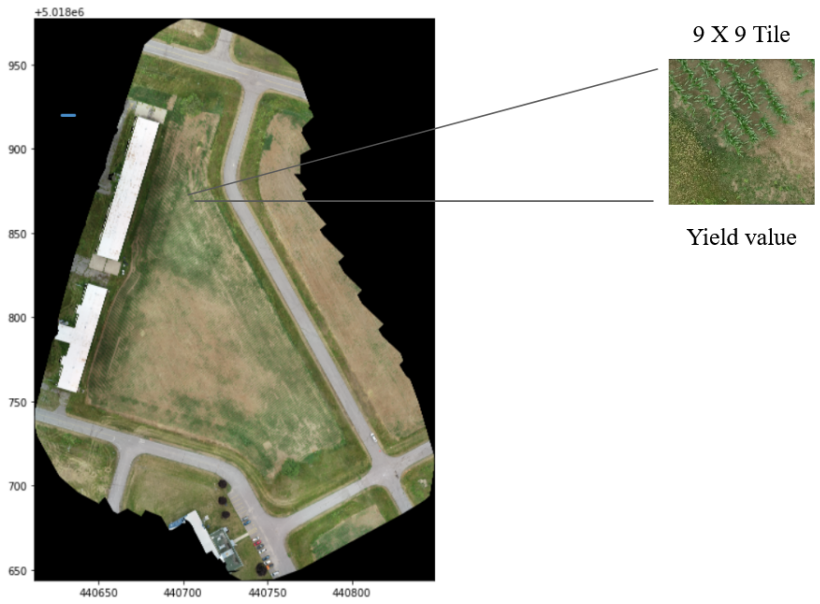

Figure 3.36: Shows a single tile of size 9 X 9 m, that represents a polygon on the field. Each tile has a single yield value that serves as the ground truth (Y)

3.5 End to End Dataset Generation Lifecycle

Fig 3.37 shows all the steps for data preparation and are described below:

1. Raw orthomosaics are processed and loaded using QGIS [20]. This process includes visually inspecting the orthomosaics that have erroneous or missing points. Mainly three orthomosaics from three growth cycles, Early growing, Mid growing and Later growing shown in Table 3.2 were considered for further analysis.

2. To optimize memory usage, the raw orthomosaics were divided into smaller rasters based on pre-defined coordinates. Since the raw orthomosaics can be as large as 3 GB, loading the corresponding cropped raster into memory during tile extraction is more efficient. The cropping process was carried out using QGIS software.

3. With the help of the Geopy Python package, square tiles were extracted based on the interpolated yield data coordinates. Each coordinate corresponds to a specific location on the field with a single yield measurement. The tiles were extracted by taking a distance of 4.5 meters in all directions from each coordinate, and these tiles served as training images for the CNN network. During training, the CNN kernels

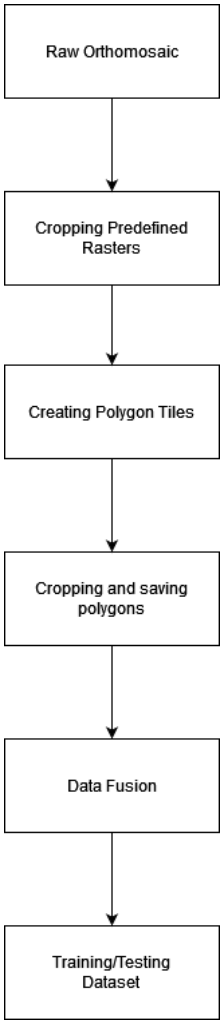

Figure 3.37: End to End Data generation lifecycle that includes initial steps from the UAV imagery processing followed by data fusion

slide over these 9 x 9 m tiles to extract spatial features and calculate loss using the ground truth yield value, which will be used to update the model weights.

4. The tiles were extracted using the rasterio package in python, and metadata such as height, width, and CRS were supplied while cropping. Each tile with a corresponding yield value was fed independently to the CNN model. Fig 3.38 shows the extracted tiles were saved based on their index in the geopandas dataframe making it easier during the building the input dataset class.

5. Finally, the geopandas dataframe was saved, where each data point contains the image path and the corresponding yield value as shown in Fig 3.39.

Figure 3.38: The tiles were saved in JPEG format and labeled with an index that corresponds to the respective entry in the GeoPandas dataframe.

	DryYield	image_path
0	91.58132	/content/drive/MyDrive/Research/geospatial/Are...
1	94.52186	/content/drive/MyDrive/Research/geospatial/Are...
2	95.08619	/content/drive/MyDrive/Research/geospatial/Are...
3	95.62759	/content/drive/MyDrive/Research/geospatial/Are...
4	88.66220	/content/drive/MyDrive/Research/geospatial/Are...

Figure 3.39: The final Geopandas Dataframe after Data Fusion

3.6 Model Building

Pytorch Framework was used for building and testing models as a part of this research. PyTorch [18] is an open-source machine learning library developed by Facebook's AI Research lab (FAIR). It is based on the popular Torch library and is primarily used for building deep neural networks. PyTorch provides a number of features that make it easy to build and train neural networks, such as automatic differentiation, dynamic computational graphs, and GPU acceleration.

PyTorch provides two main features: Tensor Computation and Deep Learning. Tensor computation involves creating and manipulating tensors, which are similar to arrays or matrices. PyTorch provides a number of functions for creating and manipulating tensors, as well as performing operations on them, such as addition, subtraction, and matrix multiplication [18]. It also supports dynamic computation graphs, which allows for more flexible and efficient training of neural networks. With dynamic computation graphs, you can change the structure of your network on the fly during training, which can be useful for implementing complex architectures.

Following are steps that were followed during this research to develop deep learning models in PyTorch: [19]

3.6.1 Importing the necessary libraries

The first step is to import the PyTorch library and other libraries required. As a part of this research following main libraries were used and imported:

1. Rasterio: It reads and writes Geo-tiff formats and provides a python API that is Numpy array-based and GeoJSON.
2. Albumentations: It is a Python library that offers fast and optimal image augmentations.
3. geopandas: It extends the data types used by pandas and allows spatial operations on geospatial data.
4. PIL: It is a Python Imaging Library and offers image processing capabilities.
5. CV2: It is a part of OpenCV (Open Source Computer Vision) and provides features such as image processing, object detection, feature detection, and description algorithms.
6. Sklearn: scikit-learn is a Python library that provides a set of tools for machine learning and statistical modeling. It is built on top of other popular Python libraries such as NumPy, SciPy, and matplotlib. Some of the key features include data normalization, supervised/unsupervised algorithms, model evaluation, and feature selection.
7. Torch: Torch is a Python library that provides a set of tools for scientific computing, including numerical operations and linear algebra. It is designed to work with multi-dimensional arrays, also known as tensors. Torch is the backbone of PyTorch, and many of the fundamental building blocks of PyTorch are implemented in torch.
8. Matplotlib: It is a Python library for creating visualizations such as charts, graphs, histograms, and other 2D plots. It provides a set of tools for creating plots in a variety of formats and styles.
9. Glob: It is a Python module that is used for file searching and pattern matching. It provides a way to search for file paths that match a specified pattern, similar to the Unix shell wildcard syntax.
10. Seaborn: It is a data visualization library that is built on top of matplotlib. It provides a higher-level interface for creating statistical graphics such as heatmaps, histograms, scatterplots, and more.

3.6.2 Data Preparation

Data Preparation is an important step in building any machine-learning model in PyTorch. This involves several steps, such as loading the data, splitting it into training and testing sets, normalizing or standardizing the data, and transforming it into tensors that can be

used by PyTorch models. The "torch.utils.data" module provides several utility classes and functions for working with datasets in PyTorch.

Some of the utility classes that were while loading the data are below [17]:

1. Dataset: It is an abstract class representing a dataset. It provides an interface for accessing individual data samples and their labels. The dataset stores the samples (x) and the corresponding labels (y).

2. DataLoader: It is a class for loading data from a dataset. It provides an interface for iterating over the data in batches, shuffling the data, and loading data in parallel using multiple workers.

3. Subset: It is a class for creating a subset of a dataset. It allows you to select a subset of the data by specifying the indices of the samples to include.

4. ConcatDataset: It is used for concatenating multiple datasets into a single dataset. It allows to combine multiple datasets into a single dataset, which can be useful for training on heterogeneous datasets.

5. RandomSampler: It provides functionality for sampling data randomly from a dataset. It provides an interface for sampling data randomly with replacement or without replacement.

6. WeightedRandomSampler: It works the same as RandomSampler but the data is sampled using weights. It provides an interface for sampling data randomly with replacement or without replacement, where each sample has a specified weight.

As a part of this research custom dataset was created by subclassing the "Dataset" class [17]. While writing the custom dataset three functions "__init__", "__len__", and "__getitem__" were implemented. As we are dealing with images (X) and yield measurement (Y), the dataloader generates a batch of images with yield measurements. The "__init__" function takes the processed geopandas dataframe as an input parameter and saves it into a member variable. This function also takes the transform dictionary that is used for performing data augmentation. The "__len__" function returns the length of the dataframe and gives the total number of data points. "__load_image" a helper function provides the functionality to load, resize, normalize images and convert to float64 format using the CV2 library. Another function "__getitem__" generates the final processed image and yield value pairs in batches.

In the case of the 2D CNN model, it reads a single image using the "__load_image" helper function from the path specified in the dataframe as shown in Fig 3.39. However, in the case of 3D CNN, it reads multiple images (2 or 3) from different phenological stages and concatenates them into a single tensor. The shape of the image sensor supported in PyTorch is $[N, C, H, W]$, where N = Batch size (number of images per batch), C = number of channels, H = height of image, W = Width of image. Therefore, using the transpose tensor function the image shape is changed from the default (H, W, C) to (C, H, W).

After building the dataset, an iterable was created using the DataLoader class. In Python, an iterable is an object that can be looped over. The DataLoader loads the data in parallel from the disk, shuffles, splits it into batches and loads it into the GPU. During training/testing the dataLoader is iterated for generating batches of images and target yield values. It generates and loads images into the batch for each iteration of training, thereby conserving memory by only loading the required data for that iteration, and subsequently, the next batch is loaded during the next iteration. The data source for this research comes from three demo fields, Field 1, Field 2, and Field 3 as shown in Fig 3.2 where each field was divided into smaller tiles of size 9 X 9 m, where each tile shows a particular part of the field.

3.6.3 Model Architecture

Model architecture refers to the structure of a neural network model. This includes the number and types of layers, the number of neurons or filters in each layer, the activation functions, and how the layers are interconnected. The architecture of a model determines its ability to learn and solve a particular problem. In PyTorch, the model architecture is defined using the "nn" module which provides a set of pre-defined layers and functions to build neural networks. These layers can be combined to create complex architectures, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformer networks.

There are two functions "__init__", and "__forward__",that are mandatory to be defined when defining a custom model architecture [18]. The "__init__" function initializes the layers of the network whereas the "__forward__" method defines the forward pass through the network, where the input (x) passes through the input layer, hidden layers and then passing the resulting hidden output through the output layer. Overall, this method specifies how input data is transformed through the different layers that are defined in the "__init__" method of the model during a forward pass. During training, the forward method is called multiple times, once for each batch of input data. The output of the forward method is used to compute the loss between the model's predictions and the true labels, and this loss is then used to update the parameters of the model during backpropagation.

The model architectures used in this research are described below:

3.6.3.1 Convolution Neural Networks with 2D Kernels

Convolution Neural Networks or CNNs are designed to handle large input data, such as images, and are able to automatically extract features from the input data through the use of convolutional layers. 2D CNNs are designed to handle 2D image data, which has width and height dimensions but no depth dimension. Therefore only the spatial relationships (height, width) were captured using this class of CNNs. Since the input dataset has images from 3 fields and 3 phenological stages, this type of model was used to find which field and phenological stage best correlates with the final yield. In 2D CNN, convolutional filters slide over the 2D image data to extract 2D spatial features, such as edges, corners, and

textures. The input is a three-dimensional tensor of RGB tile (height, width, channel), that is fed to the convolution layer where the kernel moves in two dimensions (height and width) to extract spatial features as shown in Fig 3.40.

The input dimension of the image was 512 X 512 pixels with 3 channels (RGB). The input image dimension was transposed to (3, 512, 512) to make it compatible with the model. There were a series of experiments done by resizing the image dimension to get the best performance which is discussed in Section 4.4.3. In the model's first layer of Conv2D, the input channel was set to 3 for RGB, the output channel was set to 64, and the kernel size was set to 3. As a result, the 3X3 kernel will move over the image with a stride of 1 pixel. The kernel will move both in the horizontal and vertical directions to extract the spatial features from the tile. The activation function was set to ReLU which adds the non-linearity and has been widely used in CNNs [66]. The next layer is MaxPool2D, which reduces the image dimensions by half with a stride value of 2. The MaxPool layer downsamples the feature maps produced by the convolutional layer, reducing their spatial dimensions while retaining the most important features. Following this, a dropout layer was used to add regularization where the dropout fraction was set to 0.2. The dropout layer randomly drops out a fraction of the neurons in the layer, during each training iteration, which helps to prevent the neurons from becoming too specialized to the training data. The model includes another block of 2D CNN Layer, where the input channel was set to 64 and the output channel was set to 8. This was followed by another ReLU activation function and a MaxPool2Dlayer. The overall architecture as shown in Fig 3.41 contains 2 convolution blocks that perform the feature extraction. The 3D extracted feature tensor after convolutional operations are flattened to create a single dimension vector. Three linear layers are added in a linear block where the first layer takes the flattened vector from the previous layer and reduces it to a size of 128. This is followed by two more linear layers that further reduce the dimension to 32 and finally to 1. These linear layers are fully connected layers that consist of an input layer followed by 2 hidden layers of 128 and 32 neurons respectively. Finally, the output layer is a single neuron where the loss is calculated. In this research, adding more CNN blocks to the model resulted in poor performance. This is because increasing the depth of the CNN blocks can lead to the extraction of more complex and abstract features from the input data, which may not be relevant for predicting yield.

3.6.3.2 Convolution Neural Networks with 3D Kernels - 2 phenological stages

The convolutional neural defined in the previous section 3.6.3.1 only captures the spatial relationships that exist in a single tile to predict the yield values. The proposed CNN with 3D kernels in this section stacks image tiles of the same area from two different phenological stages. Stacking the images from two different growth stages captures the temporal relationships and helps the model to capture how the growth of crops changes over time. The existing CNN network with 2D convolutional operation fails to capture this relationship since the kernel moves only in two dimensions i.e. horizontally and vertically. 3D CNN has convolutional layers with kernels that slide over the data in three dimensions, therefore, capturing the spatial and temporal relationships. The input data contains field

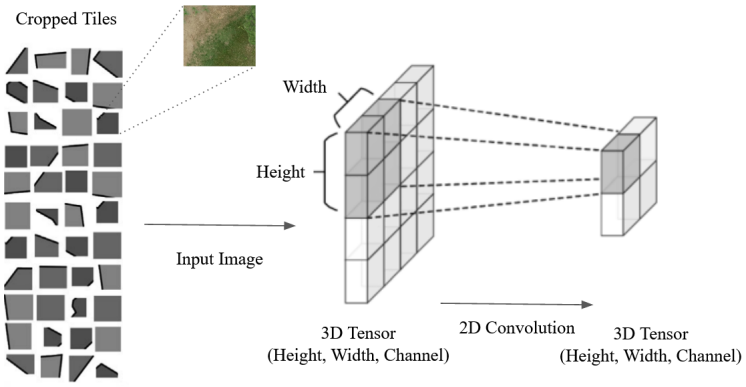

Figure 3.40: Shows the 2D CNN Convolution operation where the input is a 3D Tensor (Height, Width, Channel). The input image represents a portion of the field from one of the growing days.

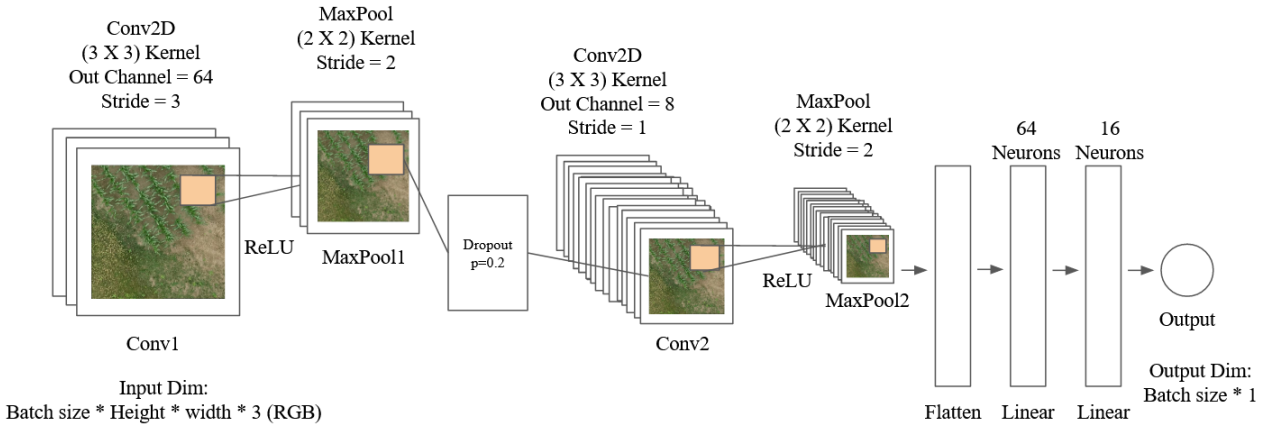

Figure 3.41: The model architecture of CNN with 2D filters consists of 2 convolution blocks, followed by fully connected layers. The input is RGB image tile of dimension [512 X 512 X 3].

images from 3 different phenological stages as shown in Table 4.1. Therefore, the input was grouped into 2 stages at a time. For instance, Images from the Early growing season 20^{th} of June, and Mid growing season 18^{th} of July were stacked together in depth. Experiments were done, by stacking images from Early-Late, Mid-Late, and Early-Late to find which growth pair gives the best prediction performance and results are discussed in the Section 4.4.1.2.

As shown in Fig 3.42, the input to the model is a 4-dimensional tensor (height, width, channel, temporal depth) where height and width represent the spatial features, channel represents three different channels Red, Green, Blue, and temporal depth represents the images from two different phenological stages. Hence, the output feature maps from con-

volutional operations preserve both the spatial and temporal relationships. The Overall Architecture is shown in Fig. 3.43. The input is of shape (512, 512, 3, 2), where (512,512) represents the height and width of the tile, the third dimension (3) represents Red, Green, and Blue, and the fourth dimension (2) represents the two paired phenological stages. The architecture consists of 2 sequential convolutional blocks and 2 fully connected layers, totaling 4 layers. In the initial convolutional block, a Conv3D layer is utilized with a kernel size of (2,2,2), padding of (2,2,2), and the output channel set to 64. This is followed by a MaxPool3D layer with the Kernel size of (2,2,2) that reduces the first two dimensions by half. A dropout layer with a drop fraction of 0.2 is applied to avoid overfitting. This follows another convolution block where the first layer is again a Conv3D layer where a kernel size of (2,2,2), and padding of (2,2,2) is applied with the output channel set to 8. During the model compilation process, it was observed that reducing the padding size below 2 led to an issue. This is because the convolutional layer decreases the spatial dimension and increases the number of channels, causing the output to shrink to zero. Hence, to prevent this, a padding of (2,2,2) was also used for the second convolutional block. This layer is followed by another MaxPool3D layer with the Kernel size of (2,2,2). Finally, the output feature-maps from the Convolution operations are flattened to a single-dimension vector. The output vector is fed to two sequential linear layers of size 128, and 32 respectively with ReLU activation in between. Finally, the output size is set to 1, where the loss is calculated.

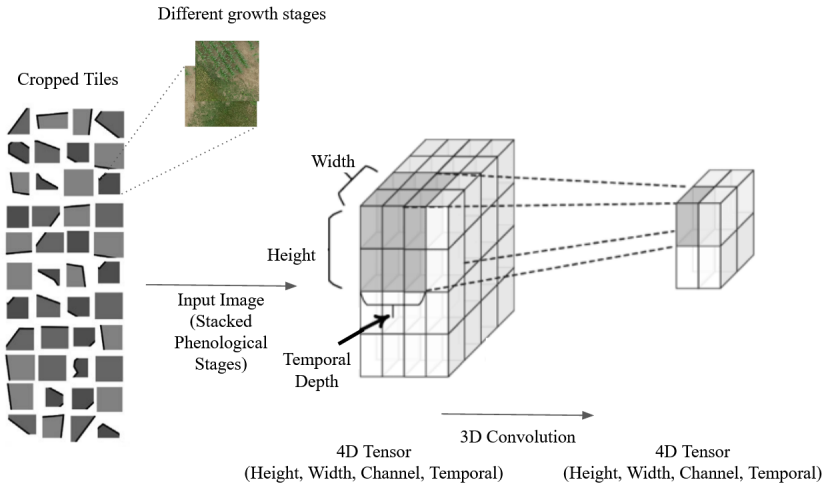

Figure 3.42: Shows the 3D CNN Convolution operation where the input is a 4D Tensor (Height, Width, Channel, Temporal Depth). The temporal depth represents two phenological images stacked together in a single tensor depthwise

3.6.3.3 Convolution Neural Networks with 3D Kernels - 3 phenological stages

The 3D Convolutional Neural Network model in the previous section combines images from two different phenological stages depthwise into a single tensor. 3D CNN proposed in this

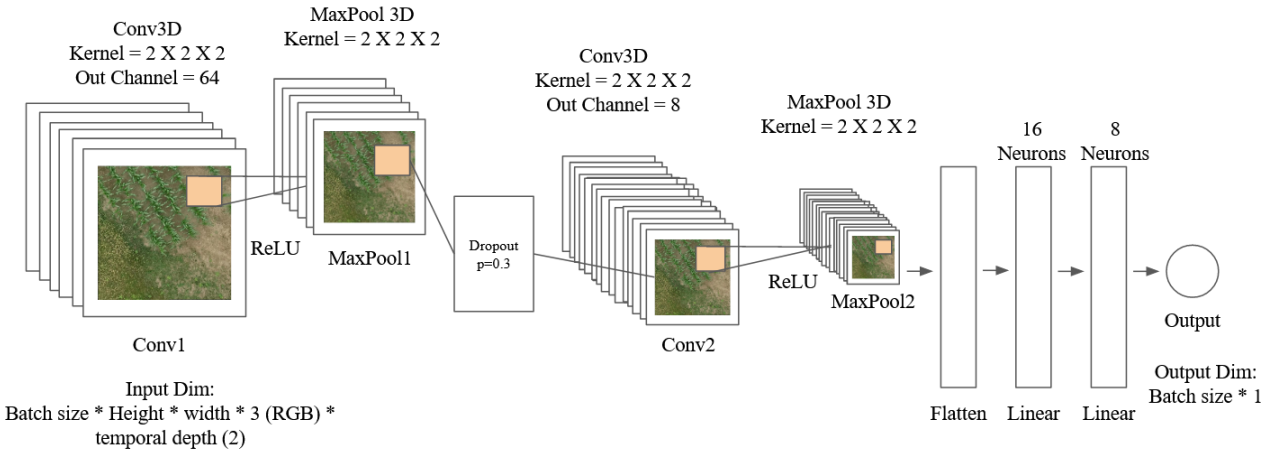

Figure 3.43: The model architecture of CNN with 3D filters where pair of phenological stages are added as temporal dimension consists of 2 convolution blocks, followed by fully connected layers. The input is RGB image tile of dimension [512 X 512 X 3 X 2], where the last dimension is the temporal dimension.

section takes input where all three phenological images from early, mid, and later growing seasons were stacked together.

As shown in Fig 3.42 the input is again a four-dimensional tensor that combines tiles from the Early growing season 20th June and Mid growing season 18th July, and Later growing 29th August 4.1. Preserving the temporal relationship for the entire growing season can be achieved by combining images from all phenological stages. Therefore, the 3D CNN proposed in this section has access to the complete crop growth stages. The overall architecture is shown in Fig 3.44. The input is a 4D tensor of shape (512, 512, 3, 3), where the first two dimensions represent the height and width in pixels, the third dimension (3) represents the channels Red, Green, and Blue, and the fourth dimension (3) represents the depthwise stacking of tiles from three different phenological stages. The convolution and linear blocks are similar to 3.43. The initial convolution block comprises a Conv3D Layer which employs a kernel size of 3, a padding value of 2, and produces an output channel of 32. This is succeeded by a MaxPool3D Layer that uses a kernel of size 2 to halve the first two dimensions of the tensor. To prevent overfitting, a dropout layer with a dropout fraction of 0.2 is inserted between the blocks. The second convolution block consists of another Conv3D layer with the same kernel size and padding with the output dimension set to 8. This is followed by a MaxPool3D layer with a kernel size of 2. The output from these two convolutional blocks is flattened into a one-dimensional vector and fed to a fully connected network. The network consists of a single hidden layer with 16 neurons followed by a single neuron in the output layer. The final dimension of the tensor at the output layer is [batch size, 1].

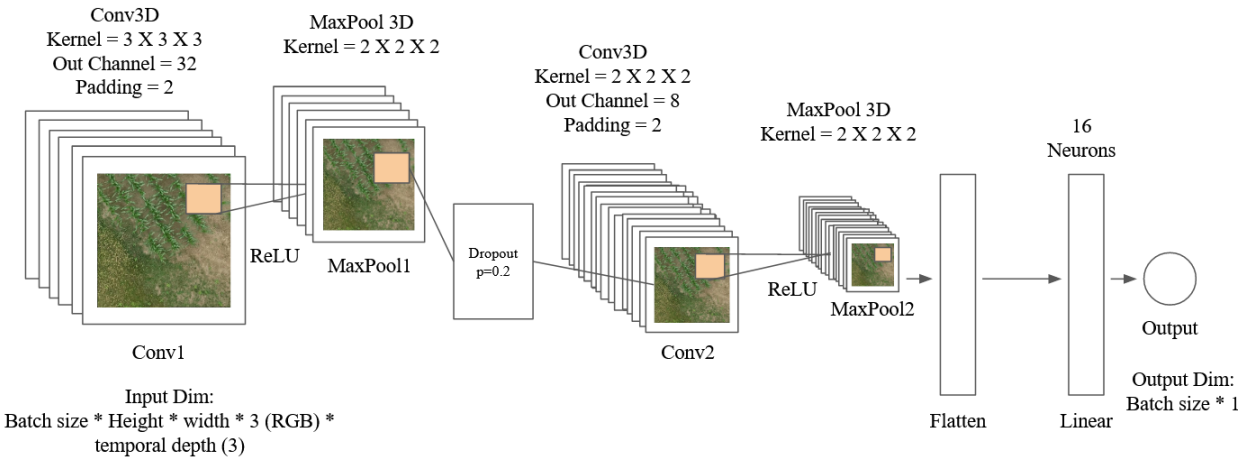

Figure 3.44: The model architecture of CNN with 3D filters where all the 3 phenological stages are added as temporal dimension consists of 2 convolution blocks, followed by fully connected layers. The input is RGB image tile of dimension [512 X 512 X 3 X 3], where the last dimension is the temporal dimension.

3.6.4 Loss function

All the Deep learning networks heavily rely on the loss functions to measure the error between the predicted and the actual value. It is crucial in determining the effectiveness of an algorithm on a use case. The loss function in this research calculates the error between the actual yield value which is measured by the harvester and the predicted yield value by the deep learning models. Some of the reasons for using a loss function from a modeling perspective are below:

1. Optimization: The loss function is used to optimize the model by adjusting the model parameters to minimize the loss. The goal of the model is it adjust parameters such that the difference between the actual yield value and predicted yield value is minimized

2. Model Selection: Different loss function behaves differently as per the model architecture and the use case. This helps in selecting the best-performing model.

3. Model Interpretation: Visualizing the predicted values shows the model sensitivity to outliers, etc. Therefore, it helps in understanding how the model has learned the data. Using such insights gives indications about model architecture complexity, choice of the loss function, and other related hyperparameters.

Pytorch a variety of loss functions for regression use cases. The following loss functions were implemented and tested as a part of this research:

3.6.4.1 Mean Squared Error Loss

It is a criterion that calculates the mean squared error 3.1 or the squared L2 norm between the input X and the target Y.

$$MSELoss = \frac{1}{N}\sum_{i=1}^{N}(y_{pred_i} - y_{true_i})^2 \tag{3.1}$$

N is the total number of samples
y_{pred_i} is the predicted value for the ith sample
y_{true_i} is the true value for the ith sample

MSE has been widely used in crop yield prediction as a loss function to optimize model performance [123] [105]. MSELoss is a convex function, which means that only a single global minimum exists. This makes it possible for optimization algorithms, such as gradient descent to converge. The global minimum is the optimal solution so having a convex function makes sure that the algorithm will converge to the same solution every time regardless of the initial parameters. MSE is also a differentiable function therefore the gradients can be calculated at any point. As a part of this research, this makes it possible to optimize the model parameter to decrease the difference between the predicted yield value and the actual and work with optimization functions like SGD and Adam. As discussed the Section 3.1.2 there are erroneous large yield values in the dataset which are due to a wide range of environmental factors such as weather. The MSE penalizes large errors more than smaller errors, therefore the trained proposed models are more sensitive to large errors.

3.6.4.2 L1 Loss

This criterion measures the Mean absolute error between each input data point and the final yield value 2.5.4.2.

$$L1Loss = \frac{1}{N}\sum_{i=1}^{n}|y_{pred_i} - y_{true_i}| \tag{3.2}$$

N is the total number of samples
y_{pred_i} is the predicted value for the ith sample
y_{true_i} is the true value for the ith sample

The main advantage of testing L1 loss is due to its less sensitivity to outliers as compared to the MSE Loss due to the reason that the absolute value treats positive and negative errors equally. Therefore, it is more tolerant of the data points which are significantly deviated from the average values. L1 loss also promotes sparsity, which makes some of the model coefficients zero. The model producing sparse solutions reduces the overall complexity. It also helps in feature selection since the weights are reduced to zero therefore, certain

features have null contributions to the model performance. Since it helps reduce the model complexity and selects only the important feature the overall computational efficiency is improved. Another main advantage of using L1 Loss is its simplicity of understanding and implementation.

Both MSE Loss and L1 Loss gave almost the same performance. However, L1 loss performed slightly better due to its less sensitivity to outliers. The outlier yield values exist due to several heterogeneous factors on the farm such as climate, irrigation, etc. Another observation was that the model training was faster in the case of L1 loss, possibly due to its promotion of sparsity in the model weights. More insights are discussed in Section 4.

3.6.5 Optimizers

Optimizers in deep learning are the algorithms that are used to adjust the model parameters based on the loss calculated at the last layer during training. The goal of adjusting weight is to minimize the error by finding the optimal configuration for the weights and the bias. *torch.optim* is the package that has the implementation of various optimization algorithms in PyTorch. The optimizers also accelerate the convergence of the model during training by adjusting the learning rate based on the gradient. This solves the problem of overshooting the global minimum when the learning rate is too large and also reduces the training time. Some of the optimizers also offer regularization to avoid overfitting the training data. A good choice of optimizer can increase the robustness of the model by minimizing the loss function in the presence of noisy or sparse data resulting in more stable model performance. The following optimizers were tested in the process of building the yield-predicting model:

3.6.5.1 Adam

Adam or Adaptive Moment Estimation [106] is a popular optimizer used in deep learning, particularly in convolutional networks. [42], [123] [80] used Adam as an optimizer for building the yield prediction model in different scenarios. It is a gradient based algorithm which is computationally efficient and works with high dimensional parameter spaces. It uses a combination of first-order (mean) and second-order (uncentered variance) moments of the gradients to adaptively adjust the learning rate for each parameter. The update rule for the parameter θ is below:

1. Calculation of the gradient:

$$g_t = \nabla_\theta L(\theta_t) \tag{3.3}$$

2. The first and second moment are estimated for the current time step t:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{3.4}$$

   where β_1 and β_2 are the decay rates for the first and second moment estimates, respectively.

3. Finally, the parameter θ is updated as below:

$$\theta_{t+1} = \theta_t - \alpha \frac{m_t^{\text{hat}}}{\sqrt{v_t^{\text{hat}}} + \epsilon} \tag{3.5}$$

where α is the learning rate, and ϵ is a small constant

One of the advantages of using Adam is it adapts the learning rate automatically by using the first and second-moment estimates of the gradients. The first-moment estimate helps the optimizer estimate the direction of the gradient, while the second-moment estimate helps the optimizer estimate the magnitude of the gradient. By using both of these estimates, the optimizer can adjust the learning rate for each parameter individually, based on the history of the gradients for that parameter. Specifically, it scales the learning rate for each parameter by the ratio of the first-moment estimate to the second-moment estimate, which helps to balance the update step based on the direction and magnitude of the gradients. This allows the optimizer to adjust the learning rate adaptively based on the gradients, rather than using a fixed learning rate for all parameters. In the experiments done as a part of this research, Adam yielded the best and most stable performance. The training time for the model was also faster as compared to the other optimizers.

3.6.5.2 SGD

SGD or Stochastic Gradient Descent [139] is another popular optimization algorithm for minimizing the cost function of the deep learning model. It is an iterative method that updates the model parameters in each iteration by computing the gradient of the loss function with respect to the parameters using a randomly selected subset of the training data, called a mini-batch. It moves the parameters in the negative gradient to minimize the loss function. Here, the learning rate determines the step size of the parameter updates, hence is an important hyperparameter. It controls the size of the update and can significantly affect the performance of the algorithm.

The Stochastic Gradient Descent (SGD) algorithm can be expressed mathematically as:

$$\theta \leftarrow \theta - \alpha \nabla_\theta J_{\mathcal{B}}(\theta) \tag{3.6}$$

where θ is the model parameters, α is the learning rate, $\nabla_\theta J_{\mathcal{B}}(\theta)$ is the gradient of the cost function with respect to the parameters computed on a randomly selected mini-batch $\mathcal{B}$, and the $\leftarrow$ symbol represents the update of the parameters.

The gradient of the cost function is computed as:

$$\nabla_\theta J_{\mathcal{B}}(\theta) = \frac{1}{m} \sum_{i \in \mathcal{B}} \nabla_\theta J_i(\theta) \tag{3.7}$$

where m is the size of the mini-batch, $J_i(\theta)$ is the cost function on the i-$^{\text{th}}$ training example, and $\sum_{i \in \mathcal{B}}$ denotes the sum over the mini-batch examples.

During training the yield prediction model, the training time for SGD was higher as compared to Adam. SGD is typically more effective for simpler models that have a smaller number of parameters. However, when there are a large number of model parameters due to the presence of spatial and temporal relationships, it did not perform well as Adam in the case of 3D Convolution based models. Therefore, SGD was not considered for final experimentation.

3.6.6 Image Augmentation

Image Augmentation is a technique to increase the size of the training dataset by creating new variations of the existing images. The goal of image augmentation is to improve the robustness and generalization of the model by exposing it to a wider range of input variations. Horizontal Flip, Shift scale rotate, RGBShift, and ChannelShuffle were the techniques used in this research to augment the input data and are further discussed in the section 4.4.4.

Albumentations [2] was the library built on top of OpenCV that was used to perform image augmentation with Pytorch. It provides a simple way to generate a wide variety of augmented images. The tensor was subjected to transformation on the device using a dictionary passed to the dataloader, which was possible due to GPU support. This approach enables the application of various augmentations in a single pipeline, allowing for the implementation of all the transformations mentioned above within a single dictionary.

3.6.7 Model Training

After defining the Model architecture 3.6.3, Loss Function 3.6.4, the Optimizer 3.6.5, and the Dataloader 3.6.2, the next step is training the model.

Following are some functions that were used during training:

1. Fit: It takes the required hyperparameters from the primary function including the number of epochs and the learning rate. It also initializes a dictionary named 'history' that keeps track of training and the validation loss for each epoch. It then iterates over the predefined number of epochs and calls the train_one_epoch and valid_one_epoch functions. These are called function returns train loss, validation loss, and the intermediate model. Subsequently, a comparison is made between the current score and the best validation scores, and the model is saved to a predefined output directory accordingly. The losses returned by the function are added to the history dictionary, which is used to monitor the training performance across epochs outside of the model.

2. train_one_epoch: This function is called for the entire data loader once for every epoch and returns the trained model, and the overall loss. It iterates over the data loader,

which generates a batch of image tiles and the corresponding yield values. Both generated images and yield values are sent to the device (GPU) for training where the model is also present. The yield predictions are made by passing these images to the model and the loss is calculated using the criterion between the predicted values and the actual values. The following three methods are essential in model learning that perform the gradient descent:

(a) loss.backward(): It is used to compute the gradients of the loss function. As discussed the loss function is used to estimate the performance of the model. The gradients of the loss function represent the direction and magnitude of the changes that are required in the model parameters to decrease the error between the predicted and the actual yield value.

(b) optimizer.step(): This method updates the parameters of the model based on the gradients calculated by the loss.backward() method. This is the step where the actual learning takes place by adjusting the model parameters so that the overall loss is reduced.

(c) optimizer.zero_grad(): This method is responsible for resetting the gradients of the optimizer to zero before the gradient of the next batch is calculated. The gradients are calculated by the loss.backward() gets accumulated for each batch, so it is crucial to reset them to zero before the computation of the next batch.

The epoch loss is then calculated by summation of losses from all the batches generated by the data loader and dividing by the dataset size as shown below. The running loss is the summation of loss per batch divided by the batch size and the dataset size is the summation of batch size which is identified from the first dimension of the data loader.

$$\begin{aligned} running\_loss \mathrel{+}= (loss.item() \cdot batch\_size) \\ dataset\_size \mathrel{+}= batch\_size \\ epoch\_loss = running\_loss / dataset\_size \end{aligned} \tag{3.8}$$

3. valid_one_epoch: It works the same as the train_one_epoch function, however, it takes model predictions of the images generated from the validation data loader, calculates and returns the final validation loss. The primary distinction here is that no gradient descent is executed, which is achieved by decorating this function using `@torch.no_grad()` which tells pytorch not to perform the gradient calculation and update parameters.

The validation and training losses are monitored per epoch to assess the model performance. The training loss after every successive epoch determines if the model is learning the training set. Therefore, the training loss should decrease subsequently. Constant training loss shows underfitting (a situation where the model is not learning) and the increased training loss shows quality issues and incorrect hyperparameters. Ideally, validation loss should also decrease in subsequent epochs which shows that the model is able to generalize well on the unseen data. Overfitting occurs if the validation loss increases over time, which can be tackled by using techniques like regularization and early stopping.

3.6.8 Model Evaluation

Model evaluation is assessing the performance of the model on a separate test dataset. In this research, the test set comprised images that were not encountered by the model during the training phase, thereby enabling the evaluation of the model's ability to generalize to unseen data. The model performance was estimated by analyzing the calculated evaluation metrics as described in Section 2.5.4. Since this is a regression problem where we had continuous yield values, we used evaluation metrics such as RMSE, MAPE, and Correlation coefficient to test model performance. The testing data loader was created by taking every 10^{th} image tile from the testing fields. This was made sure to avoid adding any overlapping tiles during testing since the consecutive tiles have some overlap. Since MAPE and RMSE were the most used evaluation metrics for previous research as described in Section 2.5.4, they were employed to evaluate the modeling errors. Another evaluation metric, Pearson's correlation coefficient which is a statistical measure that indicates the strength of the linear relationship between two variables was used. In the context of yield prediction, a higher correlation coefficient indicates a stronger linear relationship between the predicted yield values and the actual yield values. It's worth noting that the correlation coefficient may not account for more intricate, non-linear relationships therefore it was used in combination with MAPE and RMSE. The training process utilized image tiles from a single field, while the testing process utilized image tiles from the remaining fields.

The following are the advantages of creating a test set and assessing model performance in other fields:

1. Generalization of the unseen dataset: The model evaluation is done on a separate dataset from the other fields that the model has not seen during the training process. The main purpose of using a test set is to measure how well the model generalizes to new, unseen images from the other fields. During the training process, the model learns spatial and temporal relationships in the training data and adjusts its parameters accordingly to decrease the loss.

2. Correlation between fields: The test fields are not used during the training process, therefore, it shows the model generalization capability in the fields that are in the same geographic location but differ in farming techniques. The fields also differ in external factors such as moisture, topography, soil nutrient content, and Fertilizer usage. Therefore, testing on a different field shows better model performance assuming these different distinctive factors. Furthermore, a model trained on images from a specific field and then evaluated on images from different fields can provide a correlation coefficient that indicates the degree of similarity between the predicted and actual values. This coefficient is analyzed to determine whether images from one field can be used for training to achieve satisfactory results for other fields as well. Such an approach can help reduce the training complexity and computational requirements of the final yield prediction model.

3. Model Selection: Testing on different fields helps in model selection. The selection is done based on the average performance of the trained model on the data from

the unseen fields. The experimental results discussed in the Section 4 show model performance on the test set whereas the validation set was used for hyperparameter tuning.

4. Improving performance: The insights gained from evaluating the model on the test set were used to improve the model's generalization ability. For example, based on the evaluation metrics on the test set, the model was tweaked with changes such as adding more layers in the architecture, adding regularization, changing the loss function, tweaking the learning rate, etc.

3.6.9 Hyperparameter Tuning

Fine-tuning hyperparameters plays a crucial role in developing a deep-learning model for yield estimation. These parameters are predetermined by the user before the training commences and are not acquired by the model during the training phase. They are utilized to govern multiple facets of the learning process and can substantially influence the model's performance. Hyperparameter tuning is the process of selecting the optimal values for these tunable parameters to achieve the best performance on a given task such as yield prediction. This is done based on the performance of the model on the validation set. The loss observed during training and validation for each epoch can provide valuable information on which hyperparameters need to be adjusted. The following steps were followed for hyperparameter tuning:

3.6.9.1 Generation of the validation set

The cropped field tiles were divided into train and validation sets. The ratio of train and validation was kept at 80:20 respectively as shown in Fig 3.45. There were two ways of dividing the data into train and validation sets:

1. Recurring Holdout set: In this method, a fixed proportion i.e. 20% of the field tiles was used as a validation set where every 5^{th} tile was appended. Since there was an overlap between consecutive tiles, taking every 5^{th} tile will generate distinct data where consecutive tiles are not overlapped. Moreover, since every field has a variable number of tiles, taking every 5^{th} into the validation set, also keeps the proportion to 80:20. The validation indexes were found using the following Python code:

```
validation_ids = list(range(0, df_train.shape[0], 5))
```

2. Sequential Holdout set: In this method, a particular area of the field was chosen for validation. This was achieved by taking sequentially the last 20% of records. Creating the validation set using this strategy makes sure that the data that exists in the validation set is never used during training.

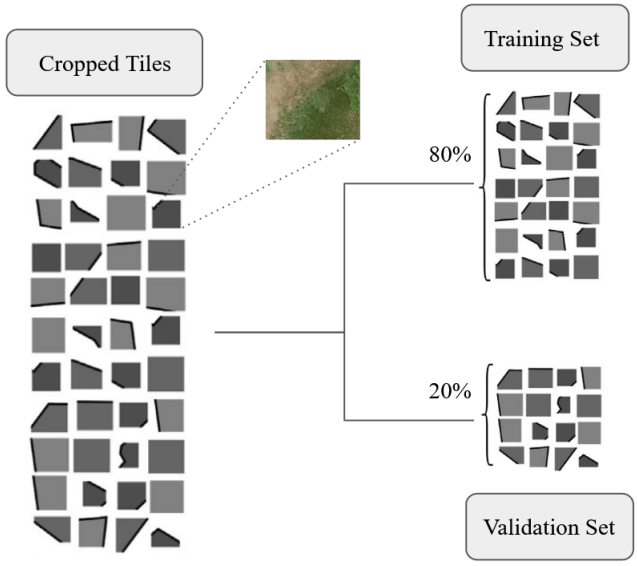

Figure 3.45: Shows the train validation split with a ratio 80:20

```
n = int(len(df_train) * 0.2)
validation_ids = df_train.index[-n:]
```

3.6.9.2 Model Hyperparameters

The following hyperparameters were tweaked to improve the performance of the model to reduce train loss and improve generalization. These hyperparameters were included in the model design, and their values were set based on prior research and then fine-tuned to suit the training data. The initial values with the range of all the hyperparameters are shown in the Table 3.4

Hyperparameter	**Intial Value**	**Range**
Batch Size	16	[16,32,64,128]
Image Size	(256,256)	[(128,128),(256,256),(512,512)]
epochs	10	[5,10,15]
Number of layers (CNN Block)	2	[1,2,3]
learning rate	0.1	[0.1,0.01,0.001]
Stride	1	[1,2,3]
Kernel Size	3	[2,3]
Dropout	0	[0.1,0.2,0.3,0.4]
Number of Filters	32	16,32,64

Table 3.4: The table shows hyperparameter initial value with the tuning range

1. Learning rate: The learning rate is a hyperparameter that determines the step size at each iteration while moving towards the minimum of the loss function. The learning rate determines the magnitude of the update made to the weights of the network during each iteration of the training process. A high learning rate can result in the weights changing too quickly, leading to instability in the training process, while a low learning rate can result in the weights changing too slowly, leading to slow convergence or getting stuck in a local minimum. The learning rate was set to 0.01 initially. When the learning rate is too high in a CNN training process, the model may experience difficulty converging or even oscillating between different solutions. On the other hand, if the learning rate is too low, the training process may become slow to converge or even get stuck in a local minimum, leading to suboptimal results. Given a loss function L and a set of weights W in the network, the learning rate η determines the amount by which the weights are updated during each iteration of the optimization process, according to the following equation:

$$W' = W - \eta \frac{\partial L}{\partial W} \tag{3.9}$$

where W' represents the updated weights, η represents the learning rate, L represents the loss function, and $\frac{\partial L}{\partial W}$ represents the gradient of the loss function with respect to the weights W.

2. Batch Size: It refers to the number of image tiles that are processed and propagated through the network in a single forward/backward pass. During the training, the dataset is divided into batches of size n, and the weights of the network are updated after processing each batch. The dataloader in Pytorch provides a convenient way to iterate over the dataset and load data in batches. During the 3.6.7 train_one_epoch and valid_one_epoch methods, the iteration of the training and validation loader in each pass generates one batch of training tiles with respective yield values. Larger batch sizes lead to faster training, as the model can process more samples in parallel. However, larger batch sizes require more memory, and the learning was less stable due to the noise introduced by the mini-batch gradient estimation. On the other hand, smaller batch sizes resulted in slower training times, but it allowed for more frequent weight updates and improved the convergence of the model. Smaller batch sizes regularize the model and prevent overfitting. The initial batch size was set to 16 and was increased subsequently. For the number of batches B required to cover the entire dataset:

$$B = \lfloor N/m \rfloor$$

where N is the total number of samples in the dataset, and m is the batch size.

3. Epochs: The number of epochs is a hyperparameter that determines the number of times the entire training dataset is processed during the training process. In each epoch, the train_one_epoch and valid_one_epoch methods are called for the entire

training and validation set respectively. Therefore, one epoch consists of one forward pass and one backward pass of all training tiles through the network. Too few epochs lead to underfitting, and too many epochs lead to overfitting, as the model learns the training data too much. The optimal number of epochs is optimized by monitoring the training and validation loss graph during training. A common technique of early stopping was used to stop the training process when the validation loss increased for certain epochs. The number of epochs were set to 10 initially and was adjusted based on performance on the validation set in conjunction with other hyperparameters such as learning rate and batch size.

4. Number of layers: The number of layers represents different layers such as Conv3D (convolutional), MaxPool, Dense, etc. layers that are responsible for extracting the spatial and temporal relationships from the input images. Increasing the number of layers allows the model to learn more complex representations of the input tiles, but can also increase the risk of overfitting if the model becomes too complex. The number of layers also affects the computational resources required to train the model as adding layers adds more tunable parameters that require more computational resources such as memory and processing power, which can impact the training time and scalability of the model.

5. Dropout rate: Dropout is a regularization technique used to prevent overfitting in deep-learning CNNs. Dropout works by randomly dropping out a fraction of the nodes in the network during training, which helps to prevent the network from relying too heavily on any one input feature or set of features. The p-value passed in the dropout layer determines the fraction of nodes that are randomly dropped out during training. Setting the dropout rate too low results in overfitting, while setting it too high results in underfitting. The initial value was set to 0.1 and then increased in conjunction with other hyperparameters.

6. Stride: It determines the number of pixels the filter moves in each step over the image to extract features during the convolution operation. Initially, the stride was set to 1, which means that the filter moves one pixel at a time. Then it was increased further, a larger stride reduced the spatial resolution of the output feature map, while a smaller stride preserves more information. Decreasing the stride size to 0 reduced the feature map dimension to 0, which caused an issue while training. Therefore, strides were tested with values 2 and 3. A small stride is typically used when the input data is small or when the spatial relationships between pixels are important, while a large stride is typically used when the input data is large or when spatial resolution is not as critical. Therefore, the model was tested with large stride values. The dimension of the output feature map after passing through the convolution layer is given below:

$$W_{out} = \left\lfloor \frac{W_{in} - F + 2P}{S} + 1 \right\rfloor$$

$$H_{out} = \left\lfloor \frac{H_{in} - F + 2P}{S} + 1 \right\rfloor$$

W_{in} and H_{in} are the width and height of the input image, respectively. F is the size of the filter or kernel, P is the padding, and S is the stride. The resulting values of W_{out} and H_{out} are the width and height of the output feature map, respectively.

7. Number of Filters: It determines the depth of the output feature map. The filter itself is a 3D tensor in case of 2D convolution, with dimension [filter_height, filter_width, input_depth]. During the forward pass, the filter is convolved with the input images (which have the same depth as the filter) to produce a 2D activation map, therefore in the case of RGB tiles, the input_depth of the filter is 3. Increasing the number of filters allows the model to learn more complex spatial and temporal relationships in the input image tiles, and also increases the computational cost and overfitting issues. The initial number of filters was set to 3 for all the convolutional layers (Conv2D and Conv3D) and then tweaked in conjunction with other hyperparameters.

8. Kernel Size: It is the size of the filter/kernel used during the convolutional operation. The size is typically a square matrix. It determines the receptive field of the convolutional layer, which is the area where the input image that each neuron in the layer is connected to. A larger kernel will have a more receptive area and allows it to capture more complex features that are spread over a larger area of the input image. On the other hand, smaller filters are used when the features are simpler and local. The initial size was set to 3, where more local and simpler features were extracted from the tiles, and increased subsequently to extract more complex features.

Chapter 4

Experimental Results

Overall, we aimed to achieve the following objectives in this research:

1. Develop a deep learning model that can accurately predict corn yield using UAV data captured during different crop growth phases fused with the ground yield measurements captured at the time of harvesting.

2. Assessing the generalization performance of the model by predicting the yield of other fields (that were not a part of the training phase) and examining their interrelationships to determine the level of similarity among them.

3. Comparing the performance of the Spatial and Spatial-Temporal models and demystifying if adding the temporal dimension improves the prediction performance.

4. Finding the phenological stage that exhibits the strongest correlation with the final yield and gives the best prediction performance when the model is trained only using image tiles from that particular day.

5. Comparing the performance of two Spatial-Temporal models (a) 3D CNN (2) that inputs paired images from two different phenological stages (b) 3D CNN (3) that has input images from all three growth stages stacked together depthwise.

6. Revealing if increasing the training data using techniques like image augmentation improves the prediction performance.

7. Finding the best-performing image resolution, from a set of three and checking if the notion of high resolution always performs better is valid for this use case.

8. Evaluating the model's performance using two well-known regression loss functions, namely (a) L1 Loss and (b) MSE Loss.

9. Testing Regularization with a set of techniques that add additional constraints to the model's parameters during training, improves performance on the validation and test set.

10. Performing hyperparameter tuning to further enhance the prediction results.

To achieve the above objectives, the orthomosaics that were processed by IndroRobotics by combining drone images taken during several flights were used as training/testing data from three fields (Field 1, 2, and 3) and three different phenological stages (Early, Mid, and Late). The yield measurements captured by Trimble in the form of a shapefile during harvesting served as the ground truth data. The images and yield values were combined together using data fusion where they share the same spatial resolution. Several deep convolutional neural network architectures were trained and evaluated on metrics including RMSE, MAPE, and Correlation coefficient. In this section, the results of the experiments are presented to address the above objectives and overcome the existing challenges in yield prediction. The results of these experiments demonstrate the potential of RGB imagery in developing a prediction model with high accuracy. Finally, the results of the experiments are analyzed to determine how these findings can help farmers reduce costs, make timely agricultural decisions to improve agricultural practices, and increase crop yield.

4.1 Training and Testing Data

The training data for the model included the RGB drone orthomosaics that were cropped into tiles of size 9 X 9 m as described in the Section 3.1.3 and fused with yield values as shown in Fig. 3.39 based on the geospatial coordinates. The training of the model was done on image tiles from a single field and the testing was done on the image tiles from other fields. The purpose of this evaluation was to assess the performance of the model on fields with varying farming techniques and geographical locations that are unseen to the model during the training phase. Initially, the yield distribution was analyzed, as the training performance of the model highly depends on the data distribution as shown in Fig 4.1. If the data is biased and skewed, the model may not learn the underlying patterns and relationships. The Field 1 data was a normal distribution with one mean, however, Field 2 and Field 3 distribution is a bimodal distribution with two peaks centered around two different means. The Training data was further divided into Training and Validation sets in the ratio of 80:20 respectively, based on two techniques (a) Recurring Holdout and (b) Sequential Holdout, as described in Section 3.6.9.1.

There were two major image post-processing techniques employed in the dataloader. (a) Resizing the image: to a consistent size ensures that all the images in the dataset have the same tensor size. It also reduces overfitting as smaller images contain fewer details and are more likely to capture yield density features instead of capturing specific details of the crops (b) Normalizing the image pixels: The pixel values for RGB image tiles can range from 0 to 255, where 0 represents black and 255 represents white. Dividing the pixel values by 255 scales the final values between 0 and 1, which makes it easier for the model to learn and have stabilized training. It also solves problems such as vanishing gradients, where gradients become very small during training and the model stops learning.

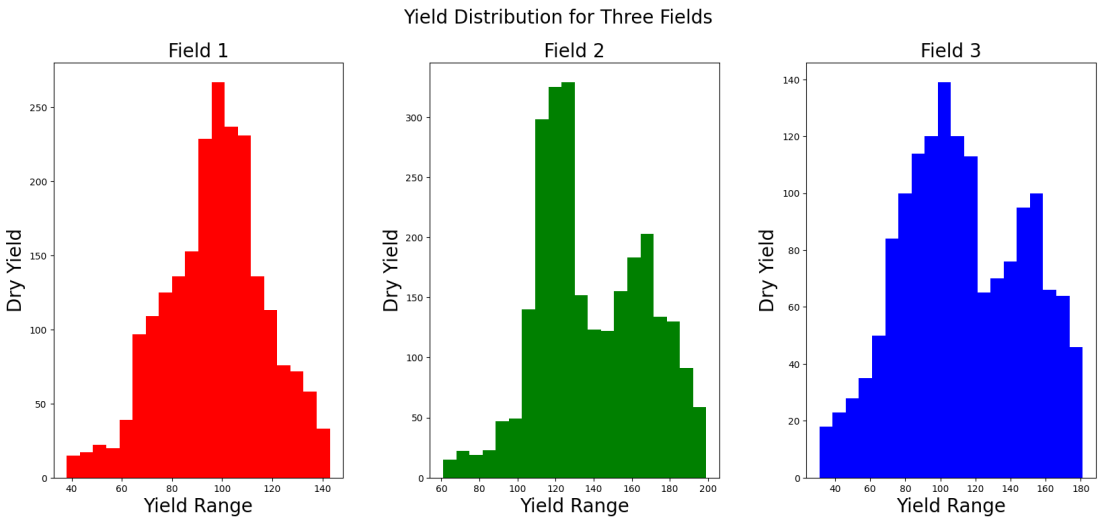

Figure 4.1: Comparison of Dry Yield Distribution for three demo Fields: Field 1 (in red), Field 2 (in green), and Field 3 (in blue). The histograms show the frequency distribution of processed corn yield measurements in each field, with yield range on the x-axis and the number of observations on the y-axis

4.2 Evaluation Metrics

The yield prediction is a regression problem, where the model predicts continuous values. The purpose of the evaluation metric is to measure the accuracy of the model in predicting yield. By evaluating the performance on (a) Test set: the model's generalization capability on unseen images from different fields (b) Train set: the model's capability to learn patterns in the train set (c) Valid set: the model's generalization performance on unseen data from the same field is analyzed. There are three main evaluation metrics that were used to find model performance described in the Section 2.5.4.

1. Mean Squared Error (RMSE): It measures the average root squared difference between the predicted and actual yield measurements. A lower RMSE value indicates that the predicted value is close to the actual value. It captures the magnitude of errors illustrating the greater impact of large errors on the overall RMSE value.

2. Mean Absolute Percentage Error (MAPE): It measures the average percentage difference between the predicted and actual yield measurements. It captures the relative error and is advantageous in this research due to the fact that the yield values have a wide range and vary significantly in different regions of the field. The MAPE is a scale-independent metric that allows for the comparison of model accuracy between different fields or regions. This means that the performance of a model trained on one set of yield values can be evaluated and compared to a model trained on a different set of yield values without being influenced by differences in the scales of the values.

3. Correlation coefficient (CORR): It shows the strength of the linear relationship between the actual and the predicted value. If the value is close to +1 then the model is

able to capture the underlying relationship. The drawback is that a high correlation does not guarantee a good and accurate model. To account for this we have used it in conjunction with other metrics.

By combining all of the above evaluation metrics, different aspects of model performance are evaluated. A low RMSE shows that the model is making predictions close to the actual values on average. Having a low MAPE shows that the model predictions are accurate regardless of the scale. Finally, a high correlation coefficient illustrates the linear strength of the relationship between true and predicted yield. Therefore, a combination of all of these metrics gave a complete picture of the model's strengths and weaknesses.

4.3 Experimental setup

The development of the yield prediction model was done on the PyTorch Framework. The model architecture consists of several CNN Blocks that have layers including Convolution, MaxPool, ReLU, and Dropout. This was followed by fully connected layers and 1 neuron in the output layer where the loss was calculated. All the code was written in Python programming language and Pytorch [18] framework was used. Input Orthomosaics from three fields (1,2,3) belonging to three different phenological stages were of size 12.13 GB. The orthomosaics were cropped locally using QGIS and were uploaded to Google Drive. Google Colab Notebook was then used to further convert these cropped slices to image tiles of size 9 X 9 m and fused with the yield data.

4.3.1 Computational Infrastructure

There are three versions of computational devices used:

1. Local Computation: Asus Zenbook with Intel i5 8^{th} generation processor with 8GB RAM and 512 GB SSD was used as the local computation device. The 8 GB RAM is of type LPDDR3, and the memory speed was 3.4 GHz with Windows 10 Operating System. Following are the tasks that were done on this computational infrastructure:

   (a) Visualizing the Orthomosaic: This was done using QGIS Windows software. The objective of doing a visual analysis was to make sure that the Orthomosaic is free from any drone-related error described in Section 3.1.4. For instance, as shown in Fig. 4.2, it was discovered that Field 3 had an unfocused image from the July 18^{th} flight. Moreover, visual inspection helped in determining the appropriate data processing steps that included image resizing, cropping the images into slices, and normalization. The cropping coordinates for different slices were determined using QGIS.

   (b) Cropping into slices: The bigger orthomosaics were cropped into smaller slices described in the Section 3.3.2 was also done on local computation using the QGIS command line tool called OSGeo4W Shell.

Figure 4.2: Visual inspection helped in determining that the orthomosaic of Field 3 captured during the July 18th flight was unfocused.

(c) Storage: All the raw orthomosaics and the raw yield data were temporarily stored in this device.

(d) Interpolation: As described in Section 3.2.10, after preprocessing, the raw yield data was interpolated using the Vesper 1.6 software. This software was locally installed on this device and the processed yield data was given as an input. The final interpolated data from each field was also temporarily stored on the device.

2. Google Drive Storage: The main advantage of using Google Drive as storage is that it can be mounted as a normal file system to Colab notebooks where all the computation takes place. An additional storage of 200 GB was added to the drive to store all the corresponding data. The drive stored the cropped slices, tiles, colab notebooks, trained models, processed geopandas dataframe, and the raw shapefiles from the fields.

3. Google Colaboratory: It is a cloud-based service provided by Google that allows to write and run Python code in Jupyter notebooks. One of the major reasons for using Colab is its ease of use and Google Drive can be mounted to any notebook that behaves like a normal file system. It offers three types of hardware resources (a) CPU (Central processing units) for lighter tasks such as data processing and visualization (b) GPUs (Graphic Processing Units) for heavier tasks that involve heavy computations like training deep learning models, and (c) TPUs (Tensor Processing Units) that are specialized in performing matrix operations and are used to perform larger scale deep learning tasks such as image processing, speech recognition, etc. Another reason for using Colab is web availability and no requirements/configuring of additional packages or software. The compute resources offered by Colab can be grouped into two categories:

Listing 4.1: The below two code lines written in python was used to mount google drive to colab notebook where it acts like a normal file system with high IOPS

```
from google.colab import drive
drive.mount('/content/drive')
```

(a) Basic Compute: The free version of Google Colab provides access to CPUs, GPUs, and TPUs up to a specific quota and has a time limit of 12 hours. One of the GPUs available is the Nvidia Tesla V100-SXM2-16GB, which has a memory capacity of 16GB. The compute is one-demand, so it does not offer all the computing every time. Post-processing the yield data involved running several data cleaning techniques discussed in Section 3.2, visualizing the output for interpolation, creating tiles, and fusing with yield measurements all of these were performed on Basic compute. The main reason was that these operations were not computationally heavy.

(b) Premium Compute: Colab Pro is a paid service that offers compute units in bundles, such as 100 or 500 units. It provides NVIDIA A100-SXM4-40GB GPU that has a memory size of 40 GB and is 60-70% faster than V100 which is supported in the free version. The CPU has a considerable RAM capacity, capable of providing up to 80 GB of memory, whereas the GPU provides a maximum memory size of 40 GB. The resource allocation is however dynamic based on the demand. We trained and tested three different CNN models on this resource tier. The first model was a 2D CNN that aimed to capture spatial relationships. The second model was a 3D CNN that stacked image tiles from two different phenological stages and the third model was also a 3D CNN that stacked images from Early, Mid, and Later growing seasons, both can capture the spatial and temporal relationships. To finalize the models, we conducted hyperparameter tuning and experimented with various techniques, such as image augmentation, different optimizers, regularization, and loss functions, all of it was done on the Premium Tier.

4.4 Numerical Results

4.4.1 Phenological Stage Analysis

The analysis of the phenological stage was to track and analyze the growth stages of the crop. Experiments were performed by different model architectures described in the Section 3.6.3 to (a) Determine the phenological stage that provides the most informative data to the model and (b) Assess the model's performance by comparing its results when given a single image versus when presented with two images representing distinct growth stages, merged together. The three phenological stages (Early, Mid, and Later) and date of acquisition are shown in Table 4.1. The reasons for considering these dates were (a) free from any erroneous drone-related measurement as described in Section 3.1.4 for all the demo fields (b) dates were in the middle of the respective growth season (c) provided the best performances individually when trained using 2D CNNs described in Section 3.6.3.1.

Phenological Stage	Date
Early	June 20, 2021
Mid	July 18, 2021
Later	August 29, 2021

Table 4.1: The table shows the phenological stages and the respective dates when the images were taken using the drone

4.4.1.1 Independent Phenological Stage Analysis

The objective of these experiments was to consider each phenological stage independently. The aim was to determine the phenological stage that is best correlated with the final yield measurements. Since the temporal dimension is missing, 2D CNN described in section 3.6.3.1 was utilized to extract the spatial features from the image pixels. A batch of image tiles from a single phenological stage was passed as training data to the model. The model extracts the salient image features using the convolutional layers and passes them to fully connected layers. The outcomes of these experiments from the Comparison Bar Graph shown in the Fig 4.3 are listed below:

1. The Mid stage exhibited the lowest MAPE value at 24.57, while the Early stage had a slightly higher MAPE of 26.1, and the Later stage had the highest MAPE of 30.2.
2. Similarly, the Mid stage demonstrated the lowest RMSE value at 31.79, whereas the Early and Later stages had higher RMSE values centered around 35.
3. The CORR was highest during the Early and Mid stages, with the Early stage having a slightly higher value at 0.52.
4. The Later phenological stage had the highest MAPE and RMSE values and the lowest Correlation suggesting that the model performance was the worst when trained on image tiles using that phase.

The graph suggests that the performance of the spatial models varied depending on the phenological stage, indicating the importance of considering the growth stage when predicting yields using these models. The Early and the Mid stage were seen to be more correlated with the final yield suggesting that these images provide more spatial information to the model and has the potential to get good prediction performance.

4.4.1.2 Paired Phenological Stages Analysis

The aim of these experiments was to combine 2 phenological stages, with the second stage serving as a temporal dimension, and assess whether incorporating information about the growth stage enhances performance. 3D CNN described in Section 3.6.3.2 was used, where the input was a 4D tensor. The height and width represent the spatial dimensions and the phenological stage acts the a temporal dimension. The 3D kernels traverse the first

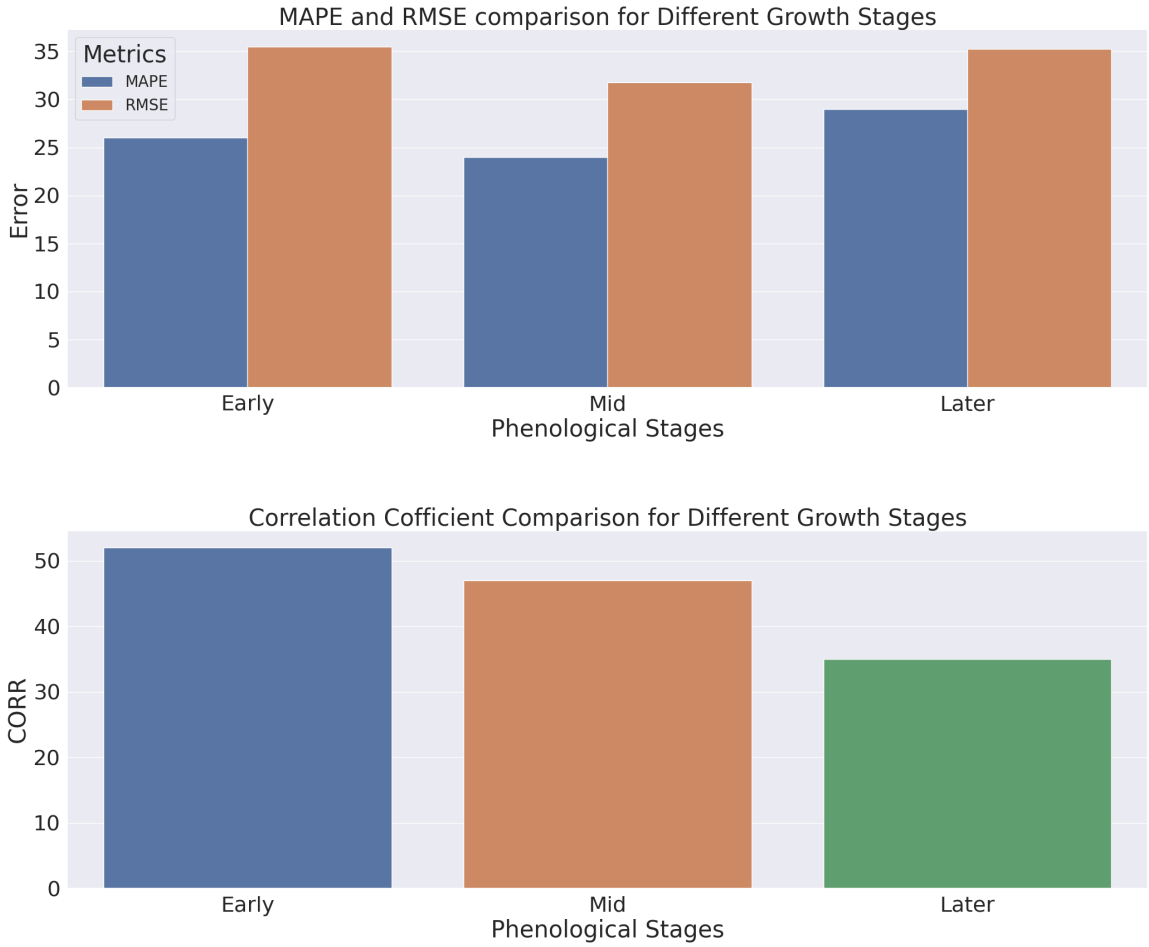

Figure 4.3: Performance of Spatial models on different independent growth stages. The Correlation Coefficient was scaled by a factor of 100 for comparison

two dimensions to capture spatial information from the image pixels, while also traversing through the depth dimension to observe the changes in these pixels over time. The following are the insights captured from the performance graph shown in Fig 4.4:

1. MAPE and RMSE values increased from Early-Mid to Mid-Later and then to Early-Later, indicating that the model's performance worsens as the crop stage growth progresses.

2. The Early-Mid phenological stage has the lowest errors for both metrics (RMSE and MAPE) while maintaining the highest correlation. The MAPE for this stage was 18.10 and the RMSE was 22.26. For the Mid-Later stage, the MAPE increases to 21.20, and the RMSE increases to 24.76. Finally, for the Early-Later stage, the MAPE decreases to 19.21, but the RMSE increases further to 26.31.

3. The CORR values ranged between 0.28-0.42, with the Early-Mid pair having the highest value at 0.42, indicating a stronger linear relationship between the predicted and actual values for this pair.

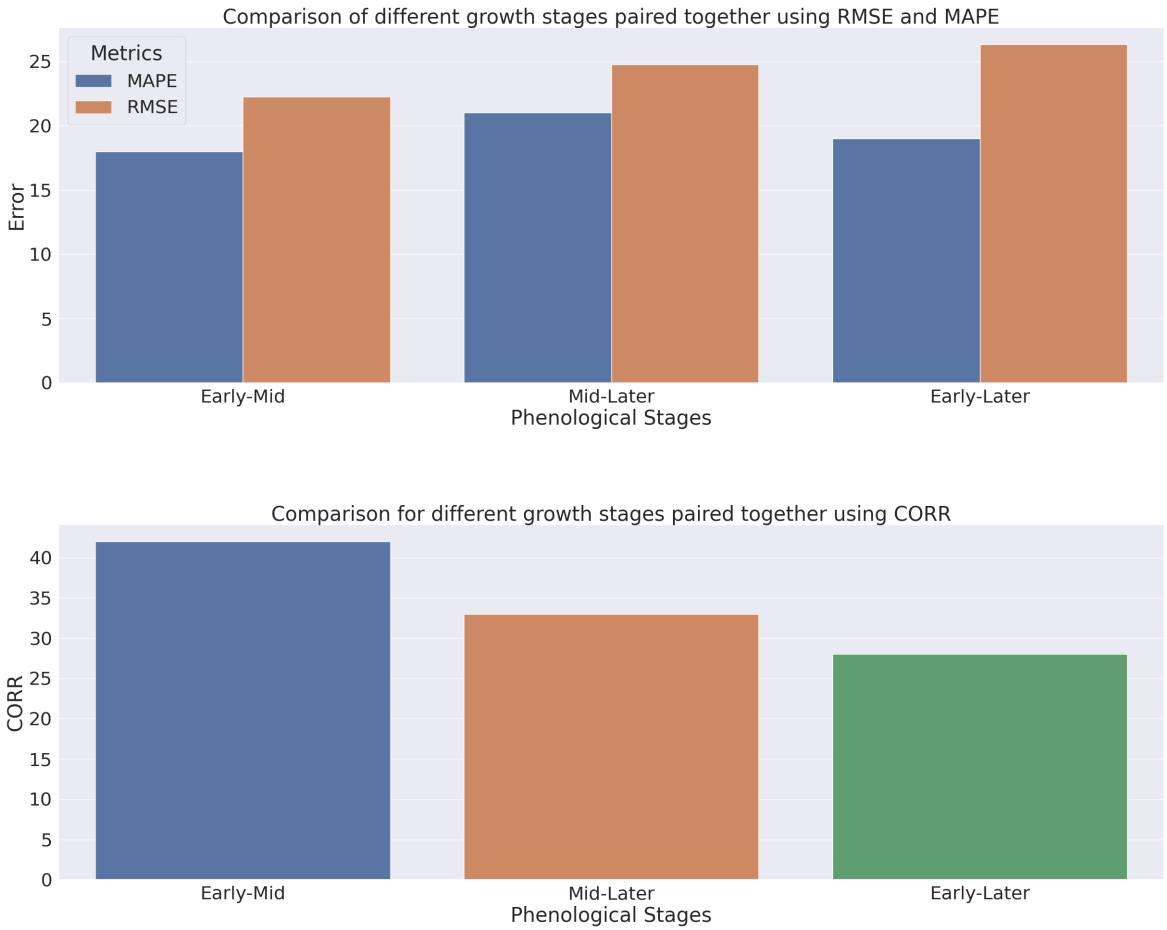

Figure 4.4: Performance of Spatial-temporal model on paired growth stages. The Correlation Coefficient was scaled by a factor of 100 for comparison

4. Overall, the results suggest that the Early-Mid pair has the lowest errors in terms of MAPE and RMSE, and the highest CORR, making it the best pair for yield prediction.

The differences in the evaluation metrics across the three paired periods are due to a variety of factors, such as different farming techniques, soil moisture content, changes in weather patterns, variations in crop growth, or shifts in environmental conditions. The performance graph shows that the Early-Mid pair performed best and combining either of them with the Later stage degrades the performance. Though the plot illustrates that there is minor variation in errors between different pairs of phenological stages these results are better than the independent analysis done in Section 4.4.1.1. This shows that the Early and Mid growing pair proved to be more correlated to the final yield and adding the temporal dimension improved the model's performance.

4.4.2 Spatial Model vs Spatial-Temporal Model Comparison

The Spatial model uses 2D CNN described in Section 3.6.3.1 which captures the spatial variability of the crop yield across different fields. The model uses independent tiles from various fields and analyzes the crop characteristics such as crop growth, color, etc. to predict yield. This model assumes that the factors affecting yield vary only in space and not in time.

Spatial-temporal models, on the other hand, incorporate both spatial and temporal information to predict crop yield. These models take into account the dynamic changes in soil, climate, and other factors over time. The model stacks field tiles from different phenological stages and understands how growth changes over time to identify long-term trends in crop production. Two proposed Spatial-Temporal models, 3D CNN with two paired phenological stages 3.6.3.2 and 3D CNN with all three phenological stages stacked together 3.6.3.3 were used.

Through the comparison of Spatial models and Spatial-Temporal models, it can be determined whether incorporating the fourth dimension to capture changes in crop growth over time has an impact on the model's performance in predicting yield. 2D CNN model has fewer model parameters and requires less computational power, whereas spatial-temporal models are much more complex. In addition, the comparison can aid in selecting a model based on computational requirements and deployment complexity.

Following are the performance comparison insights derived from Fig. 4.5 between the Spatial model and Spatial-Temporal models. Here, the 3D CNN(2) represents the 3D CNN model where 2 phenological stages were grouped together and 3D CNN(3) represents where all three phenological stages were combined.

1. The MAPE values range from approximately 18 to 32, with the lowest MAPE value being 18.07 for the 3D CNN(2) and the highest MAPE value being 32.38 for one of the 3D CNN models.

2. The RMSE values for each model range from approximately 21 to 31, with the lowest RMSE value being 20.98 for one of the 3D CNN(3) models and the highest RMSE value being 30.49 for one of the 2D CNN models.

3. The 2D CNN models generally have higher RMSE values than the 3D CNN models, indicating that the 3D CNN models can more effectively capture the variability in the yield measurements.

4. The fact that the 3D CNN(3) models slightly perform better than the 3D CNN(2) models in terms of RMSE indicates that increasing the number of growth stages helps the model to improve the prediction performance of the values on the tails of the distribution i.e region of highest and lowest yields.

5. On the other hand, MAPE for the 3D CNN(3) is overall higher than that of the 2D CNN and 3D CNN(2) models. This indicates that the 3D CNN models have higher prediction errors on average compared to other models. Two possible reasons are

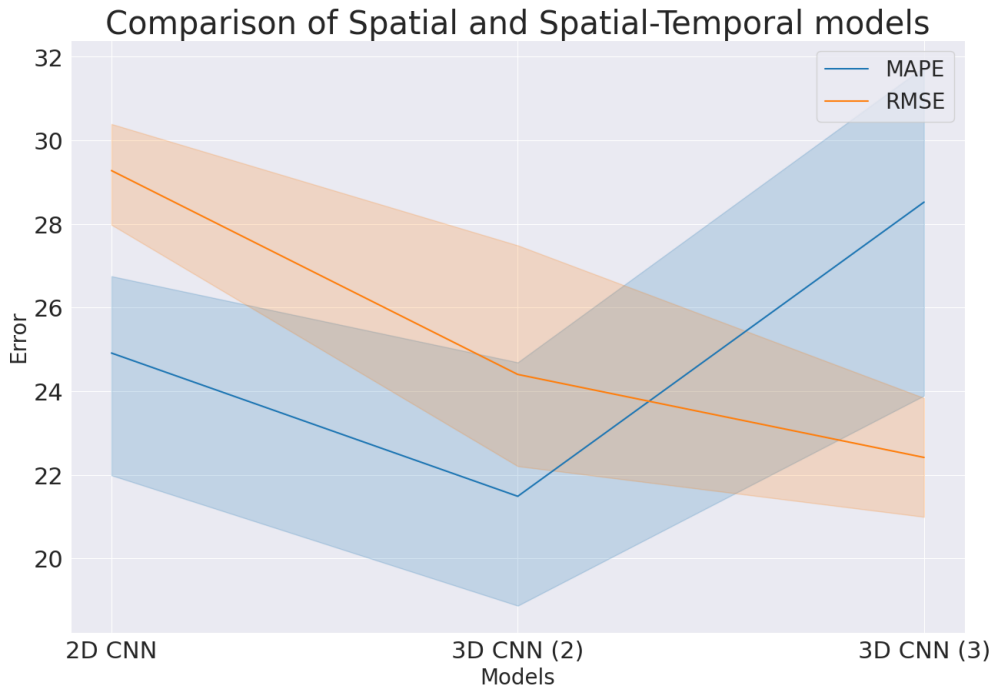

Figure 4.5: The graph shows the MAPE and RMSE comparison for Spatial and Spatial-Temporal models used in yield prediction. The x-axis shows the different models, while the y-axis shows the error values. Here, the 3D CNN(2) represents the 3D CNN model where 2 phenological stages were grouped together and 3D CNN(3) represents where all three phenological stages were combined together.

(a) adding all three growth stages may introduce more noise and variability in the data and (b) overfitting due to increased model parameters. Therefore, both the 3D CNN models were hypertuned and a series of regularization techniques were added in further experiments.

Upon analysis, it was identified that the overall performance of the Spatial-temporal models is better than Spatial models according to the evaluation metrics. Adding the temporal dimension reduced the RMSE by about 16% and MAPE by about 18%. The case where the MAPE is low for the 2D CNN model and has a high RMSE indicates that 2D CNN models tend to have lower percentage errors, but they are more likely to produce large outliers in the predictions. This behavior can be explained by the fact that RMSE places more emphasis on large errors. The 3D CNN models on the other hand have fewer large errors that contribute to a lower RMSE therefore they are able to better capture the variability in yield. A low RMSE is a more accepted case because the model is able to capture the area of high yield and low yield which are cases that lie on the tails of the distribution. Overall, adding the Phenological information as the temporal dimension improved the model performance, and the best performance was seen for the 3D CNN model where two phenological stages were coupled.

4.4.3 High and Low Spatial Resolution Imagery analysis

The UAV imagery shows insights into the health and growth of the crops. The resolution of the imagery determines the level of detail that can be observed, which in turn affects the accuracy and precision of the prediction model. High spatial resolution imagery can provide more detailed information about individual plants, while low spatial resolution imagery can provide a broader picture of the entire field. The low, medium, and high resolutions were categorized as shown in the Table 4.2. Image resolution comparison is important in yield prediction models because it helps determine the optimal image resolution for accurately predicting yield. While it may seem intuitive that higher-resolution images would always lead to better performance, this is not always the case. In some situations, higher-resolution images may contain more noise and irrelevant information that can decrease the accuracy of the yield prediction model. By comparing the performance of the yield prediction model using images of different resolutions, it is possible to determine the optimal image resolution that leads to the best performance. This information can then be used to guide decisions about the type and quality of imagery to collect in future yield prediction efforts, potentially reducing data collection costs while maintaining, or even improving, the accuracy of the model. During these experiments, three image resolutions were tested as shown in Table 4.2.

Category	**Resolution in pixels**
Low Resolution	128 X 128
Medium Resolution	256 X 256
High Resolution	512 X 512

Table 4.2: Different categories of image resolution based on pixel dimensions

The spatial model performance is shown in Fig 4.6, whereas the spatial-temporal model performance is shown in Fig. 4.7. The following insights were captured from these two graphs:

1. The MAPE values for the Spatial model ranged from 29.22 for low-resolution images to 22.32 for high-resolution images.

2. Comparatively, the MAPE values for the Spatial-Temporal models range from 20.11 for low-resolution images to 18.21 for medium-resolution images and then increased to 29.32 for high-resolution images.

3. This concludes that the best MAPE was seen for high resolution in the case of spatial models and medium resolution in the case of spatial-temporal models.

4. The RMSE values for the spatial model range from 31.91 for low-resolution images to 26.38 for high-resolution images. However, the RMSE values for the spatial-temporal model range from 22.77 for low-resolution images to 21.13 for medium-resolution images, and then increase to 26.39 for high-resolution images.

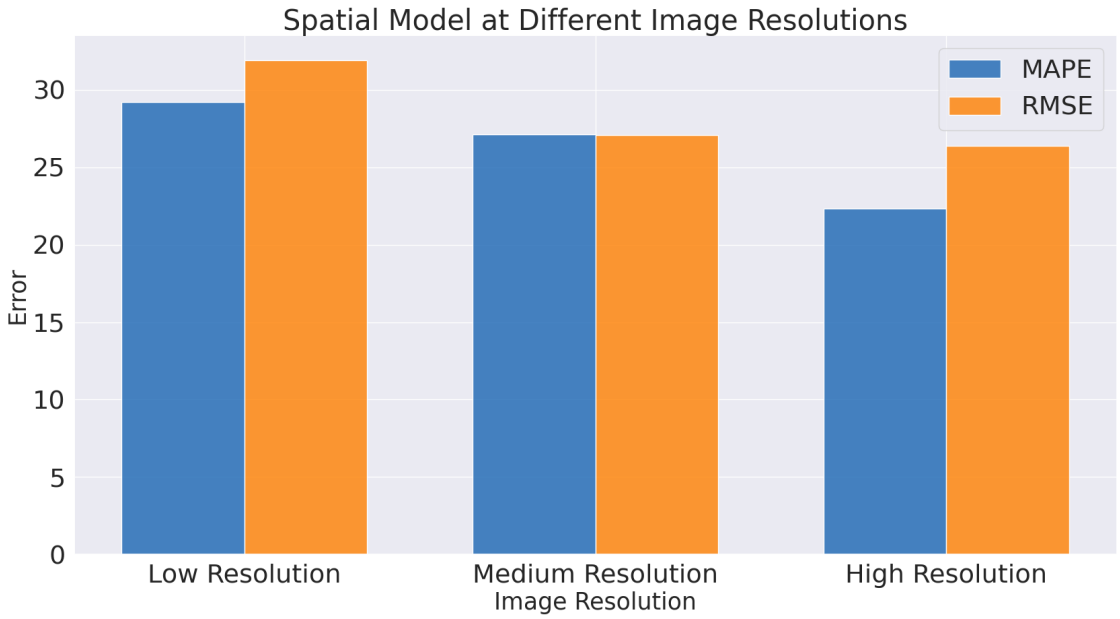

Figure 4.6: Bar chart showing the Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE) for Spatial models at different image resolutions.

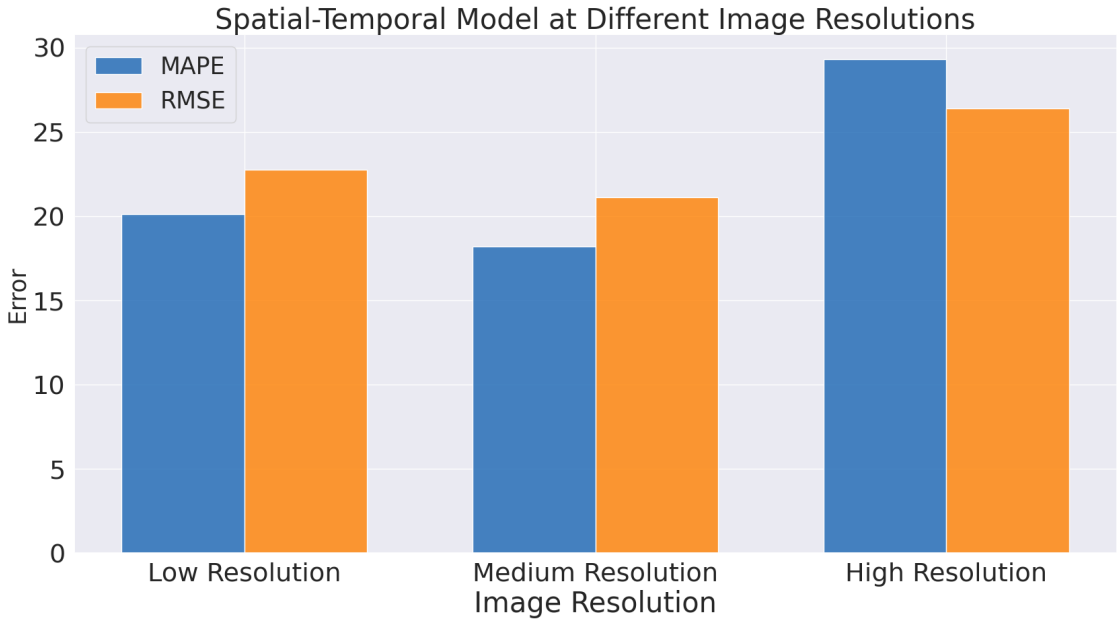

Figure 4.7: Bar chart showing the Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE) for Spatial-Temporal models at different image resolutions.

5. As the image resolution increases from low to medium, the performance of both the spatial and spatial-temporal models improves. This is indicated by the decreasing values of MAPE and RMSE as the resolution increases.

6. The spatial model exhibits the best MAPE performance with High-Resolution imagery, as it might require more spatial details to make predictions closer to the actual values. On the other hand, the Spatial-Temporal model performed better with medium-resolution imagery, as it incorporates temporal data, and adds more spatial information, or higher resolution leading to a degradation in performance due to noise.

Overall, Medium showed the best performance in the case of Spatial-Temporal models, and Medium and High showed comparative performance in the case of the Spatial model. The MAPE values show some variability with no clear trend across the different resolutions. However, the Spatial-Temporal model appears to be a better choice for predicting data at different resolutions, as it consistently achieved lower RMSE values compared to the Spatial model. In addition, the RMSE values show a steady decrease from Low to Medium resolution for both models, indicating that the models perform better with medium-resolution data. Therefore, the Medium resolution would be the most optimized choice considering both types of models. Moreover, Medium-resolution images were half the dimension of high-resolution images therefore, the images captured by the UAV could also be reduced to half the resolution for this particular use case of yield prediction, however, it might have effects on other use cases.

4.4.4 Augmentation

The augmentation technique is generally used in computer vision tasks to augment the training data in order to improve the model's generalization performance. The augmenting of the training data increases the diversity and exposes the model to different variations of input data to increase the robustness. Below augmentation techniques were used to expose the model to different lighting conditions, camera angles, and image color variations. All the transformations were part of the "BuildDataset" class which was used to create the data loaders.

1. Horizontal Flip: Flipping horizontally includes, the left side of the image becoming the right side and vice versa. This provides a model with different orientations of the field tile. The probability value was set to 0.5, meaning that the images have a 0.5 probability of getting this transformation.

2. ShiftScaleRotate: This technique is a combination of three transformations including shift, scaling, and rotation for generating new samples. The shift range value was set to 0.0625 which specifies the maximum amount of horizontal and vertical shift. The Scale limit was set to 0.05 which specifies the maximum amount of scaling factor. Lastly, the rotation limit value was set to 10 degrees which control the maximum rotation angle. The overall probability of applying the ShiftScaleRotate transformation was also set to 0.5 to add randomness.

3. RGBShift: It is a technique of shifting the pixel values of an image by the random amount in red, blue, and green channels separately. The new image is slightly different from the original one in terms of colors.

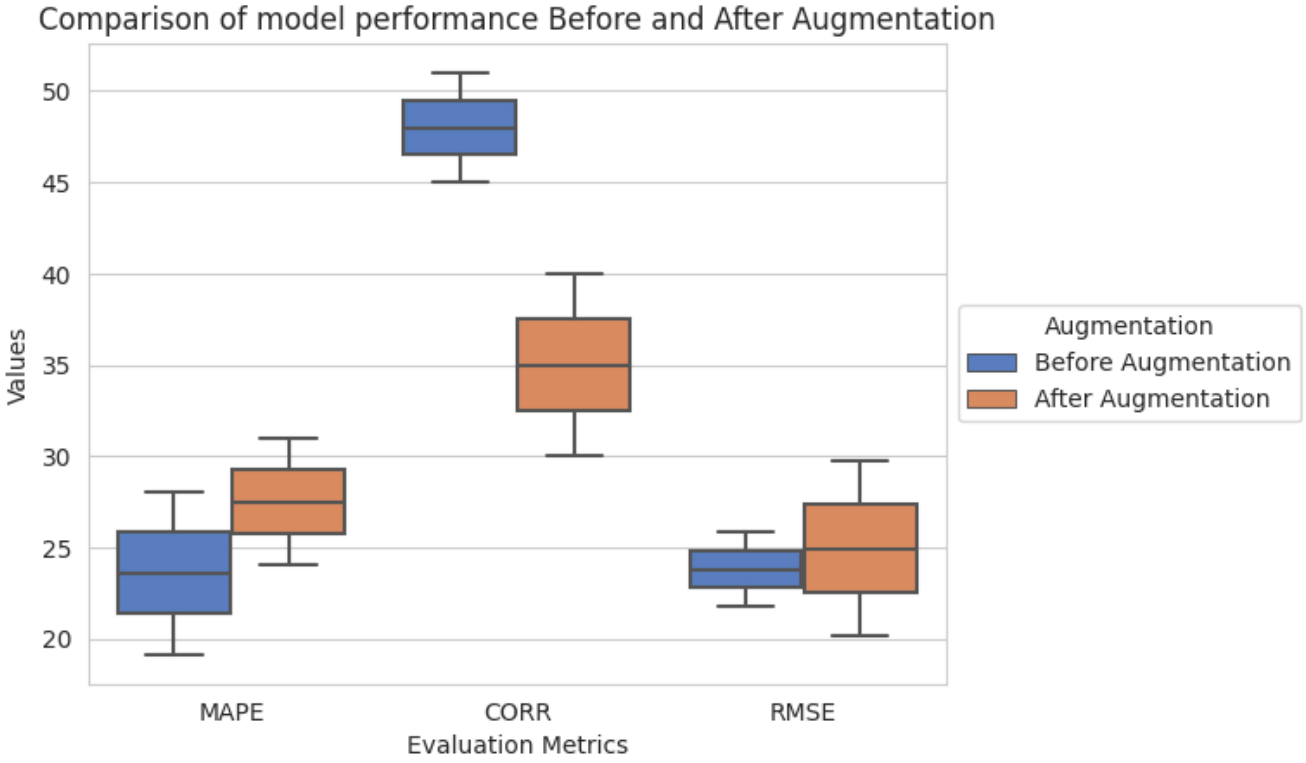

Figure 4.8: Box plots comparing the performance metrics (MAPE, CORR, and RMSE) of the model before and after Data Augmentation. The median is represented by the line in the center of the box, and the whiskers represents the range of the data

Several models were tested with and without image augmentation applied in the training dataset. The performance graph is shown in the Fig 4.8 and the following observations were made:

1. The MAPE value averaged around 23.7 before applying augmentation and increased to an average value of 27.6 after applying augmentation. The maximum MAPE of 28.12 before augmentation was approx. equal to the average value after augmentation which shows the performance degraded after applying augmentation.

2. The CORR value decreased slightly by 0.12, indicating that the model's predictions were less correlated with the actual data after applying augmentation.

3. The RMSE value averaged around 24.1 before augmentation and 24.9 after augmentation which shows that augmentation did not have any significant difference in terms of RMSE. However, the spread of RMSE was more after augmentation.

Overall, the performance of the models seems to have become more varied after augmentation, with larger errors and less correlation with the actual data. A potential reason could be that the augmentation has increased the size of the dataset, but not necessarily improved the quality of the data by adding more diversity. The training data already had

enough information and adding more data led to overfitting. Additionally, upon examination, it was observed that the model's predictions were aligned more closely with the mean of the training dataset after augmentation. This can be attributed to the fact that a significant portion of our training set comprises values within two standard deviations of the mean. Consequently, the inclusion of augmented images primarily from this range of distribution resulted in bias within the model. A potential avenue for future research in this domain is to augment images from the extremities of the distribution, specifically focusing on areas with the lowest and highest yield. Therefore, image augmentation was not included in the final model-building pipeline.

4.4.5 Loss functions

The selection of a loss function in CNN models is crucial as it defines the objective to be optimized during training, influencing the model's ability to learn and make accurate predictions. Two different loss functions mentioned in Section 3.6.4 MSE Loss and L1 Loss were compared. The reason for testing these two loss functions was because they behave differently in case of outliers. The squared error term in the MSE loss gives more weight to larger errors, making it more sensitive to outliers. On the other hand, L1 (MAE) loss is less sensitive to outliers since it treats all errors equally. Following were the insights from the performance comparison graph shown in Fig 4.9

1. It was observed that the RMSE values for L1 Loss (ranging from 20.3 to 25.2) were generally lower than those for models trained using MSE Loss (ranging from 24.9 to 31.56).

2. Similarly, the MAPE values for L1Loss (ranging from 17.9 to 21.67) were lower than those for MSE Loss (ranging from 21.7 to 25.8).

3. This suggests that the model trained with L1 Loss performed better in terms of accuracy and precision compared to the one trained with MSE Loss.

4. By examining the values, it is evident that there is no direct correlation between the RMSE and MAPE values. For instance, while the RMSE value is relatively high for the last data point (31.56), the corresponding MAPE value (22.42) is lower compared to the previous data point (21.7) with a lower RMSE.

The low correlation between these metrics suggests that the RMSE and MAPE capture different aspects of the model's performance and should be considered together for a comprehensive evaluation. L1 Loss is overall less sensitive to outliers than MSELoss. The outliers are prominent in the yield prediction task, due to the presence of numerous heterogeneous factors on the fields such as extreme weather events or other unexpected changes in growing conditions. L1 Loss also encourages sparsity in the model's predictions as it reduces model weights to zero for many of the features, indicating that those points have little or no impact on the outcome. Therefore, the robustness of L1Loss was seen more than that of MSE Loss to noise or fluctuations in the data. Moreover, the model

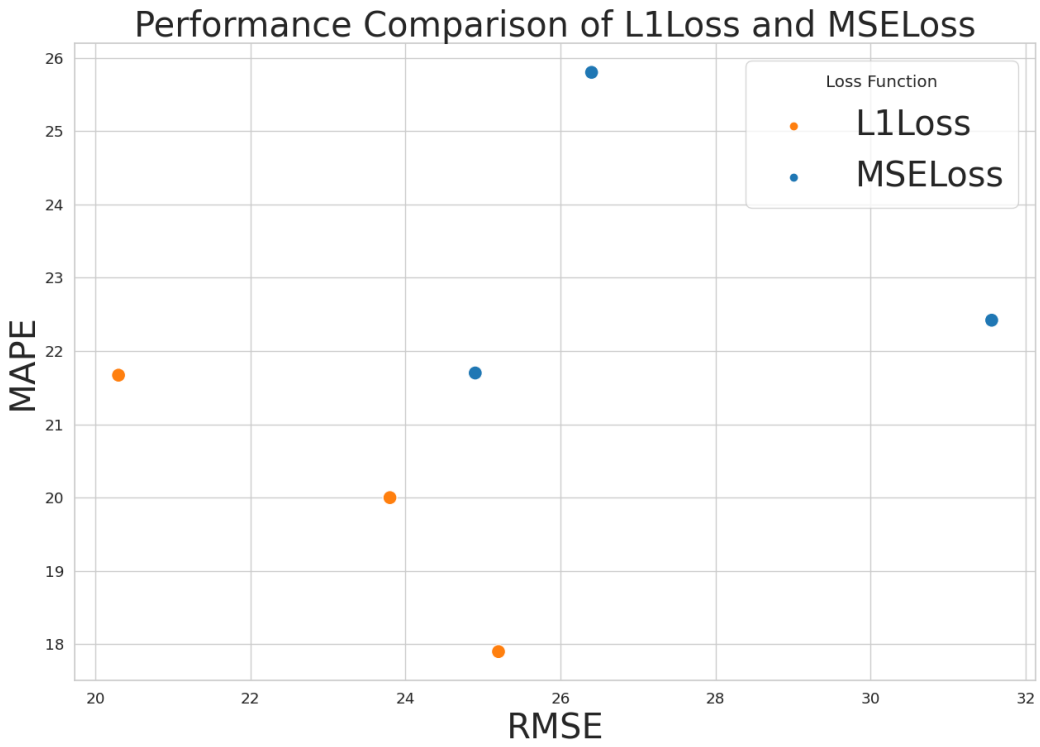

Figure 4.9: Comparison of L1 Loss and MSE Loss of different models using RMSE and MAPE. Each point represents a model's performance using a particular loss function. L1 Loss models are denoted with orange color, while MSELoss models are denoted with blue color

training time was comparatively faster for the models trained using L1Loss making it the ideal choice for this research.

4.4.6 Regularization

The results obtained by comparing the Spatial vs Spatial-Temporal models revealed that, in terms of Mean Absolute Percentage Error, the 3D CNN model exhibited poorer performance than the 2D CNN model, although performance in terms of RMSE was better and promising. Upon analysis, the volume of small errors was found more as compared to large errors in the case of 3D CNN models. Therefore, these models were better in detecting areas with extreme yield values and produced fewer significant errors. Considering the higher prevalence of small errors, regularization techniques were utilized due to the concern that the 3D CNN models, with their enhanced capability to capture complex spatial and temporal features, might be more susceptible to overfitting. Three different regularization techniques (a) Dropout (b) Batch Normalization (c) Early Stopping were tried. All three techniques were applied independently and were not combined. A comparison line chart before and after regularization is shown in Fig 4.10.

1. Before regularization, MAPE values ranged from 18.1 to 22.5. After regularization,

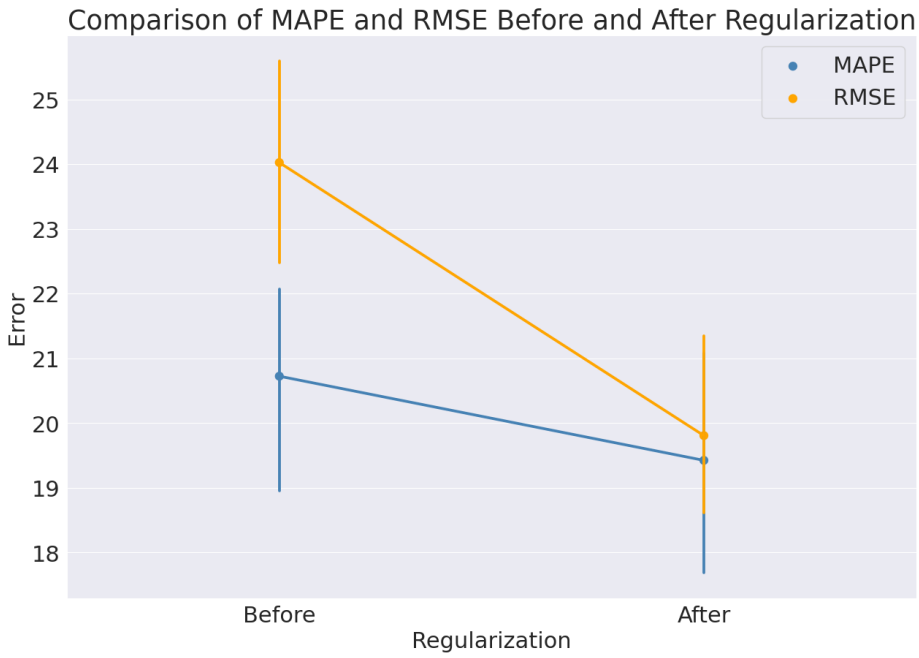

Figure 4.10: The graph shows the impact of regularization on the model performance, as measured by mean absolute percentage error (MAPE) and root mean square error (RMSE). The x-axis indicates Regularization, while the y-axis represents the errors.

MAPE almost remained constant or decreased slightly by 4%. Regularization effectively reduced the relative errors in predictions.

2. RMSE values ranged from 21.77 to 26.33, with the highest value at 26.33 before applying regularization. After regularization, RMSE values remained relatively consistent, ranging from 18.11 to 21.7.

3. RMSE values showed notable improvement after regularization, indicating reduced relative errors. MAPE values remained relatively stable before and after regularization, with slight variations.

The experiments were done on 3D CNN models with 2 phenological stages due to their best performance so far. These results suggest that regularization was effective in bringing the RMSE values down. However, the absolute errors (MAPE) remained relatively consistent before and after regularization. Adding regularization reduced the volume of errors and improved the model generalization slightly. Overall, the results show the effectiveness of regularization in improving the model's predictions by reducing the relative errors.

4.4.7 Correlation Analysis

Correlation analysis is a statistical technique used to study the relationship between two variables. Here, this analysis was done between the performance of the trained model on

the train set and test sets from different fields. This analysis was done to understand which fields are correlated according to the model prediction. When a particular train and test field have a high correlation value, it indicates that training on that specific field captures some spatial information from the test field also, implying that the two fields share some common patterns. These patterns may arise from several factors, such as weather conditions, geographical location, and farming techniques among others.

A correlation matrix was created for this analysis. This is a table that displays the correlation coefficients between multiple variables. In this case, the variables are the predicted yield values by the models where each value represents the correlation value between the fields. The following insights were drawn out from the heatmap shown in Fig 4.11:

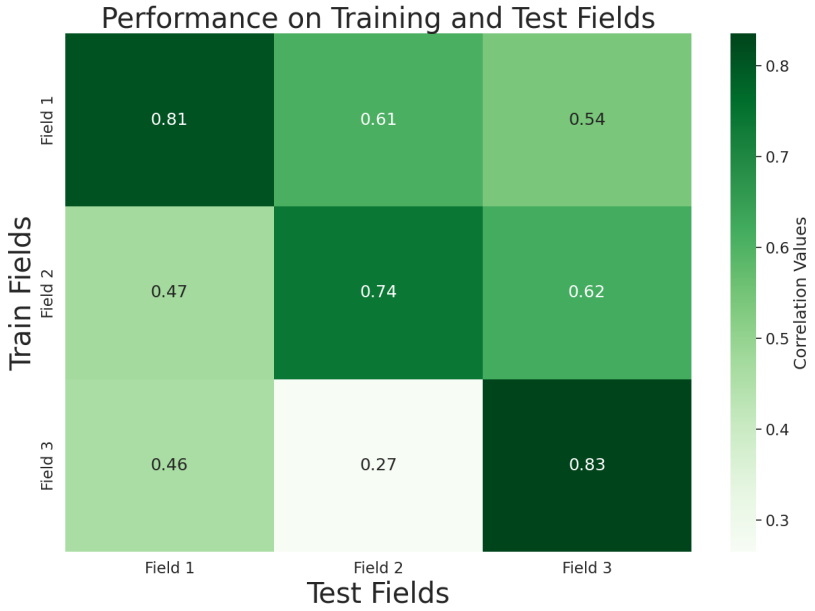

Figure 4.11: A heatmap showing the correlation coefficients between different combinations of training and testing fields for yield prediction. The rows and columns represent the training and testing fields, respectively

1. The diagonal values indicate instances when the train and test fields were identical, and they tend to be closer to 1. This suggests that the models adequately captured the patterns in the training data.

2. The data tiles present in Field 3 fit the training data the best, with the highest correlation coefficient of 0.83, but perform poorly on the test set. We can speculate that the spatial patterns and environmental factors in Field 3 may significantly differ from the other fields, leading to poor generalization. Another possibility is the number of data points for Field 3 is 1402, which is 32.11% and 39.35% less than data tiles in Field 1 and Field 2, respectively.

3. Training on Field 1 showed the best performance, where it was able to capture a correlation of 0.61 and 0.54 with Field 2 and Field 3, respectively. One reason

could be that the Field 1 data contains spatial patterns more representative of the other fields. Another reason is a large and more diverse dataset that leads to better generalization and performance on unseen data from fields 2 and field 3.

Based on the correlation matrix analysis, it can be concluded that training on Field 1 data showed the best performance on all three fields as compared to training on the other two fields. Additionally, Field 2 and Field 3 showed relatively similar performance on the test fields, with slightly better correlation values for Field 3. This shows that the data from Field 1 is enough to predict a yield that would decently correlate with Field 2 and Field 3. This could save the model computation cost of feeding data from all the fields.

4.4.8 Hyperparameter tuning

Hyperparameter tuning is the process of selecting the optimal values of hyperparameters for a machine learning/ deep learning model. In contrast to parameters that are learned from the data, hyperparameters are set by the user before the training begins. Hyperparameters mentioned in Table 3.4 were tweaked to find the optimal performance on the test set. Due to the complexity of the 3D CNN models and training time, the parameters were tweaked manually and other simultaneous experiments were also performed. The hyperparameter tuning was done by visualizing the train and valid loss at each epoch during training and the performance of the model on the test set.

Based on the above experiments the techniques that were employed before hyperparameter tuning were (a) Regularization (b) 3D CNN model (c) L1Loss (d) Adam optimizer (d) Training data from Field 1 and Testing data from Field 2 and 3 (e) Medium Resolution Images (256 X 256 Pixels) (f) Data from Early and Mid Phenological Stages. Then the following steps were taken to tune the hyperparameters and experiments are shown in table 4.3

1. The first step was to define the hyperparameters to be tested and their range. The initial values and hyperparameters are described in Table 3.4. The main tuned parameters included Batch Size, Number of epochs, and the learning rate.

2. A baseline model was defined with a batch size of 16, the number of epochs set to 10, and a high learning rate of 0.1.

3. After training, a low learning rate of 0.001 was tested. This time model overfitted the training set and a low bias and high variance situation was noticed. The learning rate was therefore increased to 0.01, which gave a better performance with RMSE 18.27 and MAPE 13.12 on the validation Set.

4. After finding the optimal learning rate, the batch size was tweaked. It determines the number of samples used in each iteration of the training process. The batch size was changed to 128, which was the largest possible range value. The larger batch size resulted in faster convergence at the cost of its performance on the validation set

Batch	LR	Epochs	CNN Blocks	Train RMSE	Valid RMSE	Train MAPE	Valid MAPE
16	0.1	10	3	15.51	20.51	10.17	14.47
16	0.001	10	3	23.93	28.93	8.64	22.83
16	0.01	10	3	13.27	18.27	10.92	13.12
128	0.01	10	3	20.14	25.14	9.33	20.29
64	0.01	10	3	11.78	16.78	9.78	15.64
32	0.01	10	3	13.36	18.36	7.56	19.08
64	0.01	10	4	21.09	26.09	8.21	24.51
64	0.01	8	5	NaN	NaN	NaN	NaN
64	0.01	8	2	9.63	12.63	7.24	11.18

Table 4.3: The above table shows the changes in the Training and the Validation set by changing hyperparameters where Batch is Batch size and LR is Learning Rate

with RMSE 25.14 and MAPE 20.29. Therefore, testing was done on both 32 and 64 batch sizes and batch size of 64 gave the best performance.

5. After finding the learning rate and batch size, the number of epochs and the model architecture were tuned together. As more layers were added to the model, it was noticed that in fewer epochs signs of overfitting were noticed on the Validation set as shown in Fig 4.12 with RMSE 26.09 and MAPE 24.51. The too-complex model also produced a NaN correlation coefficient and just a single predicted value in some cases. Therefore, 2 CNN blocks (Convolution, Pooling, and ReLU) were fixed for 3D CNN models and the number of epochs was tuned. It was found that the existing model architecture performed best with 8 epochs with RMSE 12.63 and MAPE 11.18.

The final tuned model did not make a significant improvement in performance on the test set. The test set showed a MAPE of 15.18 and an RMSE of 17.63, making it the best performance recorded after performing all the experiments and applying different optimization techniques.

4.4.9 Error Analysis

Error analysis is the process of analyzing and understanding the errors made by the trained model in predicting corn yield. It involved evaluating the performance of the model by comparing the predicted yield to the actual yield and identifying areas where the model was making mistakes. A visual inspection was done by visualizing the areas where the model's predictions were the worst based on the difference between the actual and the predicted yield values. The polygons with both of these values were plotted on the field's raw orthomosaics as shown in Fig 4.13 to understand the root cause of why the model is making large errors for these tiles. Upon, analysis two things were found:

1. There were places where the model could not predict regions of low yields. A tile located at the bottom of Fig. 4.13 has 68 as the actual yield value, indicating low

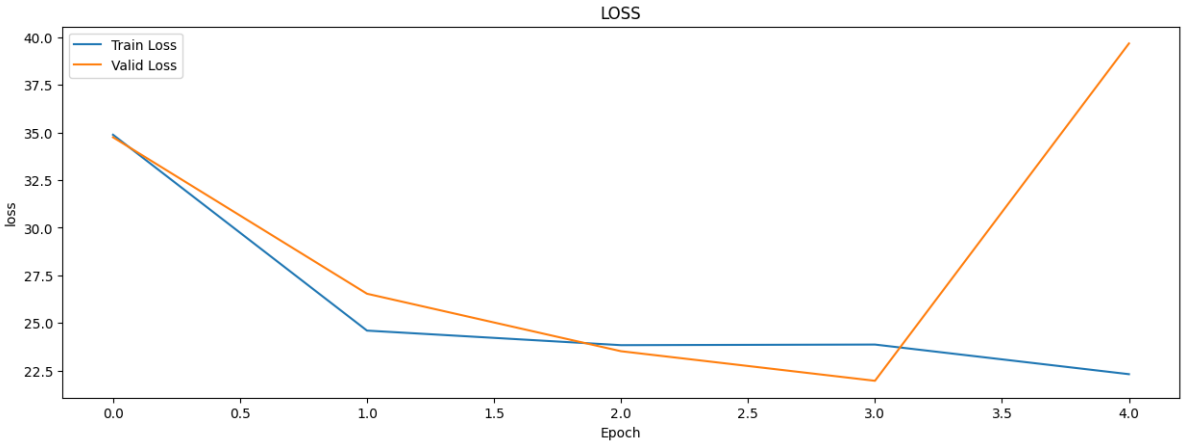

Figure 4.12: A line chart showing the model's train and valid loss during training. The x-axis shows the epochs and the y-axis represents the loss value. This behavior of the model was noticed for a complex architecture where overfitting was noticed in the 3rd epoch.

yield in that area. However, the model's prediction for that same area was 158.25 which is significantly higher than the actual yield value. Upon carefully looking at the tile, some parts of the high yield area and a random tree were also captured in the spatial dimension of the tile based on which model is giving high values. However, the model performed very well in the areas of high and medium yield values, which is in line with the fact that more tiles from medium and high yield were supplied to the model during the training phase.

2. For certain areas, the model predicted high yields, which visually appear to be accurate. However, the actual yield values were found to be lower than expected. For instance, the tile on the extreme left of Fig 4.13 shows 90 as the actual yield. However, the model predicted 171, an area of high yield. Upon analyzing the yield distribution in Field 2, it was found that a yield value of 90 corresponds to a low yield area. Therefore, the actual low yield value of this tile appears erroneous. Although, visually this area appears to be a region of high yield, which aligns with the model's prediction.

After Error analysis, it was concluded that more optimized post-processing techniques for filtering out erroneous yield measurements are required. Removal of these points by visually analyzing should also be a part of the processing pipeline. The variogram fitted during the interpolation process induced an RMSE of 113.51, which can be further reduced. Moreover, the size of tiles could also be reduced from 9X9m to a lower area. Although, the model shows good performance in capturing the areas of medium and high yield, however, the performance was comparatively poor, in the case of low-yield areas.

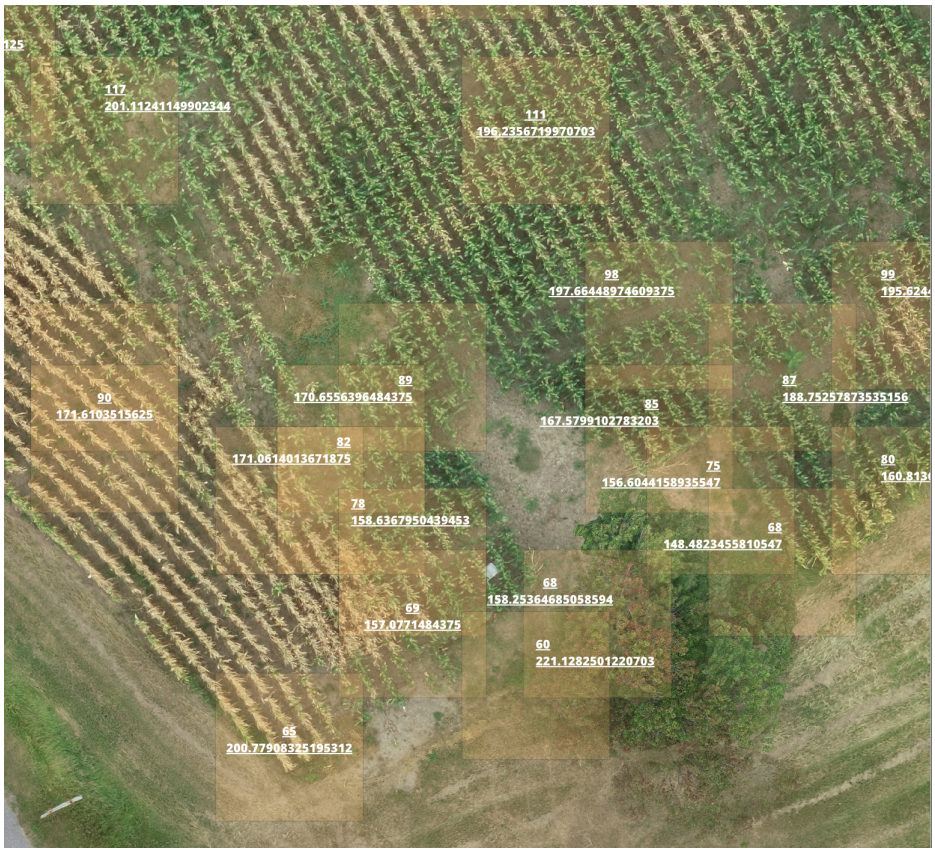

Figure 4.13: The image shows a section of Field 2 with square polygons of size 9X9 m that were extracted for model training. Each polygon displays two values - the top value represents the actual yield, while the bottom value corresponds to the predicted yield.

4.5 Experimentation conclusion

After conducting all the experiments mentioned above and error analysis, the best performance was achieved for the 3D CNN model where two phenological stages (Early and Mid) were used as temporal dimensions. This shows that including the information on crop growth over time helps the model to generalize better on the test set. The Early and Mid growing seasons displayed a stronger correlation with the final yield value, indicating that they provide more significant information compared to the later growing season. Upon visual analysis, the images from the later growing season appear visually cluttered due to the overgrown crops, further supporting this finding. Regularization in turn increased the model performance and brought down the error metrics on the test set. However, augmentation did not prove to have a positive effect on the performance. The performance degraded after augmentation which was because of overfitting, lack of diversity in the original dataset, and noise. Finally, the model was hypertuned which reduced the error on the validation set but did not show significant improvements on the test set. The MAPE was brought down to 15.18% and the RMSE to 17.63 bu/ac by using a 3D CNN model trained on Field 1. Conversely, the model achieved the best performance when the tile

resolution was set to 256 x 256 pixels. Hence, it is possible to reduce the overall image resolution by half and achieve similar/better results, particularly for this yield prediction task. However, such a reduction in resolution could have adverse effects on potential use cases in the future, such as crop health and pest detection.

The results suggest that yield prediction may be more accurate when using drone imagery during the early stages of crop growth (June-July) compared to later stages. This may be because crop growth is more predictable during the early stages when plants are still developing and are less affected by external factors. Moreover, the noise and overgrown crops are less in case of early and mid phenological stages. Following are some insights that can further improve the model performance in future research:

1. Yield errors were introduced during the Kriging process. Therefore, a more advanced method of interpolation should be used to reduce the RMSE while fitting the variogram.

2. In the error analysis, it was seen that the tiles sometimes also capture irregular areas of both high and low yields in the spatial resolution therefore, the tile size that was set to 9 X 9 m can be reduced further.

3. Segmentation or edge detection can employed to detect the crop segments and get rid of extra noise such as trees.

4. More tiles belonging to areas of low yield should be supplied to the model during training to add more diversity to the dataset.

5. The temporal dimension can be increased further by incorporating more images from the phenological stages.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In conclusion, this research demonstrates the potential of using drone imagery and spatial-temporal deep learning models for predicting crop yield. The study has shown that RGB drone imagery can provide valuable information about crop health and growth, which can be effectively used to predict yield. The study utilizes high quality 8K images captured from farms based in Ottawa . Four key steps were taken to develop the yield prediction model: data acquisition, data processing, model development, and optimization. Data Acquisition was done by Trimble yield monitoring device and drones by IndroRobotics. Experts from IndroRobotics team captured the field images for the entire growing season using drones equipped with high-resolution cameras that captured detailed images of fields. This helped to tackle challenges in the existing research such as missing temporal dimension, low-resolution images, limited flight time, image processing, and regulatory issues described in the Section 2.4.1. The second step, data processing, involved pre-processing the raw data which included removing the erroneous data points from the yield by using an end-to-end post-processing pipeline that consists of several existing techniques combined with unsupervised machine learning algorithms. On the other hand, the raw orthomosaics were processed and cropped into smaller tiles of size 9 X 9 m. Finally, the yield and image tiles were fused together using data fusion to create a yield map. Each tile represents a small section of the field and the corresponding yield for that area. The next step, involved experimenting with several deep learning convolution neural network models that can capture the spatial and temporal relationship from these tiles to predict yield. Two classes of model architecture one with 2D Kernels and the other with 3D kernels were tested and compared. The models were evaluated based on metrics including, Root Mean Squared Error (RMSE), Correlation Coefficient (CORR), and Mean Absolute Percentage Error (MAPE). The final step involved optimizing the model performance using techniques such as regularization, augmentation, and hyperparameter tuning.

Several different analyses mentioned in section 4 were done to uncover how different aspects affect the model performance and how these insights can be useful to the farmers to improve production and save cost. The study involved analyzing the correlation between

yield data from multiple fields. Based on the results, it was found that utilizing training tiles from Field 1 can produce yield predictions that are 0.57 correlated with those of Field 2 and Field 3. As a result, this approach can help save on model computation time and training resources by utilizing training data from Field 1 only. The phenological stage analysis revealed that images from the early (June 20) and mid (July 18) growing seasons have the best performance and adding the later growing season (August) degrades the performance. Overall the spatial-temporal model trained on 2 growth stages (Early and Mid) outperformed other proposed and was able to achieve a Mean Absolute Percentage Error (MAPE) of 14.13 and a Root Mean Squared Error (RMSE) of 13.18.

The results from this study have significant implications in improving farm productivity and reducing costs. The results show that (a) It may not be necessary to the train model on all the fields data which can save computation costs (b) Images from the early growing season and mid growing season can lead to the best yield prediction outcomes, therefore it might not be necessary to take images from other stages that can bring down the image acquisition costs (c) The image resolution can be brought down to half, that can save costs of using a very high-resolution camera but this might have an implication on future use cases (d) The yield prediction model can predict the end-of-season yield in advance with good accuracy so that the farmers can take critical timely farming decisions (e) Earlier researches relied on using features derived manually such NDVI from the MS images however the proposed deep CNN model in this research can automatically extract features utilizing only the RGB imagery and achieving a good performance

5.2 Future Work

The proposed model shows promising results in predicting crop yield using RGB drone imagery as input and spatial-temporal deep learning networks, however, there is still room for further improvement and expansion of the research.

1. Incorporating additional input data: The model in this study utilizes solely RGB drone imagery for its training data. However, there is potential to integrate other sources of information, such as weather data, soil characteristics, farming practices, and crop management techniques. Furthermore, the incorporation of multi-spectral data can offer the model supplementary insights that are not available through RGB images alone.

2. Prediction yield for smaller portion: Currently, the model is capable of predicting yield for a single tile of size 9 X 9 m. However, the combination of encoder and decoder architecture has the potential to extend the model's capability to predict yield for each individual pixel.

3. Image Segmentation: The segmentation process involves identifying and delineating the areas within the image that correspond to individual crop plants or fields. This helps to get rid of the noise or erroneous points present on the field and improve the yield prediction results.

4. Crop health and disease detection: The goal of this process is to identify areas within a crop field that shows signs of stress or disease so that corrective measures can be taken to prevent yield losses and improve overall crop productivity. Both supervised and unsupervised methods can be used to classify crops as healthy/non-healthy.

5. Adding images from the low yield area can improve the augmentation performance.

6. Exploration of other hybrid architecture that can extract the spatial and temporal relationships such as CNN-LSTM.

References

[1] "Agexpert field," https://www.agexpert.ca/en/products/field.html, accessed: 2023-04-25.

[2] "albumentations image augmentation," https://albumentations.ai/, accessed: 2023-03-24.

[3] "Artificial neural networks," https://www.javatpoint.com/artificial-neural-network, accessed: 2023-02-10.

[4] "Autel evo ii drone," https://droneshopcanada.ca/collections/buy-autel-robotics-in-canada/products/copy-of-autel-evo-ii, accessed: 2023-02-20.

[5] "Constrained application protocol," https://datatracker.ietf.org/doc/html/rfc7252, accessed: 2023-02-04.

[6] "Evo ii gimbal camera," https://autel-robotics.myshopify.com/products/drones-accessories-evo-ii-gimbal-camera, accessed: 2023-02-20.

[7] "Farmersedge farmcommand," https://farmersedge.ca/farmcommand/, accessed: 2023-04-30.

[8] "Geopandas documentation," https://geopandas.org/en/stable/docs.html, accessed: 2023-02-24.

[9] "Global corn production in 2021-22," https://www.statista.com/statistics/1156213/global-corn-production/, accessed: 2023-01-25.

[10] "Indro robotics," https://indrorobotics.ca/, accessed: 2023-02-01.

[11] "Internet of things," https://www.oracle.com/ca-en/internet-of-things/what-is-iot/, accessed: 2023-02-01.

[12] "Mae vs mape," https://stephenallwright.com/mae-vs-mape/, accessed: 2023-02-17.

[13] "Mean absolute error," https://en.wikipedia.org/wiki/Mean_absolute_error, accessed: 2023-02-15.

[14] "Mean absolute percentage error," https://en.wikipedia.org/wiki/Mean_absolute_percentage_error, accessed: 2023-02-15.

[15] "Message queuing telemetry transport," https://mqtt.org/, accessed: 2023-02-04.

[16] "Predictive ability of machine learning methods for massive crop yield prediction," vol. 12. [Online]. Available: https://revistas.inia.es/index.php/sjar/article/view/4439

[17] "pytorch data utils," https://aigeekprogrammer.com/data-preparation-with-dataset-and-dataloader-in-pytorch/, accessed: 2023-03-03.

[18] "pytorch documentation," https://pytorch.org/, accessed: 2023-03-02.

[19] "pytorch model building example," https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html, accessed: 2023-03-03.

[20] "Qgis documentation," https://docs.qgis.org/3.22/en/docs/training_manual/, accessed: 2023-03-02.

[21] "Raster geospatial data," https://www.neonscience.org/resources/learning-hub/tutorials/dc-raster-data-r, accessed: 2023-02-01.

[22] "Root mean squared error," https://c3.ai/glossary/data-science/root-mean-square-error-rmse/, accessed: 2023-02-16.

[23] "Support vector machine," https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/, accessed: 2023-02-05.

[24] "Vector geospatial data," https://www.neonscience.org/resources/learning-hub/tutorials/intro-vector-data-r, accessed: 2023-02-02.

[25] "Vesper documentation," https://precision-agriculture.sydney.edu.au/wp-content/uploads/2019/08/Vesper_1.6_User_Manual.pdf, accessed: 2023-02-27.

[26] "World's population by 2050," https://www.un.org/en/desa/world-population-projected-reach-98-billion-2050-and-112-billion-2100, accessed: 2023-03-25.

[27] "Precision agriculture—a worldwide overview," *Computers and Electronics in Agriculture*, vol. 36, no. 2, pp. 113–132, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169902000960

[28] "Convolutional neural networks in predicting cotton yield from images of commercial fields," *Computers and Electronics in Agriculture*, vol. 171, p. 105307, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169919319878

[29] A. M. S. Ahamed, N. T. Mahmood, N. Hossain, M. T. Kabir, K. Das, F. Rahman, and R. M. Rahman, "Applying data mining techniques to predict annual yield of major crops and recommend planting different crops in different districts in bangladesh," in *2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. IEEE, 2015, pp. 1–6.

[30] I. Ahmad, U. Saeed, M. Fahad, A. Ullah, M. Habib ur Rahman, A. Ahmad, and J. Judge, "Yield forecasting of spring maize using remote sensing and crop modeling in faisalabad-punjab pakistan," *Journal of the Indian Society of Remote Sensing*, vol. 46, pp. 1701–1711, 2018.

[31] R. Akhter and S. A. Sofi, "Precision agriculture using iot data analytics and machine learning," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 8, Part B, pp. 5602–5618, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157821001282

[32] M. Al-Arab, A. Torres-Rua, A. Ticlavilca, A. Jensen, and M. McKee, "Use of high-resolution multispectral imagery from an unmanned aerial vehicle in precision agriculture," in *2013 IEEE International Geoscience and Remote Sensing Symposium-IGARSS*. IEEE, 2013, pp. 2852–2855.

[33] I. Ali, F. Cawkwell, E. Dwyer, and S. Green, "Modeling managed grassland biomass estimation by using multitemporal remote sensing data—a machine learning approach," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 7, pp. 3254–3264, 2016.

[34] J. Antorán, J. U. Allingham, and J. M. Hernández-Lobato, "Depth uncertainty in neural networks," 2020. [Online]. Available: https://arxiv.org/abs/2006.08437

[35] Aparna, Y. Bhatia, R. Rai, V. Gupta, N. Aggarwal, and A. Akula, "Convolutional neural networks based potholes detection using thermal imaging," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 3, pp. 578–588, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157818312837

[36] M. Arias, M. Á. Campo-Bescós, and J. Álvarez-Mozos, "Crop classification based on temporal signatures of sentinel-1 observations over navarre province, spain," *Remote Sensing*, vol. 12, no. 2, p. 278, 2020.

[37] S. Arslan and T. S. Colvin, "An evaluation of the response of yield monitors and combines to varying yields," *Precision Agriculture*, vol. 3, pp. 107–122, 2002.

[38] ——, "Grain yield mapping: Yield sensing, yield reconstruction, and errors," *Precision Agriculture*, vol. 3, pp. 135–154, 2002.

[39] A. Baggio, "Wireless sensor networks in precision agriculture," in *ACM workshop on real-world wireless sensor networks (REALWSN 2005), Stockholm, Sweden*, vol. 20. Citeseer, 2005, pp. 1567–1576.

[40] F. H. R. Baio, D. C. Santana, L. P. R. Teodoro, I. C. d. Oliveira, R. Gava, J. L. G. de Oliveira, C. A. d. Silva Junior, P. E. Teodoro, and L. S. Shiratsuchi, "Maize yield prediction with machine learning, spectral variables and irrigation management," *Remote Sensing*, vol. 15, no. 1, p. 79, 2022.

[41] J. E. Ball, D. T. Anderson, and C. S. Chan, "A comprehensive survey of deep learning in remote sensing: Theories, tools and challenges for the community," *CoRR*, vol. abs/1709.00308, 2017. [Online]. Available: http://arxiv.org/abs/1709.00308

[42] A. Barbosa, R. Trevisan, N. Hovakimyan, and N. F. Martin, "Modeling yield response to crop management using convolutional neural networks," *Computers and Electronics in Agriculture*, vol. 170, p. 105197, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169919308543

[43] M. S. Basir, M. Chowdhury, M. N. Islam, and M. Ashik-E-Rabbani, "Artificial neural network model in predicting yield of mechanically transplanted rice from transplanting parameters in bangladesh," *Journal of Agriculture and Food Research*, vol. 5, p. 100186, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666154321000880

[44] W. D. Batchelor, B. Basso, and J. O. Paz, "Examples of strategies to analyze spatial and temporal yield variability using crop models," *European Journal of Agronomy*, vol. 18, no. 1-2, pp. 141–158, 2002.

[45] J. S. Bates, C. Montzka, M. Schmidt, and F. Jonard, "Estimating canopy density parameters time-series for winter wheat using uas mounted lidar," *Remote Sensing*, vol. 13, no. 4, 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/4/710

[46] J. P. Beal and L. Tian, "Time shift evaluati on to improve yield map quality," *Applied Engineering in Agriculture*, vol. 17, no. 3, p. 385, 2001.

[47] A. Beck, S. Searcy, and J. Roades, "Yield data filtering techniques for improved map accuracy," *Applied engineering in Agriculture*, vol. 17, no. 4, p. 423, 2001.

[48] M. Belgiu and L. Drăguţ, "Random forest in remote sensing: A review of applications and future directions," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 114, pp. 24–31, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271616000265

[49] H. R. Bin Abdul Rahim, M. Q. Bin Lokman, S. W. Harun, G. L. Hornyak, K. Sterckx, W. S. Mohammed, and J. Dutta, "Applied light-side coupling with optimized spiral-patterned zinc oxide nanorod coatings for multiple optical channel alcohol vapor sensing," *Journal of Nanophotonics*, vol. 10, no. 3, pp. 036 009–036 009, 2016.

[50] S. Blackmore and M. Moore, "Remedial correction of yield map data," *Precision agriculture*, vol. 1, 2003.

[51] P. Bose, N. K. Kasabov, L. Bruzzone, and R. N. Hartono, "Spiking neural networks for crop yield estimation based on spatiotemporal analysis of image time series," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 11, pp. 6563–6573, 2016.

[52] A. Brenning, H. Piotraschke, and P. Leithold, "Geostatistical analysis of on-farm trials in precision agriculture," *GEOSTATS*, pp. 1131–1136, 2008.

[53] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104.

[54] M. Breunig, P. E. Bradley, M. Jahn, P. Kuper, N. Mazroob, N. Rösch, M. Al-Doori, E. Stefanakis, and M. Jadidi, "Geospatial data management research: Progress and future directions," *ISPRS International Journal of Geo-Information*, vol. 9, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2220-9964/9/2/95

[55] Y. Chen, C. Li, P. Ghamisi, C. Shi, and Y. Gu, "Deep fusion of hyperspectral and lidar data for thematic classification," in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2016, pp. 3591–3594.

[56] G. Cheng and J. Han, "A survey on object detection in optical remote sensing images," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 117, pp. 11–28, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271616300144

[57] H. Cheng, L. Damerow, Y. Sun, and M. Blanke, "Early yield prediction using image analysis of apple fruit and tree canopy features with neural networks," *Journal of Imaging*, vol. 3, no. 1, 2017. [Online]. Available: https://www.mdpi.com/2313-433X/3/1/6

[58] T. Chou, *Precision: Principles, Practices and Solutions for the Internet of Things*. Lulu Press, Inc, 2016.

[59] S. Chung, K. Sudduth, and S. Drummond, "Determining yield monitoring system delay time with geostatistical and data segmentation approaches," *Transactions of the ASAE*, vol. 45, no. 4, p. 915, 2002.

[60] A. Cilfone, L. Davoli, L. Belli, and G. Ferrari, "Wireless mesh networking: An iot-oriented perspective survey on relevant technologies," *Future Internet*, vol. 11, no. 4, p. 99, 2019.

[61] B. A. Cruden, D. Prabhu, and R. Martinez, "Absolute radiation measurement in venus and mars entry conditions," *Journal of Spacecraft and Rockets*, vol. 49, no. 6, pp. 1069–1079, 2012.

[62] S. K. Datta, C. Bonnet, and N. Nikaein, "An iot gateway-centric architecture to provide novel m2m services," pp. 514–519, 2014.

[63] L. Deng and X. Li, "Machine learning paradigms for speech recognition: An overview," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 5, pp. 1060–1089, 2013.

[64] M. Dholu and K. Ghodinde, "Internet of things (iot) for precision agriculture application," in *2018 2nd International conference on trends in electronics and informatics (ICOEI)*. IEEE, 2018, pp. 339–342.

[65] A. Dobermann and J. Ping, "Geostatistical integration of yield monitor data and remote sensing improves yield maps," *Agronomy journal*, vol. 96, no. 1, pp. 285–297, 2004.

[66] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, vol. 503, pp. 92–108, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222008426

[67] M. Egmont-Petersen, D. de Ridder, and H. Handels, "Image processing with neural networks—a review," *Pattern Recognition*, vol. 35, no. 10, pp. 2279–2301, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320301001789

[68] M. El-Mekawy, A. Östman, and I. Hijazi, "A unified building model for 3d urban gis," *ISPRS International Journal of Geo-Information*, vol. 1, no. 2, pp. 120–145, 2012. [Online]. Available: https://www.mdpi.com/2220-9964/1/2/120

[69] Y. Everingham, C. Smyth, and N. Inman-Bamber, "Ensemble data mining approaches to forecast regional sugarcane crop production," *Agricultural and Forest Meteorology*, vol. 149, no. 3-4, pp. 689–696, 2009.

[70] Y. Everingham, J. Sexton, D. Skocaj, and G. Inman-Bamber, "Accurate prediction of sugarcane yield using a random forest algorithm," *Agronomy for sustainable development*, vol. 36, pp. 1–9, 2016.

[71] farms.com, "Remote sensing," *Precision Agriculture*, vol. 27, no. 2, pp. 1–22, 2017.

[72] S. Fei, M. A. Hassan, Y. Xiao, X. Su, Z. Chen, Q. Cheng, F. Duan, R. Chen, and Y. Ma, "Uav-based multi-sensor data fusion and machine learning algorithm for yield prediction in wheat," *Precision Agriculture*, pp. 1–26, 2022.

[73] J. L. Fernandes, N. F. F. Ebecken, and J. C. D. M. Esquerdo, "Sugarcane yield prediction in brazil using ndvi time series and neural networks ensemble," *International Journal of Remote Sensing*, vol. 38, no. 16, pp. 4631–4644, 2017.

[74] J. Fohringer, D. Dransch, H. Kreibich, and K. Schröter, "Social media as an information source for rapid flood inundation mapping," *Natural Hazards and Earth System Sciences Discussions*, vol. 3, p. 4231–4264, 07 2015.

[75] L. S. Galvao, J. R. dos Santos, D. A. Roberts, F. M. Breunig, M. Toomey, and Y. M. de Moura, "On intra-annual evi variability in the dry season of tropical forest: A case study with modis and hyperspectral data," *Remote Sensing of Environment*, vol. 115, no. 9, pp. 2350–2359, 2011.

[76] N. Gandhi and L. Armstrong, "Applying data mining techniques to predict yield of rice in humid subtropical climatic zone of india," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2016, pp. 1901–1906.

[77] N. Gandhi, L. J. Armstrong, O. Petkar, and A. K. Tripathy, "Rice crop yield prediction in india using support vector machines," in *2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2016, pp. 1–5.

[78] N. Gandhi, O. Petkar, and L. J. Armstrong, "Rice crop yield prediction using artificial neural networks," in *2016 IEEE Technological Innovations in ICT for Agriculture and Rural Development (TIAR)*, 2016, pp. 105–110.

[79] Gangadhar, "Crop yield and rainfall prediction in tumakuru district using machine learning," 2019.

[80] K. Gavahi, P. Abbaszadeh, and H. Moradkhani, "Deepyield: A combined convolutional neural network with long short-term memory for crop yield forecasting," *Expert Systems with Applications*, vol. 184, p. 115511, 2021.

[81] J. Geng, J. Fan, H. Wang, X. Ma, B. Li, and F. Chen, "High-resolution sar image classification via deep convolutional autoencoders," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 11, pp. 2351–2355, 2015.

[82] A. A. Gitelson, Y. Gritz, and M. N. Merzlyak, "Relationships between leaf chlorophyll content and spectral reflectance and algorithms for non-destructive chlorophyll assessment in higher plant leaves," *Journal of plant physiology*, vol. 160, no. 3, pp. 271–282, 2003.

[83] J. González-García, R. L. Swenson, and A. Gómez-Espinosa, "Real-time kinematics applied at unmanned aerial vehicles positioning for orthophotography in precision agriculture," *Computers and Electronics in Agriculture*, vol. 177, p. 105695, 2020.

[84] A. Gonzalez-Sanchez, J. Frausto-Solis, and W. Ojeda-Bustamante, "Predictive ability of machine learning methods for massive crop yield prediction," *Spanish Journal of Agricultural Research*, vol. 12, no. 2, pp. 313–328, 2014.

[85] Y. Guo, G. Yin, H. Sun, H. Wang, S. Chen, J. Senthilnath, J. Wang, and Y. Fu, "Scaling effects on chlorophyll content estimations with rgb camera mounted on a uav platform using machine-learning methods," *Sensors*, vol. 20, no. 18, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/18/5130

[86] M. He, X. Li, Y. Zhang, J. Zhang, and W. Wang, "Hyperspectral image classification based on deep stacking network," in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2016, pp. 3286–3289.

[87] M. E. Holzman, F. Carmona, R. Rivas, and R. Niclòs, "Early assessment of crop yield from remotely sensed water stress and solar radiation data," *ISPRS journal of photogrammetry and remote sensing*, vol. 145, pp. 297–308, 2018.

[88] B. Hoyle, M. M. Rau, K. Paech, C. Bonnett, S. Seitz, and J. Weller, "Anomaly detection for machine learning redshifts applied to sdss galaxies," *Monthly Notices of the Royal Astronomical Society*, vol. 452, no. 4, pp. 4183–4194, 2015.

[89] J. Huang, J. L. Gómez-Dans, H. Huang, H. Ma, Q. Wu, P. E. Lewis, S. Liang, Z. Chen, J.-H. Xue, Y. Wu *et al.*, "Assimilation of remote sensing into crop growth models: Current status and perspectives," *Agricultural and forest meteorology*, vol. 276, p. 107609, 2019.

[90] A. Huete, K. Didan, T. Miura, E. P. Rodriguez, X. Gao, and L. G. Ferreira, "Overview of the radiometric and biophysical performance of the modis vegetation indices," *Remote sensing of environment*, vol. 83, no. 1-2, pp. 195–213, 2002.

[91] G. Idoje, T. Dagiuklas, and M. Iqbal, "Survey for smart farming technologies: Challenges and issues," *Computers & Electrical Engineering*, vol. 92, p. 107104, 2021.

[92] E. H. Isaaks, R. M. Srivastava *et al.*, *Applied geostatistics*. Oxford university press New York, 1989, vol. 561.

[93] N. Islam, M. M. Rashid, F. Pasandideh, B. Ray, S. Moore, and R. Kadel, "A review of applications and communication technologies for internet of things (iot) and unmanned aerial vehicle (uav) based sustainable smart farming," *Sustainability*, vol. 13, no. 4, 2021. [Online]. Available: https://www.mdpi.com/2071-1050/13/4/1821

[94] A. Jain, J. Mao, and K. Mohiuddin, "Artificial neural networks: a tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[95] J. H. Jeong, J. P. Resop, N. D. Mueller, D. H. Fleisher, K. Yun, E. E. Butler, D. J. Timlin, K.-M. Shim, J. S. Gerber, V. R. Reddy *et al.*, "Random forests for global and regional crop yield predictions," *PloS one*, vol. 11, no. 6, p. e0156571, 2016.

[96] B. Joernsgaard and S. Halmoe, "Intra-field yield variation over crops and years," *European Journal of Agronomy*, vol. 19, no. 1, pp. 23–33, 2003.

[97] M. D. Johnson, W. W. Hsieh, A. J. Cannon, A. Davidson, and F. Bédard, "Crop yield forecasting on the canadian prairies by remotely sensed vegetation indices and machine learning methods," *Agricultural and Forest Meteorology*, vol. 218-219, pp. 74–84, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168192315007546

[98] C. Ju and H. I. Son, "Multiple uav systems for agricultural applications: Control, implementation, and evaluation," *Electronics*, vol. 7, no. 9, p. 162, 2018.

[99] A. Kamilaris and F. Ostermann, "Geospatial analysis and internet of things in environmental informatics," *arXiv preprint arXiv:1808.01895*, 2018.

[100] A. Karnieli, N. Agam, R. T. Pinker, M. Anderson, M. L. Imhoff, G. G. Gutman, N. Panov, and A. Goldberg, "Use of ndvi and land surface temperature for drought assessment: Merits and limitations," *Journal of climate*, vol. 23, no. 3, pp. 618–633, 2010.

[101] W. Khan, A. Daud, J. A. Nasir, and T. Amjad, "A survey on the state-of-the-art machine learning models in the context of nlp," *kuwait journal of science*, vol. 43, 2016.

[102] S. Khanal, J. Fulton, A. Klopfenstein, N. Douridas, and S. Shearer, "Integration of high resolution remotely sensed data and machine learning techniques for spatial prediction of soil properties and corn yield," *Computers and Electronics in Agriculture*, vol. 153, pp. 213–225, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169918300334

[103] S. Khanal, K. KC, J. P. Fulton, S. Shearer, and E. Ozkan, "Remote sensing in agriculture—accomplishments, limitations, and opportunities," *Remote Sensing*, vol. 12, no. 22, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/22/3783

[104] P. Killeen, I. Kiringa, and T. Yeap, "Corn grain yield prediction using uav-based high spatiotemporal resolution multi-spectral imagery," in *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, 2022, pp. 1054–1062.

[105] D. Kim, M. Kim, and W. Kim, "Wafer edge yield prediction using a combined long short-term memory and feed- forward neural network model for semiconductor manufacturing," *IEEE Access*, vol. 8, pp. 215 125–215 132, 2020.

[106] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[107] L. Li, S. Zhang, and B. Wang, "Plant disease detection and classification by deep learning—a review," *IEEE Access*, vol. 9, pp. 56 683–56 698, 2021.

[108] Z. C. Lipton, "A critical review of recurrent neural networks for sequence learning," *CoRR*, vol. abs/1506.00019, 2015. [Online]. Available: http://arxiv.org/abs/1506.00019

[109] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[110] G. Lyle, B. Bryan, and B. Ostendorf, "Post-processing methods to eliminate erroneous grain yield measurements: review and directions for future development," *Precision agriculture*, vol. 15, pp. 377–402, 2014.

[111] ——, "Post-processing methods to eliminate erroneous grain yield measurements: review and directions for future development," *Precision agriculture*, vol. 15, pp. 377–402, 2014.

[112] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, "Internet of things (iot) security: Current status, challenges and prospective measures," in *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2015, pp. 336–341.

[113] M. Maimaitijiang, V. Sagan, P. Sidike, S. Hartling, F. Esposito, and F. B. Fritschi, "Soybean yield prediction from uav using multimodal data fusion and deep learning," *Remote Sensing of Environment*, vol. 237, p. 111599, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425719306194

[114] Y. Malbéteau, K. Johansen, B. Aragon, S. K. Al-Mashhawari, and M. F. McCabe, "Overcoming the challenges of thermal infrared orthomosaics using a swath-based approach to correct for dynamic temperature and wind effects," *Remote Sensing*, vol. 13, no. 16, 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/16/3255

[115] I. Marcu, C. Voicu, A. M. C. Drăgulinescu, O. Fratu, G. Suciu, C. Balaceanu, and M. M. Andronache, "Overview of iot basic platforms for precision agriculture," in *Future Access Enablers for Ubiquitous and Intelligent Infrastructures: 4th EAI International Conference, FABULOUS 2019, Sofia, Bulgaria, March 28-29, 2019, Proceedings 283*. Springer, 2019, pp. 124–137.

[116] A. P. Marques Ramos, L. Prado Osco, D. Elis Garcia Furuya, W. Nunes Gonçalves, D. Cordeiro Santana, L. Pereira Ribeiro Teodoro, C. Antonio da Silva Junior, G. Fernando Capristo-Silva, J. Li, F. Henrique Rojo Baio, J. Marcato Junior, P. Eduardo Teodoro, and H. Pistori, "A random forest ranking approach to predict yield in maize with uav-based vegetation spectral indices," *Computers and Electronics in Agriculture*, vol. 178, p. 105791, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169920319591

[117] Y. Mekonnen, S. Namuduri, L. Burton, A. Sarwat, and S. Bhansali, "Review—machine learning techniques in wireless sensor network based precision agriculture," *Journal of The Electrochemical Society*, vol. 167, p. 037522, 01 2020.

[118] M. Michałowska and J. Rapiński, "A review of tree species classification based on airborne lidar data and applied classifiers," *Remote Sensing*, vol. 13, no. 3, 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/3/353

[119] S. Moges, W. Raun, R. Mullen, K. Freeman, G. Johnson, and J. Solie, "Evaluation of green, red, and near infrared bands for predicting winter wheat biomass, nitrogen

uptake, and final grain yield," *Journal of plant nutrition*, vol. 27, no. 8, pp. 1431–1441, 2005.

[120] S. Mourtzinis, F. J. Arriaga, K. S. Balkcom, and B. V. Ortiz, "Corn grain and stover yield prediction at r1 growth stage," *Agronomy journal*, vol. 105, no. 4, pp. 1045–1050, 2013.

[121] S. Mukherjee and G. Biswas, "Networking for iot and applications using existing communication technology," *Egyptian Informatics Journal*, vol. 19, no. 2, pp. 107–127, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110866517301093

[122] K. Naseer Qureshi, S. Din, G. Jeon, and F. Piccialli, "An accurate and dynamic predictive model for a smart m-health system using machine learning," *Information Sciences*, vol. 538, pp. 486–502, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025520306113

[123] P. Nevavuori, N. Narra, P. Linna, and T. Lipping, "Crop yield prediction using multitemporal uav data and spatio-temporal deep learning models," *Remote Sensing*, vol. 12, no. 23, p. 4000, 2020.

[124] P. Nevavuori, N. Narra, and T. Lipping, "Crop yield prediction with deep convolutional neural networks," *Computers and Electronics in Agriculture*, vol. 163, p. 104859, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169919306842

[125] S. Nezami, E. Khoramshahi, O. Nevalainen, I. Pölönen, and E. Honkavaara, "Tree species classification of drone hyperspectral and rgb imagery with deep learning convolutional neural networks," *Remote Sensing*, vol. 12, no. 7, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/7/1070

[126] L. H. Nguyen, J. Zhen, Z. Lin, H. Du, Z. Yang, W. Guo, and F. Jin, "Spatial-temporal multi-task learning for within-field cotton yield prediction," *CoRR*, vol. abs/1811.06665, 2018. [Online]. Available: http://arxiv.org/abs/1811.06665

[127] M. Oliver and R. Webster, "A tutorial guide to geostatistics: Computing and modelling variograms and kriging," *Catena*, vol. 113, pp. 56–69, 2014.

[128] B. Omoniwa, R. Hussain, M. A. Javed, S. H. Bouk, and S. A. Malik, "Fog/edge computing-based iot (feciot): Architecture, applications, and research issues," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4118–4149, 2018.

[129] L. P. Osco, J. Marcato Junior, A. P. Marques Ramos, L. A. de Castro Jorge, S. N. Fatholahi, J. de Andrade Silva, E. T. Matsubara, H. Pistori, W. N. Gonçalves, and J. Li, "A review on deep learning in uav remote sensing," *International Journal of Applied Earth Observation and Geoinformation*, vol. 102, p. 102456, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S030324342100163X

[130] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *CoRR*, vol. abs/1511.08458, 2015. [Online]. Available: http://arxiv.org/abs/1511.08458

[131] E. Panagiotou, G. Chochlakis, L. Grammatikopoulos, and E. Charou, "Generating elevation surface from a single rgb remotely sensed image using deep learning," *Remote Sensing*, vol. 12, no. 12, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/12/2002

[132] X. E. Pantazi, D. Moshou, T. Alexandridis, R. L. Whetton, and A. M. Mouazen, "Wheat yield prediction using machine learning and advanced sensing techniques," *Computers and electronics in agriculture*, vol. 121, pp. 57–65, 2016.

[133] M. Paul, S. K. Vishwakarma, and A. Verma, "Analysis of soil behaviour and prediction of crop yield using data mining approach," in *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, 2015, pp. 766–771.

[134] J. Ping and A. Dobermann, "Processing of yield map data," *Precision Agriculture*, vol. 6, pp. 193–212, 2005.

[135] R. E. Plant, "Site-specific management: the application of information technology to crop production," *Computers and electronics in agriculture*, vol. 30, no. 1-3, pp. 9–29, 2001.

[136] C. Poeplau, "Measuring and modelling soil carbon stocks and stock changes in livestock production systems: guidelines for assessment; version 1-advanced copy," 2019.

[137] M. Rahnemoonfar and C. Sheppard, "Real-time yield estimation based on deep learning," in *Commercial + Scientific Sensing and Imaging*, 2017.

[138] C. Rao and Y. Liu, "Three-dimensional convolutional neural network (3D-CNN) for heterogeneous material homogenization," *CoRR*, vol. abs/2002.07600, 2020. [Online]. Available: https://arxiv.org/abs/2002.07600

[139] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[140] T. Robinson and G. Metternicht, "Comparing the performance of techniques to improve the quality of yield maps," *Agricultural Systems*, vol. 85, no. 1, pp. 19–41, 2005.

[141] J. W. Rouse, R. H. Haas, J. A. Schell, D. W. Deering *et al.*, "Monitoring vegetation systems in the great plains with erts," *NASA Spec. Publ*, vol. 351, no. 1, p. 309, 1974.

[142] G. Ruß and A. Brenning, "Data mining in precision agriculture: Management of spatial information," in *Computational Intelligence for Knowledge-Based Systems Design*, E. Hüllermeier, R. Kruse, and F. Hoffmann, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 350–359.

[143] G. Ruß, R. Kruse, M. Schneider, and P. Wagner, "Data mining with neural networks for wheat yield prediction," in *Advances in Data Mining. Medical Applications, E-Commerce, Marketing, and Theoretical Aspects: 8th Industrial Conference, ICDM 2008 Leipzig, Germany, July 16-18, 2008 Proceedings 8*. Springer, 2008, pp. 47–56.

[144] F. Safety, "Smart farming and food safety internet of things applications—challenges for large scale implementations," *Alliance for Internet of Things Innovation (AIOTI): Brussels, Belgium*, 2015.

[145] V. Sagan, M. Maimaitijiang, S. Bhadra, M. Maimaitiyiming, D. R. Brown, P. Sidike, and F. B. Fritschi, "Field-scale crop yield prediction using multi-temporal worldview-3 and planetscope satellite data and deep learning," *ISPRS journal of photogrammetry and remote sensing*, vol. 174, pp. 265–281, 2021.

[146] O. Satir and S. Berberoglu, "Crop yield prediction under soil salinity using satellite derived vegetation indices," *Field crops research*, vol. 192, pp. 134–143, 2016.

[147] U. Shafi, R. Mumtaz, J. García-Nieto, S. A. Hassan, S. A. R. Zaidi, and N. Iqbal, "Precision agriculture techniques and practices: From considerations to applications," *Sensors*, vol. 19, no. 17, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/17/3796

[148] M. Shahbandeh, "Corn industry worldwide - statistics facts," *statista*, vol. 27, no. 2, pp. 1–22, 2021.

[149] Y. Shao, J. Ren, and J. Campbell, "Multitemporal remote sensing data analysis for agricultural application," 2018.

[150] S. Shearer, S. Higgins, S. McNeil, G. Watkins, R. Barnhisel, J. Doyle, J. Leach, and J. Fulton, "Data filtering and correction techniques for generating yield maps from multiple combine harvesting systems," in *Proceedings of ASAE Annual International Meeting, Minneapolis, Minnesota, USA, Paper No*, vol. 971034, 1997, pp. 1–7.

[151] S. Shearer, M. Veal, T. Stombaugh, S. Higgins, T. Mueller, C. Dillon, J. Fulton *et al.*, "Post processing correction to improve the accuracy of yield monitor data in grain crops." in *Proceedings of the 7th International Conference on Precision Agriculture and Other Precision Resources Management, Hyatt Regency, Minneapolis, MN, USA, 25-28 July, 2004*. Precision Agriculture Center, University of Minnesota, Department of Soil ..., 2004, pp. 173–186.

[152] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE access*, vol. 7, pp. 53 040–53 065, 2019.

[153] G. Simbahan, A. Dobermann, and J. Ping, "Screening yield monitor data improves grain yield maps," *Agronomy Journal*, vol. 96, no. 4, pp. 1091–1102, 2004.

[154] A. Singh, B. Ganapathysubramanian, A. K. Singh, and S. Sarkar, "Machine learning for high-throughput stress phenotyping in plants," *Trends in plant science*, vol. 21, no. 2, pp. 110–124, 2016.

[155] P. Skobelev, D. Budaev, N. Gusev, and G. Voschuk, "Designing multi-agent swarm of uav for precise agriculture," in *Highlights of Practical Applications of Agents, Multi-Agent Systems, and Complexity: The PAAMS Collection: International Workshops of PAAMS 2018, Toledo, Spain, June 20–22, 2018, Proceedings 16*. Springer, 2018, pp. 47–59.

[156] N. Son, C. Chen, C. Chen, V. Minh, and N. Trung, "A comparative analysis of multitemporal modis evi and ndvi data for large-scale rice yield estimation," *Agricultural and forest meteorology*, vol. 197, pp. 52–64, 2014.

[157] statcan.gc.ca, "Corn: Canada's third most valuable crop," *Canadian Agriculture at a Glance*, vol. 27, no. 2, pp. 1–22, 2011.

[158] K. A. Sudduth and S. T. Drummond, "Yield editor: Software for removing errors from crop yield maps," *Agronomy Journal*, vol. 99, no. 6, pp. 1471–1482, 2007.

[159] R. Sujatha and P. Isakki, "A study on crop yield forecasting using classification techniques," in *2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16)*, 2016, pp. 1–4.

[160] N. Tantalaki, S. Souravlas, and M. Roumeliotis, "Data-driven decision making in precision agriculture: The rise of big data in agricultural systems," *Journal of Agricultural & Food Information*, vol. 20, no. 4, pp. 344–380, 2019.

[161] A. S. Terliksiz and D. T. Altýlar, "Use of deep neural networks for crop yield prediction: A case study of soybean yield in lauderdale county, alabama, usa," in *2019 8th International Conference on Agro-Geoinformatics (Agro-Geoinformatics)*, 2019, pp. 1–4.

[162] L. Thylén, P. Algerbo, A. Giebel *et al.*, "An expert filter removing erroneous yield data." in *Proceedings of the 5th International Conference on Precision Agriculture, Bloomington, Minnesota, USA, 16-19 July, 2000*. American Society of Agronomy, 2000, pp. 1–9.

[163] D. C. Tsouros, S. Bibi, and P. G. Sarigiannidis, "A review on uav-based applications for precision agriculture," *Information*, vol. 10, no. 11, 2019. [Online]. Available: https://www.mdpi.com/2078-2489/10/11/349

[164] C. J. Tucker, B. Holben, J. Elgin Jr, and J. Mcmurtrey III, "Relationship of spectral data to grain yield variation," *Photogrammetric Engineering and Remote Sensing*, vol. 46, 1980.

[165] K. Uto, H. Seki, G. Saito, and Y. Kosugi, "Characterization of rice paddies by a uav-mounted miniature hyperspectral sensor system," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6, no. 2, pp. 851–860, 2013.

[166] T. van Klompenburg, A. Kassahun, and C. Catal, "Crop yield prediction using machine learning: A systematic literature review," *Computers and*

Electronics in Agriculture, vol. 177, p. 105709, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169920302301

[167] D. Vasisht, Z. Kapetanovic, J. Won, X. Jin, R. Chandra, S. N. Sinha, A. Kapoor, M. Sudarshan, and S. Stratman, "Farmbeats: An iot platform for data-driven agriculture." in *NSDI*, vol. 17, 2017, pp. 515–529.

[168] S. Verma, R. Gala, S. Madhavan, S. Burkule, S. Chauhan, and C. Prakash, "An internet of things (iot) architecture for smart agriculture," in *2018 fourth international conference on computing communication control and automation (ICCUBEA)*. IEEE, 2018, pp. 1–4.

[169] A. X. Wang, C. Tran, N. Desai, D. Lobell, and S. Ermon, "Deep transfer learning for crop yield prediction with remote sensing data," in *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*, 2018, pp. 1–5.

[170] S. Wang, J. Zhou, T. Lei, H. Wu, X. Zhang, J. Ma, and H. Zhong, "Estimating land surface temperature from satellite passive microwave observations with the traditional neural network, deep belief network, and convolutional neural network," *Remote Sensing*, vol. 12, no. 17, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/17/2691

[171] G. Warren and G. Metternicht, "Agricultural applications of high-resolution digital multispectral imagery: evaluating within-field spatial variability of canola (brassica napus) in western australia," *Photogrammetric Engineering and Remote Sensing*, vol. 71, no. 5, pp. 595–602, 2005.

[172] R. Whetton, Y. Zhao, S. Shaddad, and A. M. Mouazen, "Nonlinear parametric modelling to study how soil properties affect crop yields and ndvi," *Computers and Electronics in Agriculture*, vol. 138, pp. 127–136, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169916311486

[173] X. Xu, P. Gao, X. Zhu, W. Guo, J. Ding, C. Li, M. Zhu, and X. Wu, "Design of an integrated climatic assessment indicator (icai) for wheat production: A case study in jiangsu province, china," *Ecological Indicators*, vol. 101, pp. 943–953, 2019.

[174] Z. Xu, K. Guan, N. Casler, B. Peng, and S. Wang, "A 3d convolutional neural network method for land cover classification using lidar and multi-temporal landsat imagery," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 144, pp. 423–434, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271618302223

[175] J. Xue and B. Su, "Significant remote sensing vegetation indices: A review of developments and applications," *Journal of sensors*, vol. 2017, 2017.

[176] Y. xue Su, H. Xu, and L. jiao Yan, "Support vector machine-based open crop model (sbocm): Case of rice production in china," *Saudi Journal of Biological Sciences*, vol. 24, no. 3, pp. 537–547, 2017, computational Intelligence

Research Approaches in Bioinformatics and Biocomputing. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319562X17300335

[177] C. Zhang and J. M. Kovacs, "The application of small unmanned aerial systems for precision agriculture: a review," *Precision agriculture*, vol. 13, pp. 693–712, 2012.

[178] L. Zhang, Z. Zhang, Y. Luo, J. Cao, and F. Tao, "Combining optical, fluorescence, thermal satellite, and environmental data to predict county-level maize yield in china using machine learning approaches," *Remote Sensing*, vol. 12, no. 1, p. 21, 2019.

[179] S. Zhang and G. Zhao, "A harmonious satellite-unmanned aerial vehicle-ground measurement inversion method for monitoring salinity in coastal saline soil," *Remote Sensing*, vol. 11, p. 1700, 07 2019.

[180] W. Zhao, T. Li, B. Qi, Q. Nie, and T. Runge, "Terrain analytics for precision agriculture with automated vehicle sensors and data fusion," *Sustainability*, vol. 13, no. 5, 2021. [Online]. Available: https://www.mdpi.com/2071-1050/13/5/2905

[181] Y. Çakır, M. Kırcı, and E. O. Güneş, "Yield prediction of wheat in south-east region of turkey by using artificial neural networks," in *2014 The Third International Conference on Agro-Geoinformatics*, 2014, pp. 1–4.