IOT BASED LOW-COST PRECISION INDOOR FARMING

by

Madhu Lekha Guntaka

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Industrial Engineering

School of Industrial Engineering

West Lafayette, Indiana

August 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Dharmendra Saraswat, Co-chair

School of Agricultural and Biological Engineering

Dr. Vaneet Aggarwal, Co-chair

School of Industrial Engineering

Dr. Vincent Duffy

School of Industrial Engineering

Approved by:

Dr. Abhijit Deshmukh

Dedicated to my family and amazing friends at Purdue for their continuous support and care

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my advisor Prof. Dharmendra Saraswat for his continuous support and belief in me throughout my research, for his patience, motivation, and enthusiasm. His mentoring helped me overcome numerous obstacles through the study and write this thesis. Next, I would like to thank Dr. Petrus Langenhoven for guiding me through the plant growth and helping me with all the facilities needed to make this study possible.

I would also like to thank Dr. Vaneet Aggarwal for agreeing to be a co-chair on my committee and for his insightful comments about the algorithms developed. I offer my sincere appreciation for the perspective added by Dr. Vincent Duffy.

I thank my fellow mates in digital agriculture discovery group : Varun Aggarwal and Aanis Ahmad for helping me with the numerous questions I had and sharing their experiences. I am grateful to Benjamin Hancock for enlightening me the first glance of work in the software and helping me debug codes practically anytime of the day.

My completion of this project could not have been accomplished without the help of two students – Alfonso and Adesh for preparing the seeds and harvesting the plants timely. I also thank Nathan Deppe, the HLA plant growth facility manager for providing inputs in some of the hardware decisions made. In addition, I thank statistical consulting service for their continual services in experimental design and assistance in data analysis.

I would like to acknowledge the AgSEED program for funding the study, department of ABE for providing with a research assistantship to pursue this opportunity and department of HLA for the facilities.

Finally, and most importantly, I am extremely grateful for the continuous support of my family in pursuing my interests and their efforts to understanding my work. A special thanks to all my friends at Purdue, especially Lakshman and Ashwin for all the encouragement and emotional support.

TABLE OF CONTENTS

Contents

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ACRONYM	FULL FORM
AE	Autoencoder
AI	Artificial Intelligence
ANN	Artificial Neural Network
ARIMA	Auto Regressive Integrated Moving Average
CNN	Convolutional Neural Network
CUSUM	Cumulative Sum
DO	Dissolved Oxygen
EC	Electric Conductivity
EE	Elliptic Envelope
FPR	False Positive Rate
GMM	Gaussian Mixture Model
GPIO	General Purpose Input/Output
HBOS	Histogram based outlier score
iForest	Isolation Forest
IoT	Internet of Things
KDE	Kernel Density Estimation
KNN	K-Nearest Neighbors
LOF	Local Outlier Factor
LSTM	Long Short-Term Memory
MCU	Micro Controller Unit
NN	Neural Network
PAR	Photosynthetically Active Radiation
PPFD	Photosynthetic Photon Flux Density
PWM	Pulse Width Modulation
QoS	Quality of Service
RH	Relative Humidity

RMSE	Root Mean Squared Error
RPi	Raspberry Pi
SVM	Support Vector Machine
Temp.	Temperature
TPR	True Positive Rate
WSN	Wireless Sensor Network

ABSTRACT

There is a growing demand for indoor farm management systems that can track plant growth, allow automatic control and aid in real-time decision making. Internet of Thing (IoT)-based solutions are being applied to meet these needs and numerous researchers have created prototypes for meeting specific needs using sensors, algorithms, and automations. However, limited studies are available that report on comprehensive large-scale experiments to test various aspects related to availability, scalability and reliability of sensors and actuators used in low-cost indoor farms. The purpose of this study was to develop a low-cost, IoT devices driven indoor farm as a testbed for growing microgreens and other experimental crops. The testbed was designed using off-the-shelf sensors and actuators for conducting research experiments, addressing identified challenges, and utilizing remotely acquired data for developing an intelligent farm management system. The sensors were used for collecting and monitoring electrical conductivity (EC), pH and dissolved oxygen (DO) levels of the nutrient solution, light intensity, environmental variables, and imagery data. The control of light emitting diodes (LEDs), irrigation pumps, and camera modules was carried out using commercially available components. All the sensors and actuators were remotely monitored, controlled, and coordinated using a cloud-based dashboard, Raspberry Pis, and Arduino microcontrollers. To implement a reliable, real-time control of actuators, edge computing was used as it helped in minimizing latency and identifying anomalies.

Decision making about overall system performance and harvesting schedule was accomplished by providing alerts on anomalies in the sensors and actuators and through installation of cameras to predict yield of microgreens, respectively. A split-plot statistical design was used to evaluate the effect of lighting, nutrition solution concentration, seed density, and day of harvest on the growth of microgreens. This study complements and expands past efforts by other researchers on building a low cost IoT-based indoor farm. While the experience with the testbed demonstrates its real-world potential of conducting experimental research, some major lessons were learnt along the way that could be used for future enhancements.

1. INTRODUCTION

The global farming industry is trying various measures for mitigating impacts due to climate change, dwindling natural resources due to rapid urbanization, and meeting demands to achieve food and nutritional security for a rapidly increasing population. These challenges are being met head-on by farmers through the adoption of advances in agro-technologies that enable practicing of science-based, efficient, and sustainable farming to complement years of traditional farming experience (Astill et al., 2020; Schimmelpfennig, 2016). Advances in the field of machinery and Internet of Things (IoT) devices are helping generate various kinds of sensor data from agricultural operational environments that are creating opportunities for development/application of Artificial Intelligence (AI) methods to interpret and understand this huge amount of data.

Today, there is a growing trend towards the design and establishment of controlled environments for agriculture (CEA), involving automated monitoring and control, to increase the capacity, economic viability, and efficiency of indoor farms (Gómez et al., 2019). Many of the modern indoor farms rely on sensors and controls for specific purposes that inspired other researchers to create new concepts and experimental models of closed-loop systems (Jaiswal et al., 2019; Zamora-Izquierdo et al., 2019). This study created one such closed-loop system equipped with low-cost sensors, controls, and smart monitoring (approximate total cost $10k, for details refer to Appendix A) for an experimental indoor farm, that was used for investigating effect of different treatments on microgreens production.

1.1 Motivation/ Significance

Interest in precision indoor farming is growing due to advances in artificial sources of illumination (Paucek et al., 2020), efficient heating and cooling technologies (Huang and Niu, 2016), water-efficient farming methods (AlShrouf, 2017), and the use of IoT devices for monitoring and controlling various functionalities of the farm (Madushanki et al., 2019) interest is reinforced by a shrinking skilled workforce available for farm work (Zahniser et al., 2018). However, indoor farming is 3-5 times more expensive than traditional farming owing to high initial capital cost, energy, and labor expenses (Admeasure, 2014). The present study was undertaken to

explore development of a low-cost, indoor farm, that uses IoT devices and relies on open-source-based monitoring and control systems because of their promising benefits.

Although, the field of IoT has benefitted from research activities across domains over the past few years, there are several unsolved issues about management and performance of IoT devices and networks. Common issues relate to availability, reliability, scalability, mobility, data confidentiality, security, and compatibility of networks (Khanna and Kaur, 2019). Further the cost of IoT devices, constraint-free communication, correct identification and deployment, and heterogeneity of devices and protocols remains a few identified challenges within an IoT application.

There is a noticeable void in the literature on how to improve and adapt IoT solutions for real-world indoor farms beyond simple prototypes. Direct scaling in most cases implies that the number of hardware components grow very quickly with the size of the system. A careful requirement analysis and integrated system design are indispensable for a viable IoT solution in a real-world scenario. Also, identification and handling of anomalous events are essential to manage sensor/ actuator malfunctions and ensure reliability especially with low-cost sensors. Efficient real-time control poses latency-sensitive requirements that require looking beyond solely cloud-based systems and instead bring in edge computing as sort of a plug-in. But studies related to validation of edge-driven services on operational farms is limited (O'Grady et al., 2019).

Another challenge with deploying IoT solutions in farms is reaching out to remote pockets via cabling and electronics without interfering with regular farm operations. In this context, there has been a growth in usage of images for decision making in farms because of the increased availability of satellite imagery, affordable unmanned aerial vehicles (UAVs), and advancements in AI for extracting valuable information from images. Nevertheless, vision-based techniques for indoor farms still have a lot of room for exploration to help understand the effects of lighting, nutrition, and other environmental conditions (temperature, relative humidity (RH) etc.) on plant growth and yield.

1.2 Problem Statement

Indoor farms are most often located close to urban areas and have access to electricity and a reliable internet connectivity. Availability, affordability (both by the cost of sensors and resource optimization) and reliability of necessary technologies are the barriers to widespread adoption of

IoT devices, especially in a small-to-medium scale experimental indoor farm. Developing a solution with off-the-shelf sensors, implementing valid error handling, alerts, remote monitoring, and running indoor farm operations through deployment of smart logarithms that integrate monitored data for decision making, has a potential to motivate growers in their wide-scale adoption. The rationale of the study and detailed description of the design and implementation has potential to serve as an end-to-end reference for an interested grower or an IoT system developer. The ability to obtain detailed information on plant growth and yield without employing destructive techniques could be particularly interesting as it provides an example for using automation to reduce reliance on labor during critical operations (Franchetti et al., 2019).

Though the testbed was used for studying the effect of lighting, nutrition solution concentration, seed density, and day of harvest on microgreens production but it has potential to be used for conducting experiments with other plant species or treatment combinations.

1.3 Research Objectives and Approach

The aim of this research was two-fold: First, to develop an IoT indoor farm as an example of a real-world testbed for conducting experiments and addressing challenges concerning an IoT system. Second, build on the existing knowledge related to yield prediction and further it towards developing an intelligent farm. More specifically, the objectives of the study were to:

1. Design and implement an automated indoor farm production system by using off-the-shelf sensors and related components
2. Implement edge computing solutions and a cloud-based dashboard for real-time control and monitoring of the automated indoor farm
3. Build anomaly detection models to alert farmers and develop vision-based yield prediction models
4. Evaluate performance of the sensors and control system on variable application of nutrition, light intensity/duration, and seed density to microgreens production

1.4 Contributions

1.4.1 Design of automated indoor farm production system

- Designed an automated indoor farm production system that facilitated automated lighting, irrigation, nutrient application and monitoring environmental parameters (temperature, relative humidity (RH), Carbon Dioxide (CO2)) in the greenhouse. This system was the foundation for growing multiple batches of microgreens and source for acquiring data from the sensors.
- Developed a dashboard from scratch and hosted on Purdue server. It facilitated two-way communication between the user and the sensors/ actuators for implementing real-time control of the indoor farm.
- Installed a set of gantries for moving red, blue, green (RGB) sensors and thermal sensors over the growing trays to capture images of plant canopy. The gantry was remote controlled and a cartesian co-ordinate system was used to guide the motion.
- Designed calibration setups and procedures for light, flow, temperature, and RH sensors to determine the accuracy of measured data and make a choice about a particular make and model of the sensors.

1.4.2 Algorithms

- Benchmarked the performance of various anomaly detection algorithms to identify the best model for the data captured from temperature and RH sensors used in the indoor farm. The best model was utilized in realizing a framework for identification and storage of anomalies in the cloud.
- Developed a novel anomaly detection framework for fault localization in the sensor data capture pipeline.
- Designed an edge-fog-cloud operating pipeline for the system to reduce latency, ensure reliable real-time control by leveraging the computational power at edge nodes.

1.5 Thesis Organization

- Chapter 2 : Contains sections of a brief on IoT in agriculture and related work, A brief of AI-ML in applications in agriculture with focus on anomaly detection paradigms in sensor systems, related works, and yield prediction.
- Chapter 3 : Debriefs system architecture as various sub-systems with materials and methods employed in developing and deploying it, data preparation and choice of algorithms for anomaly detection and yield predictions.
- Chapter 4 : Results discussed as
    - Explanations of various design decisions and learnings from field
    - Experiments with edge computing and real-time control frameworks
    - Application of anomaly detection in system and results pertaining to various ML algorithms
    - Image dataset collected and analysis for yield prediction.
- Chapter 5 : Conclusion

2. LITERATURE REVIEW

IoT devices, robotics, AI are some of the key technology components of an automated indoor farm. This chapter provides synthesis of various research studies related to application of IoT devices in indoor farms and identified gaps under the section titled “IoT in Agriculture”. A summary of edge computing is also provided there. The discussion is further expanded by focusing on applications of “AI in Agriculture” in the next section where two applications, ‘anomaly detection’ and ‘yield prediction’ are discussed. Anomalies and their types, research trends in agriculture and common techniques used to handle them are presented. Under yield prediction, vision-based techniques for plant phenotyping and relevant work discussed.

2.1 IoT in Agriculture

2.1.1 Overview

Organizations and researchers have defined IoT systems in a multitude of forms based on applications and assets a specific proponent wants to emphasize. One such definition provided by OASIS (Open Protocols, 2014) describes IoT as: “System where the Internet is connected to the physical world via ubiquitous sensors.” A high-level functional model provided by the Alliance for Internet of Things Innovation (AIOTI) consists of 3 layers – Application, IoT, and Network. Many researchers and technology companies have built on it. Multiple architectures have been proposed from various standpoints like domain, service, function, etc. (Ray, 2018; Verdouw et al., 2019). In the field of agro-industrial and environmental applications, a review of 72 studies suggested an architecture by incorporating commonly used and relevant methodologies (Talavera et al., 2017). This IoT architecture shown in Figure 2.1. contains physical, communication, service and application layer and was adapted in the design of the testbed in this project.

IoT devices (e.g., wireless sensor networks, network-connected weather stations, cameras, and smartphones) can be used to collate a vast amount of environmental and crop performance data, ranging from time-series data from sensors to spatial data from cameras to human observations captured via mobile smartphone applications or other network-connected user interfaces. Analysis of this data in real-time from a remote location, off the farm, is possible

because of internet connectivity. By extending the sensing system to include actuators, necessary control tasks like irrigation, ventilation, misting etc. can be performed based on input(s) from a rule-based system, an AI algorithm, or a user action.

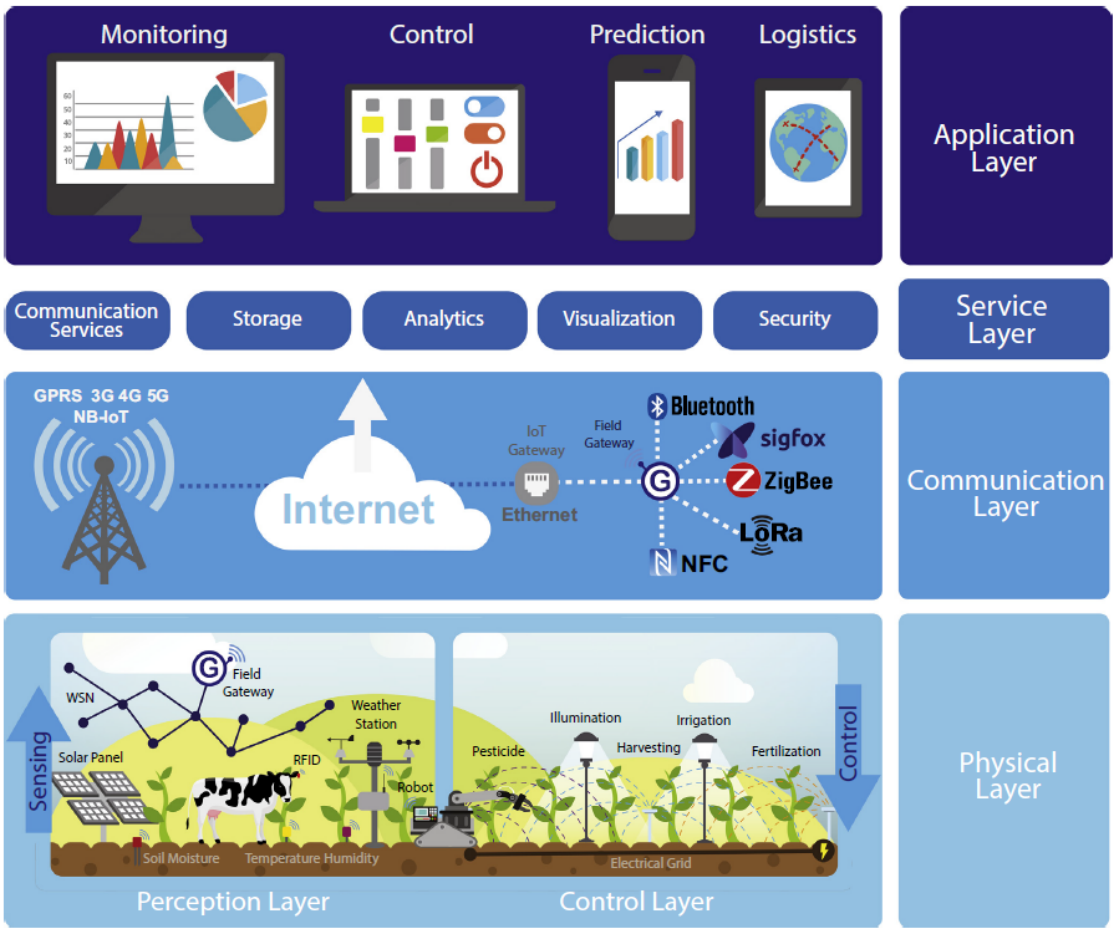


Figure 2.1 Basic IoT Architecture for agro-industrial and environmental applications (Talavera et al., 2017)

2.1.2 Context

Numerous benefits of IoT devices have led to their increased adoption and the domain of IoT has been enhanced by research activities persistently over the past few years. However, there are several unsolved issues about proper management and performance of IoT devices. Some of them are related to availability, reliability, scalability, mobility, data confidentiality, security, optimization, and compatibility of networks, etc. (Elijah et al., 2018; Khanna and Kaur, 2019; M et al., 2020). Several peer-reviewed studies also note that within an IoT environment, cost, constraint-free communication, proper deployment, and heterogeneity are additional challenges

that are to be addressed by further research (Elijah et al., 2018; Khanna and Kaur, 2019; Ummesalma et al., 2020).

A lot of literature exists about application protocols (Glaroudis et al., 2020) and connectivity methods (like Wi-Fi, Satellite, Bluetooth, cellular) (Ahmad et al., 2019; Vannieuwenborg et al., 2018). Data security and privacy issues are emerging as a domain that are primarily been of interest to computer scientists and financial institutions across the world. To deal with availability and cost of commonly available sensors/components, numerous solutions have been proposed (Danita et al., 2019; Ruengittinun et al., 2017). However, there is a noticeable void in the literature on how to improve and adapt IoT solutions for real-world indoor farms beyond simple prototypes. Direct scaling in most cases implies that the number of hardware components grows very quickly with the size of the system. Careful requirement analysis and system design are indispensable for a viable IoT solution in a real-world scenario. Along with it, a series of checks in the system over hardware and software are required to steer through the reliability challenges that are especially associated with the use of low-cost sensors.

Data management is yet another aspect of IoT devices that calls for attention. Sensors and devices can connect indirectly through the cloud, where data is centrally managed or utilize the concept of edge computing wherein data is sent directly to other local devices to collect, store, and analyze, and then share selected findings or information through the cloud. An Edge/Cloud combination offers a particularly attractive approach to deliver an efficient workflow. However, validation of edge-driven services on operational farms is limited (O'Grady et al., 2019). Along with reaping the benefits of the technology, exploration of the paradigm of edge-computing in an indoor farm is required to identify systemic challenges.

2.1.3 Related work

A brief synthesis of recent research papers related to automation in greenhouses/indoor farms is presented in Table 2.1 below.

Table 2.1. Review of literature related to IoT for automated greenhouses.

Reference	System	Contribution and advantages	Disadvantages	Scale
(Jaiswal et al., 2019)	IoT and machine learning based approach for fully automated greenhouse	Uses Thingspeak platform for development; live streaming, unlocking of greenhouse based on facial recognition/ fingerprint	No human in loop	Prototype - two plants in a plastic covered tray
(Long, 2019)	Agricultural internet of things system based on cloud computing and machine learning	Uses Zigbee for communication, AWS cloud, Wechat application, machine learning (ML) based prediction for control	No details about implementation are provided	-
(Hemming et al., 2019)	Remote control of greenhouse vegetable production with artificial intelligence — greenhouse climate, irrigation, and crop production	Grew cucumbers in a greenhouse without human intervention for 3 months -fully AI based control; comparison between different control, growth strategies and cost analysis	Experiment in an expensive industrial grade facility, yield measurements are performed manually	greenhouse
(Ruengittinun et al., 2017)	Applied internet of thing for smart hydroponic farming ecosystem (HFE)	Low-cost sensors are used and are listed explicitly	No results from sensors data or system performance	Prototype - with 12-unit hydroponic system
(Danita et al., 2019)	IoT-based automated greenhouse monitoring system	Automate irrigation and ventilation; greenhouse is divided into sections and one soil moisture sensor is placed in each section	Basic visualization: No human in loop. The proposed solution works for continuous media but not for hydroponics	greenhouse with soil grown crop
(Zamora-Izquierdo et al., 2019)	Smart farming IoT platform based on edge and cloud computing	A control strategy is implemented at the edge node (PC) that plugged into existing hardware controls of the greenhouse; cleaned, recirculated drainage water; Validation via growing two tomato cycles	Only irrigation is discussed, no details about user interface	greenhouse
(Chang et al., 2019)	Integrated monitoring platform of plant growth based on IoT edge computing in greenhouse	Moving Zigbee module for enabling connectivity in the greenhouse; Details of system implementation and uses image processing at edge	Only environmental controls are taken care of	greenhouse

2.1.4 Edge Computing

In the context of IoT, with increasing number of devices and data in a network, traditional cloud computing has limitations in terms of latency, bandwidth, round the clock accessibility and update frequency of devices, and data security (Shi et al., 2019). Increased energy consumption and scalability challenges at cloud servers have pushed the rapid growth of edge computing over the past 4 years (Shi et al., 2019). Despite a rapid growth, the concept of edge computing is still in its infancy (Varghese et al., 2016). A summary of motivation, challenges, and opportunities in edge-computing is shown in Figure 2.2 (Varghese et al., 2016). A survey of 46 state-of-the-art research papers on the use of edge computing for precision dairy, irrigation, fire detection, wildlife surveillance etc. highlights utilization of techniques for reducing latency and computational offloading (O'Grady et al., 2019). The survey points out that the systems reviewed were mostly prototypes and the application of edge computing was still in initial stages. In the current study, a requirement to perform real-time control led to performing computation at the edge.

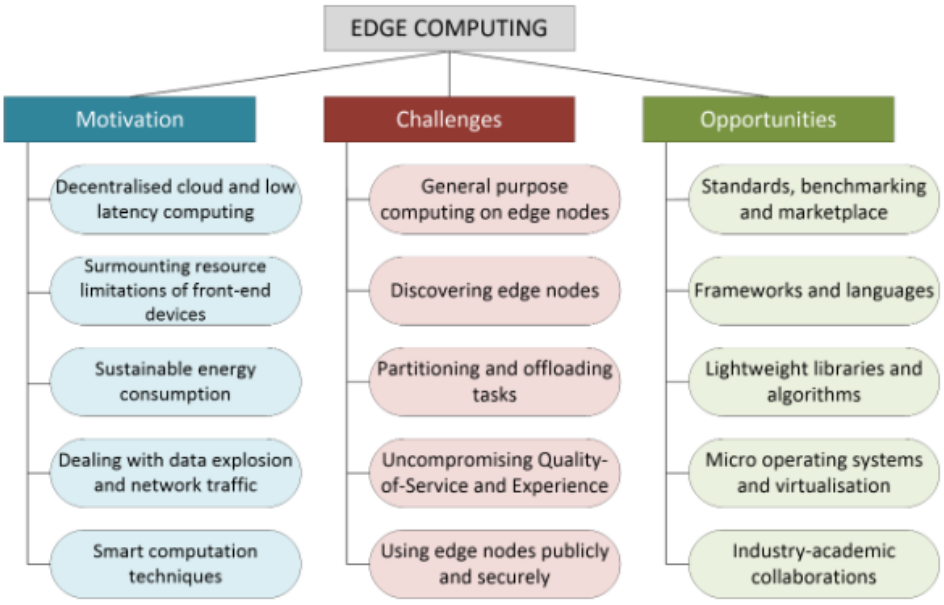


Figure 2.2. Motivation, challenges, and opportunities in edge computing (Varghese et al., 2016)

2.2 AI in Agriculture

2.2.1 Overview

Both indoor and row-crop farms produce a multitude of data points every day on temperature, humidity, soil moisture, water usage, etc. that can be leveraged and analyzed in real-time with the help of AI for obtaining useful insights (Liakos et al., 2018). The importance of such AI enabled applications increases with scale as it is infeasible for a human analyst to handle all the generated data manually. These applications can plug-in at multiple phases in the farming data pipeline and improve a wide range of tasks. By automatically processing the field data, AI-enabled systems can help improve the overall crop quality, yield, and resource utilization (Gertphol et al., 2018; Liakos et al., 2018). Weather forecasting can help crop planning, crop health monitoring can help take preventative measures, and farm robots can make tasks more efficient. A broad overview of the applications of AI in agriculture can be seen in Figure 2.3.

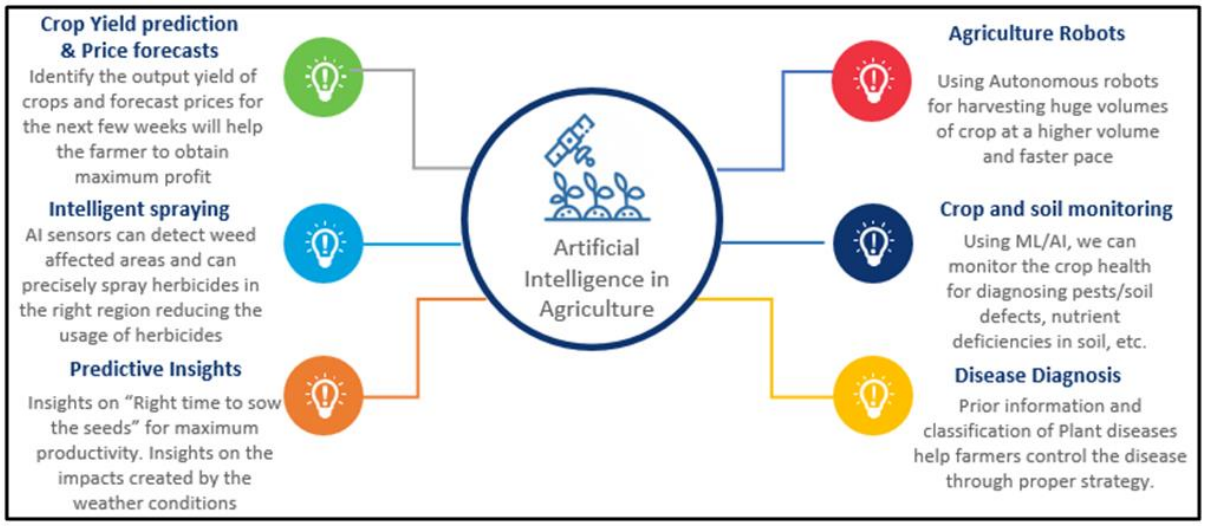


Figure 2.3. Broad picture of applications of AI in agriculture (Revanth, 2019)

In the context of increasing reliability of IoT systems and requiring clean data for predictive analytics, the current study is also focused on implementing anomaly detection algorithms for monitoring sensor data and for automated controls. In the context of moving towards closed loop and adaptive systems, AI for yield monitoring is also explored.

2.2.2 Anomaly Detection

Overview

As much as IoT paves a way for convenient data collection, the data needs to be accurate to make inferences and decisions about the farm. Abnormalities in the data captured by a sensor can occur because of multitude of reasons and their sources can be organized into two major categories (Ou et al., 2020)-

Sensor failures: Sensor misses or fails to capture data. This can directly cause an automated control system to make wrong decisions or indirectly be a source of bias in the data resulting in wrong inferences.

System failures: Sensor correctly captures the data of an anomalous event occurring in the system. This means, there is a fault occurrence in the system and corrective measures might be needed immediately and on a long-term basis to prevent crop/ equipment losses on farm.

Anomaly detection is the process of identification of such events in the system that deviate from the expected behavior. The relevance of anomaly detection is increasing by the day with deployment of increasing number of sensors on both outdoor and indoor farms and a general shift towards automation, thereby, making it extremely troublesome to manually check a huge amount of data acquired from the automated farms. Anomaly detection is based on the hypothesis that anomalies stand out trivially/ explicitly from rest of the data and can be quantitatively differentiated from the clean data. Different detection and handling techniques have been developed across diverse domains based on their nature and application. The types of anomalies, popular techniques used to handle them in different IoT applications and related applications in agriculture are discussed below. This section considers anomaly detection only in the context of an IoT-based system and exclude other management issues affecting crops in the farm such as presence of obstacles, weeds, diseases, etc.

Types of anomalies

Numerous classifications exist for differentiating anomalies in a dataset based on nature of input data and domain of application (such as agriculture, security, banking, etc.) (Foorthuis, 2020). In the context of time-series and spatial data, a broad classification approach that differentiates

between three diverse categories is described below along with a graphical representation of anomalies for a sample temperature sensor data (Chandola et al., 2009) (Figure 2.4):

Point anomalies (Global anomalies) – An individual data instance that is significantly different from rest of the dataset wherever it occurs.

Contextual anomalies (Conditional anomalies) – An individual data instance that is anomalous in a specific context but an identical instance in another context need not be considered so.

Collective anomalies – A collection of data instances together deviate from the rest of the dataset; the individual values need not be anomalies in themselves either globally or in the context.

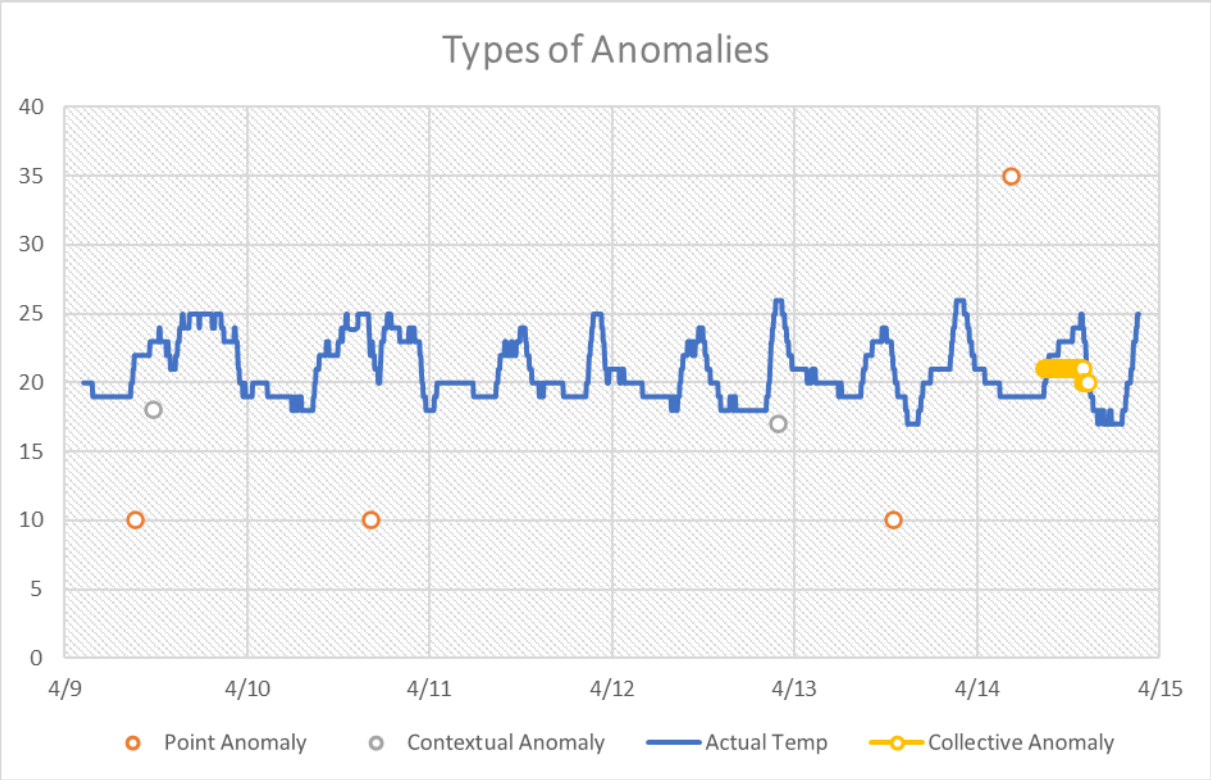


Figure 2.4. Types of anomalies shown on a sample of temperature data from indoor farm.

Common anomaly detection techniques in IoT applications

An extensive review of different techniques used in anomaly detection across domains is presented in (Chandola et al., 2009). In the context of IoT applications, discussion on broad categorization of anomalies/ outlier detection approaches are available (Ayadi et al., 2017; Gaddam et al., 2020) and shown in Figure 2.5. The pros and cons of each one of these approaches has also been systematically investigated. Based on the past studies, some of the important

questions to be posed before designing/developing an IoT-based anomaly detection algorithm are as follows–

1. Where is the computation performed – Is it centralized or distributed?
2. When is the computation performed – Is it in real-time or in post data analysis?
3. What are the resource constraints – What is the memory required, energy usage, network bandwidth etc.?
4. What is the nature of data being looked – Is it labeled or unlabeled? Is it univariate or multivariate? What is the type of data?
5. What is the computational complexity of the algorithm – Is it scalable?

Different algorithms are explored in the present study based on these considerations on the data generated from the farm and publicly available data.

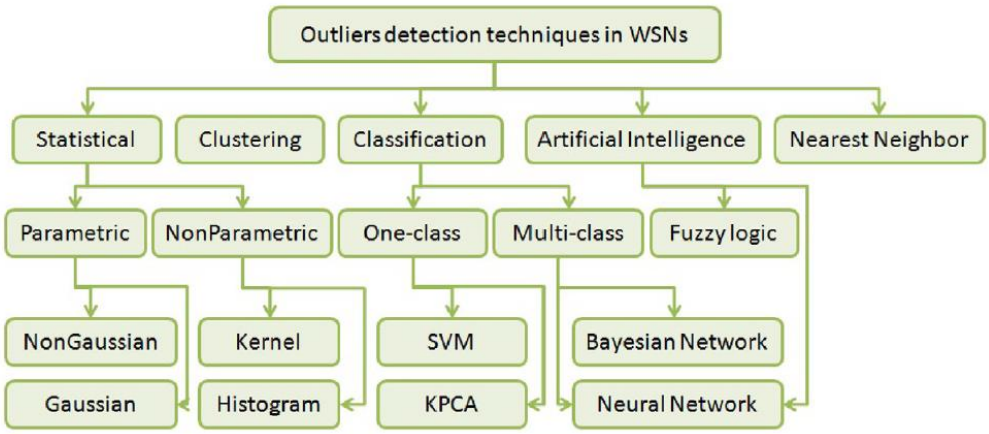


Figure 2.5. Outlier detection techniques in wireless sensor networks (Ayadi et al., 2017)

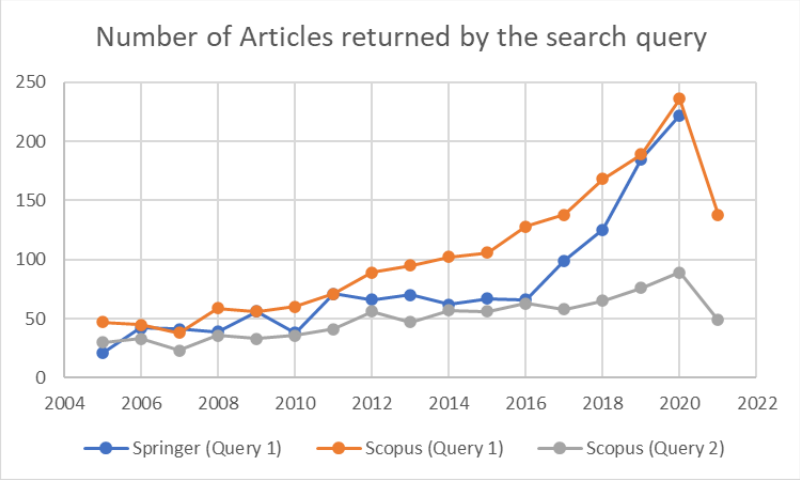
Related work

Although anomaly detection is a well explored field and a lot of systems inherently have some techniques in place to handle erroneous events, there is a gap in the documentation of models used in indoor farms. In fact, very few research papers explicitly mentioned anomaly detection techniques in smart farms or sensor networks in agriculture. A brief bibliometric analysis based on Scopus (title-abstract-keyword search) and Springer (conference paper search) databases between 2005-2021 for documents in English is provided in Figure 2.6. The trend is increasing with the search terms, but the subject area pie-diagram for the past 5 years (2015-2020) shows that only 9.8% of the studies come from agriculture and biological sciences, 16.5% belong to environmental

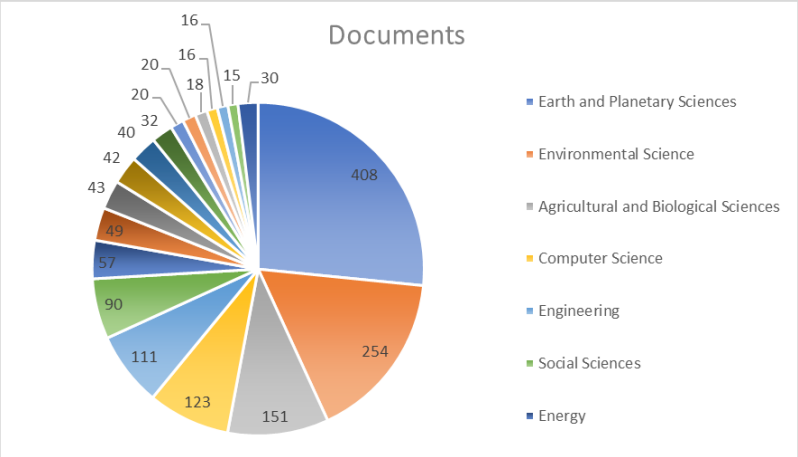
studies. A search was also carried out in Google Scholar, IEEE, Sensors Databases. Further filtering out relevant works based on quality criteria of whether the study presents a solution for the problem in a real farm/ greenhouse setting, shows some details of implementation etc. left about 22 papers between 2015-2021. A review of such papers found and filtered after extensive search in research databases for anomaly detection in agriculture is in Table 2.2.

Database Search Query 1 : '("Anomaly" OR "Outlier" OR "Sensor Failure" OR "Predictive maintenance") AND ("Agriculture" OR "Farming" OR "hydroponics" OR "greenhouse")'
Database Search Query 2 : '("Anomaly" OR "Outlier" OR "Sensor Failure" OR "Predictive maintenance") AND ("Indoor farming" OR "hydroponics" OR "greenhouse")'

(a) Search queries used for Abstract, Title, Keywords in Scopus database, Entire conference paper database in Springer



(b)Trends in search query by year



(c) Cumulative of documents by subject area from 2015-2020 returned by Query 1 in Scopus

Figure 2.6. Bibliometric analysis of anomaly detection algorithms in agriculture

Table 2.2. Review of research papers focusing on anomaly detection in Ag IoT systems

Reference	Issue addressed/ area of focus	Brief technique proposed/ used	Data used
(Yin et al., 2020)	Representations from deep learning (DL) networks require labeling training samples and the features extracted might not be suitable for anomaly detection in IoT	• Transformations on labeled data and augmentation • Convoluted neural network (CNN) for feature extraction • Classification using an anomaly scoring function	Public data : Images – CIFAR-10 and CIFAR-100; Net-flow – CTU Specific: 603 garlic images manually collected, 37 hours net-flow data from an online Ag IoT platform
(Sharma and Jain, 2018)	With no knowledge about data distribution, it is difficult to identify outliers in multivariate datasets	• Range specification by the user in the tool and keep/ delete the flagged point	Data in the application AGRETL – a hand coded ETL tool
(Liu et al., 2020)	Exploring application of ML based anomaly detection methods to vertical plant wall systems	• Different neural network (NN) models are tested • Autoencoder (AE) for point anomalies, long short-term memory (LSTM) for contextual anomalies proved best	Temperature (137, 592) and CO2 data (115, 431 data points) collected in the university
(L. Wang et al., 2017)	Traditional anomaly detection methods cannot effectively handle contextual anomalies	• A two-stage anomaly detection in Apache Spark • Stage 1 – Support vector machine to classify day/night. • Stage 2 – Gaussian mixture model (GMM) to find anomaly	45000 datapoints from Tomato greenhouse – Temp, RH, CO2, PAR
(Torres et al., 2017)	Increase the accurate data from low-cost sensors in IoT	• Data fusion from multiple sensors of same kind + use statistics-based outlier detection • Filters to smooth out any existing noise	~168 data points/ sensor. 12 soil moisture sensors in Cashew field
(Ou et al., 2020)	Real-time anomaly detection as opposed to reactive	• Prediction using regression + Quartiles for outlier detection	Soil Temp, Moisture, EC, PAR, Temp, RH
(Min and Hwang, 2021)	To forecast environment in a tomato farm	• Prediction of multiple variables for anomaly detection using LSTM	Internal Temp., external Temp., PAR, cumulative light volume, ventilation set Temp., heating set Temp., dew point Temp., RH

Table 2.2. continued

(Y.-B. Lin et al., 2019; Y.-W. Lin et al., 2020)	Sensor failure detection in farm	• Homogenous tests from multiple sensors of same kind based on calibration mappings • Heterogenous tests from multiple sensors at same node of different kind or sensor-actuator combination using adaptive thresholding	Soil Temp, Moisture, EC Barometric Pressure, Temp, CO2, RH, Ultraviolet light
(Karimanzira et al., 2021)	Case study to introduce Intelligent information management in IoT aquaponics system	• LSTM for anomaly detection • Bayesian network for fault localization from possible fault set	2 years of Water salinity, DO, Nitrates, PH, EC, Air Temp, RH, Light, Plant, Fish growth rate from images
(Abdallah et al., 2021)	Predictive maintenance problem in IoT sensors	• Tested Autoregressive integrated moving average (ARIMA), LSTM • Offline training for prediction and real-time classification based on 20% threshold from predicted value	Public Data: 5 WHIN Sensors - 58,019 datapoints for each of Temp, RH, Soil - EC, dielectric, temp, nitrate, water-nitrate
(de Souza et al., 2020)	Conceptual architecture for selection of ML based algorithms for detecting abnormal sensors	• Common unsupervised learning algorithms are tested in proposed architecture	Public Data: Forest cover – 286048 Various algorithms are compared for performance based on precision, recall, specificity, power consumption, execution time

2.2.3 Yield Prediction

Overview

Yield prediction is one of the most essential components of precision agriculture and spans the topics of yield mapping, yield estimation, matching of crop supply with demand, and crop management to increase productivity (Liakos et al., 2018). Traditionally, mathematical, or empirical crop models that use certain crop growth variables as inputs have been used to predict plant growth for research applications. However, those models required extensive agronomy and physiology experiments that limited their use for various crops.

With the rise of ML, more efforts have been directed towards the use of data-centric models that are based on correlation but do not necessarily deal with causation (Basso et al., 2013). Supervised learning models have been developed that use crop yield data from previous years to predict future yield (Liakos et al., 2018). Deep Learning models utilizing either satellite or UAV imagery (hyper-spectral, multi-spectral, RGB) for yield mapping are increasingly becoming popular. Imaging systems for various phenotyping applications are also gaining momentum where yield is one such variable proxied by stem diameter, leaf count, leaf area etc. (Li et al., 2020; Mochida et al., 2018). Nevertheless, vision-based techniques for indoor farms still have a lot of room for exploration to help understand the effects of lighting, nutrition, and other environment factors on plant growth and yield.

Data acquisition

Different types of imaging sensors have been used to collect multi-dimensional phenotypic data. Phenotyping is the quantitative assessment of structural and functional properties of plants. A basic digital RGB camera is adopted for color, texture-based applications. Hyperspectral camera, thermal infrared camera, near-infrared cameras has been used for providing complementary information to the RGB cameras to detect conditions like stress, drought etc. With advances in 3-D reconstruction and affordable options, RGB-depth (RGB-D) camera, light detection and ranging (LiDAR) devices are also being used for capturing entire plant level morphological changes (Mochida et al., 2018). In an open farm, satellite images are used to retrieve

information from field (Sishodia et al., 2020). In this application, a cheap RGB Camera and thermal camera are used to capture plant canopy.

Typical scenario in Computer Vision (CV) based phenotyping

Computer vision (CV) applications for phenotyping studies are gaining momentum (Li et al., 2020; Mochida et al., 2018). These studies are relevant for a crop like microgreens where stem length, leaf area and leaf count are directly related to crop yield. A typical scenario in a CV based plant phenotyping is shown in Figure 2.7. Schematic representation of a typical example scenario in computer vision-based plant phenotyping (Mochida et al., 2019)In the system proposed in current research, pre-processing of data particularly becomes important due to the quality of images captured owing to components used, distortions due to motion and detection, and image cropping required to extract target tray from a larger image. A single transfer learning model or other deep learning models can be utilized to perform segmentation, feature extraction, and finally predict the yield. Since it is formulated as a regression problem; hence the last step of classification would not be essential.

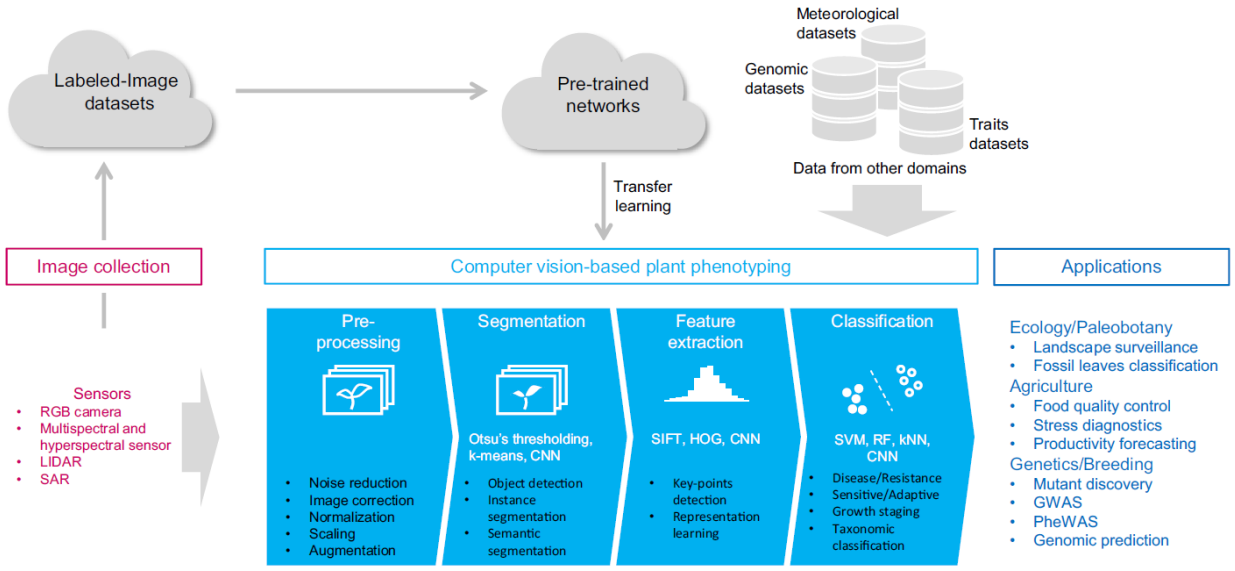


Figure 2.7. Schematic representation of a typical example scenario in computer vision-based plant phenotyping (Mochida et al., 2019)

Related work

A critical review of ML+ yield prediction algorithms, features and evaluation approaches used for crop yield prediction has recently been published (van Klompenburg et al., 2020). Out of the 50 papers that are critically reviewed, only 8 papers used images as features to predict yield while most relied on temperature (24), rainfall (17), soil information like soil type (17), soil maps (12), pH value (11) or crop information (13). The most used algorithms reported were CNNs, LSTMs, deep neural networks (DNNs). The critical review also revealed that models have been developed for limited data and that no specific conclusion can be drawn as to which model is better. This prompts for an independent study for the use case in this experiment with microgreens.

A summary of methods used for extracting leaf area/ count from images is presented in Table 2.3.

Table 2.3. Summary of literature extracting features like leaf count, leaf area from images

Reference	Title	Crop	Method	Image Background / Condition	Image Type
(K. Lin et al., 2013)	A Real Time Image Segmentation Approach for crop leaf	Multiple	Fuzzy C-means based on color quantization	greenhouse	RGB
(Franchetti et al., 2019)	Vision Based Modeling of Plants Phenotyping in Vertical Farming under Artificial Lighting	Basil	Mask R-CNN for 2D image segmentation which later is processed for 3D reconstruction	Vertical Farm	RGBD
(Chaudhary et al., 2012)	Fast and Accurate Method for Leaf Area Measurement	Multiple	Otsu's method	White background	RGB
(Z. Wang et al., 2018)	Image segmentation of overlapping leaves based on Chan–Vese model and Sobel operator	Cucumber	Chan–Vese model and Sobel operator	Field	RGB
(Praveen Kumar and Domnic, 2019)	Image based leaf segmentation and counting in rosette plants	Arabidopsis, Tobacco	Graph based method and Circular Hough Transform	Soil (Lab)	RGB
(Aich and Stavness, 2017)	Leaf Counting with Deep Convolutional and Deconvolutional Networks	Arabidopsis, Tobacco	SegNet	Soil (Lab)	RGB
(Ward et al., 2018)	Deep Leaf Segmentation Using Synthetic Data	Arabidopsis, Tobacco	Mask-RCNN	Soil (Lab)	RGB
(Nagano et al., 2019)	Leaf-Movement-Based Growth Prediction Model Using Optical Flow Analysis and Machine Learning in Plant Factory	Lettuce	Optical Flow Analysis	Vertical Farm	RGB
(Itzhaky et al., 2018)	Leaf Counting: Multiple Scale Regression and Detection Using Deep CNNs	Arabidopsis, Tobacco	FPN (Feature Pyramid Network)	Soil (Lab)	RGB

3. MATERIALS AND METHODS

3.1 Setup

Microgreens were grown in a 9x12 m walk-in glass greenhouse by blocking natural light to simulate an indoor farm with Light Emitting Diode (LED) fixtures used as the sole source of lighting. The testbed comprised of fifty-four experimental units distributed across three tables (Figure 3.1). Each experimental unit was broken into two layers: 1) a bottom 1020 tray with no drain holes to hold nutrient solution 2) a top layer of eight 12.7x12.7 cm seed tray inserts with Biostrate substrate on which microgreens are grown. LED fixtures were mounted at a height of 45cm above the top of the bench. All the circuitry, power supply, and required components were positioned underneath (or) adjacent to the corresponding tables.

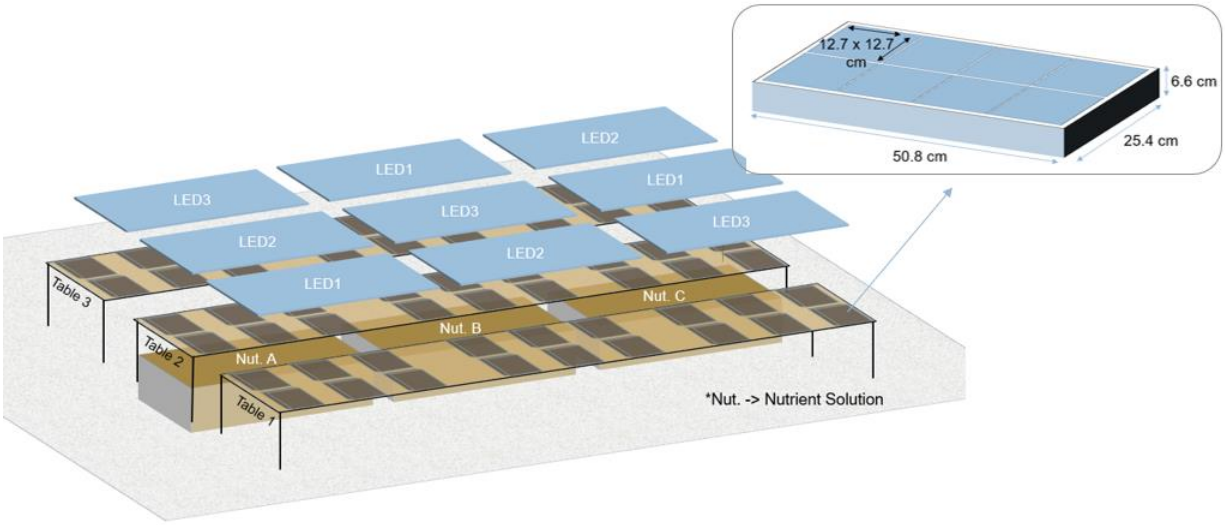


Figure 3.1. Conceptual layout of the experiment in greenhouse zone. Inset experimental unit which is an ebb and flow style tray system.

3.2 Statistical Design

The testbed was used to study factorial effect of lighting, nutrient solution concentration, seeding density and day of harvest on growth of microgreens (growth is indirectly inferred by weight or height measurements taken over 4 days of harvest). Each table consisted of three LED zones each of which accommodated 6 experimental units that were treated with three different

nutrient levels and two seeding densities. A split-plot design was used in this setting with a combination of LED spectra and intensity as whole plot factor, and nutrient solution concentration and seeding density as subplot factors. This design was randomized and replicated three times (Figure 3.2). A full-factor effects model (Equation 3.1) that considers all single factors, factor interactions and random effects of interaction of replicates and whole plot factor was used.

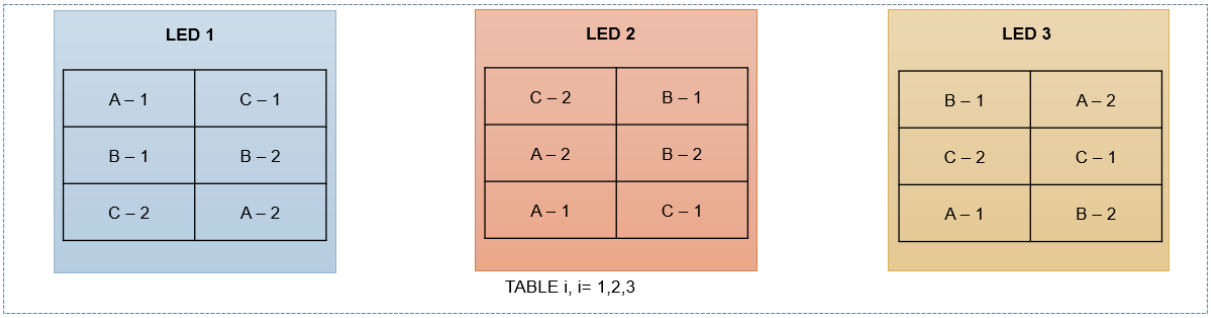


Figure 3.2. Statistical design of experiment used to evaluate factor effects on microgreen growth (A,B,C are 3 nutrient solution concentrations)

$$Weight \sim LED + Nutrient\ solution + Density + Day + LED * Nutrient\ solution + LED * Density + Nutrient\ solution * Density + Density * Day + LED * Day + Nutrient\ solution * Day + Nutrient\ solution * Density * Day + LED * Density * Day + LED * Density * Nutrient\ solution + LED * Density * Nutrient\ solution * Day + (1|Table * LED)$$

Equation 3.1. Full model for factor effects for the experimental design

3.3 Testbed Design Rationale

A modular design approach was followed to facilitate adding or removal of noes as per need and the points stated below were adopted for every functionality provided in the system.

- Use minimal number of components, thereby reducing both initial one-time setup and later maintenance costs as well as the electronics interfering with operations.
- Implement a series of checks for hardware and software used to ensure safety and reliability of the testbed.
- Remotely access and control the sensors and actuators to ensure regular monitoring and minimize human intervention for running the indoor farm.

3.4 System Architecture

The proposed system comprised of a network of sensors, actuators, Arduino microcontroller units (MCUs), Raspberry-Pis (RPis), IoT gateways, cloud database and user interfaces. An overview of hardware and software architectures is provided in Figure 3.3 and Figure 3.4. The IoT architecture contained physical, communication, service and application layers were adapted in the design of the testbed as explained below.

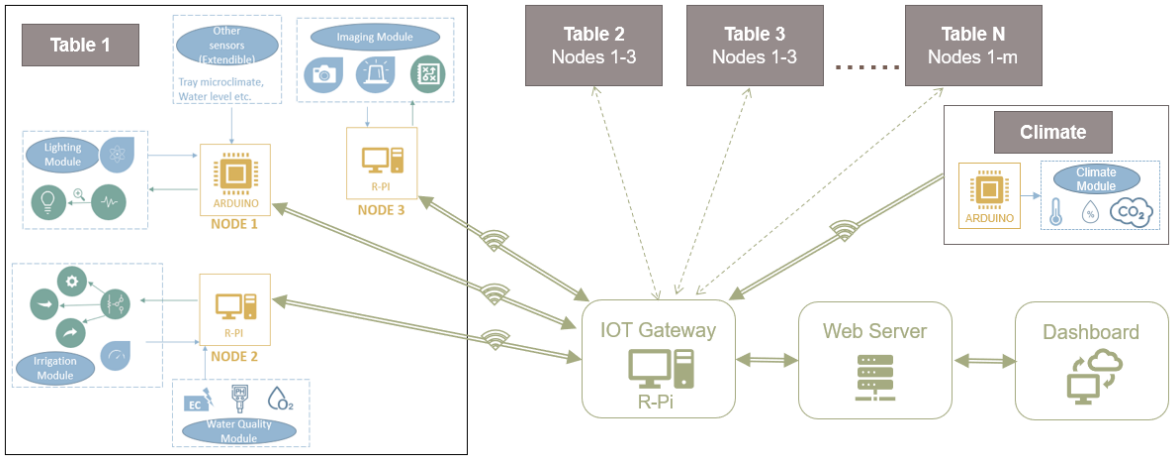


Figure 3.3. Hardware architecture showing network connectivity of various functional modules and nodes

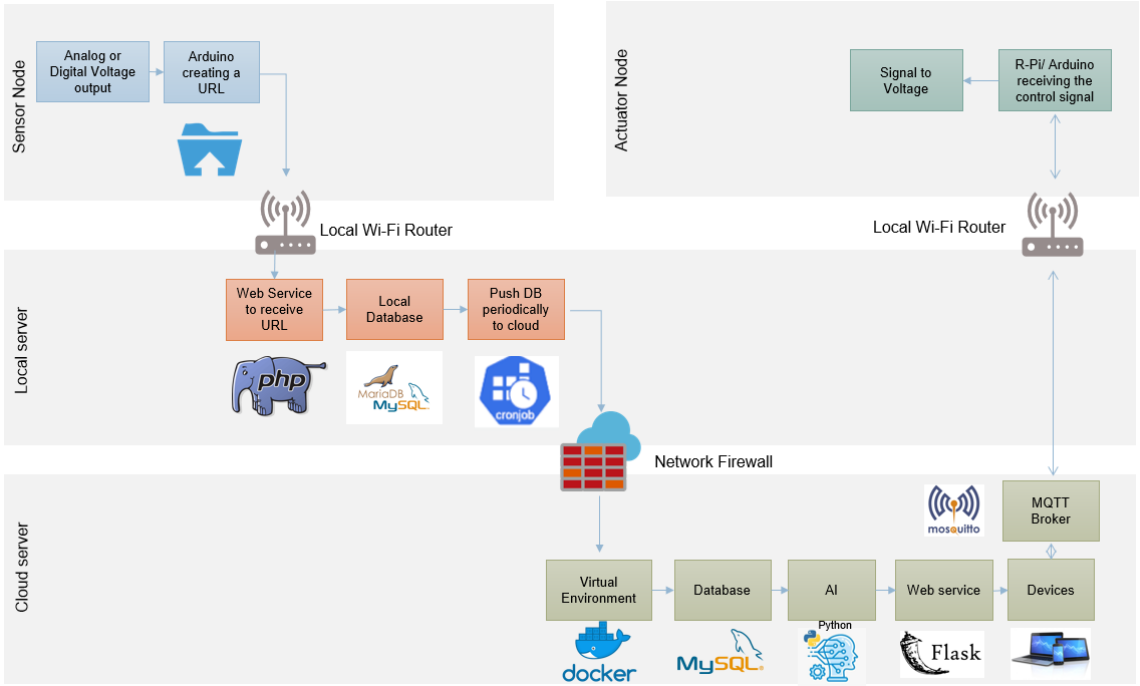


Figure 3.4. Software architecture showing typical flow of data, commands in the system

3.4.1 Physical Layer

The physical layer can be viewed as a collection of modules/ sub-systems that perform specific functions as guided by the MCUs (Arduino or RPi) (Figure 3.3). The MCU along with sensors and actuators attached to it is referred to as a node. The physical layer comprised of several nodes. Sensors collected data at specified intervals or whenever triggered by a user and then transmitted the data to the aggregator (gateway). Actuators (for example- LEDs, valves, pumps, and gantries) were controlled from client-side application/ edge node/ predefined set of instructions. A discussion on each of these is provided below.

MCUs

Ideally to facilitate a cloud based IoT system, each sensor should be equipped with its own network capability. Most of such wireless sensors or cloud-store type data loggers are extremely expensive, generate data held by proprietary systems making it difficult to integrate. Given the design intent of the system (i.e., low-cost), it was deemed more practical, cost effective to procure off-the-shelf sensors and add Wi-Fi capabilities using existing MCUs. Thus, Arduino Uno Wi-Fi Rev2 was chosen, which offered the benefits of having digital, analog I/O pins suitable to be interfaced with almost any sensor in the market and built in Wi-Fi.

In case of necessity of edge computing, a Raspberry Pi 3 B+ or 4 B was used in the node. Along with providing control via Input/Output (I/O) pins, serial ports, it enables data storage and processing on board. Given the requirement of a high number of relays in the current system design (40+ per table), the USB control was extremely handy in irrigation control.

Power supply

The phase voltage of the AC mains was stepped down from 120V to 5V using adapters (to power micro-controller) alongside which a 5V bus was maintained on the Veroboard (to power the sensors). Similarly, the relays and motors were powered by a 12 V supply. To feed the 5V data signal from the sensor to RPi general purpose input/outputs (GPIOs), level shifters to 3.3V were used. When choosing new devices, their operational ability at these three voltage levels i.e., 3.3V, 5V, and 12V DC were preferred, to avoid creation of more power rails and safety considerations. However, a variable adapter that can provide voltages in the range of 5-20 V powered the Pulse

Width Modulated (PWM) amplifier to control light intensities. Amperage was not a concern with sensors as they draw currents in the range of few milli Amperes and commercial adapters typically have higher ratings over 1 Ampere. On the other hand, it was an important criterion for driving motors to provide enough operating pressure in pumping or enough torque for pulling the gantry. Hence, a 12V 2A supply was used for driving pumps and 12V 3.5A was used for driving stepper motors. Official supplies were used for powering RPis.

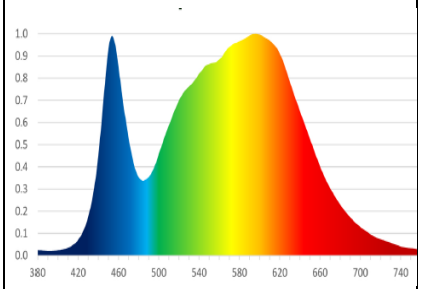
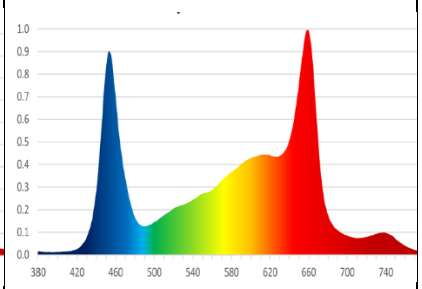
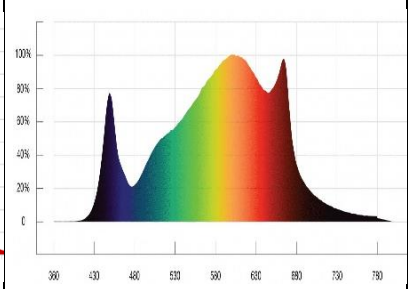
Lighting module

Three different full-spectrum white LEDs two of which were sheet type, and one bar type were studied for effects (Table 3.1). They operate in the photosynthetically active radiation (PAR) range which is light in 400-700 nanometer wavelength range. The photosynthetic photon flux density (PPFD) of these lighting fixtures varies linearly with voltage and was powered via an LED driver. A 0-10V PWM control signal was sourced from Arduino through a simple amplifier circuit to dim the light from 0-100%. The light intensity was constantly monitored by measuring the PPFD under it using a Quantum Sensor (SQ-225-SS, Apogee, Logan, Utah, USA). This sensor was calibrated specifically for use under electric light and correction factors were used according to the output spectra of LEDs to obtain the measurements within an error of 5%. The intensity at plant canopy depends on the mounting height and during the experiment an 18” height was maintained from the base of the bench.

PAR sensor calibration

For a PWM intensity control, a specific intensity value can be obtained by varying either the duty cycle or maximum voltage of the control signal. Also, the sensor output varies by the spectrum of the light, so a correction factor has been specified by the manufacturer. A mapping between intensity to duty cycle was needed to enable taking user inputs. Hence a comprehensive calibration table was created taking into consideration the variables of light spectrum, control voltage and duty cycle snippets of which are shown in Figure 3.5.

Table 3.1. Specifications of LEDs used in the experiment

Ref.	LED A	LED B	LED C
Image			
Spec-trum			
	Full Spectrum	Full Spectrum - red, blue emphasis	Full Spectrum
PPF	1014 µmol/sec	1281 µmol/sec	108-200 µmol/sec
Angle	Lambertian	Lambertian	120°
Power	620W	620W	85W * 3

Time	Light	Vmax	Duty Cycle	Raw PAR	Correction Factor	PAR Value	Duty Cycle %
5/14/2021 18:33	Patriot	10	0	5	1.13	22.09189	0%

For each of 5,6,7,8,9,10 V

0-255 in steps of 10

Obtained from Apogee correction tool

Raw PAR*0.8*4.88*Correction Factor - Multipliers obtained from datasheet

a) Snippet of data captured and processed in the calibration experiment

Light	Vmax	DutyCyclePercent	Average PAR
Bar	5	0%	15
Bar	5	4%	17
Bar	5	8%	18
Bar	5	12%	29

b) Snippet of final calibration table – When lights are at a height of 18” from table

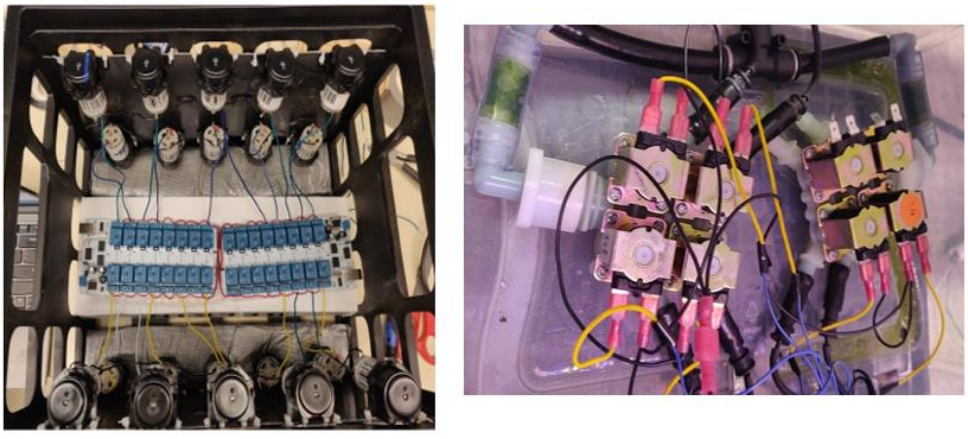
Figure 3.5. Snippets of data tables obtained from PAR Sensor calibration experiment

Irrigation module

A low voltage pump (12 V) was used to feed nutrient solution to the growth tray through a solenoid valve from a reservoir placed on the ground. A water flow meter/ sensor (FL-308T, Digiten, ShenZhen City, China) located on the line between the pump and valve was used to gauge the amount of solution flowing through. The output from this flow sensor was used to determine switching of the solenoid valves. A dedicated 12V motor was used to pump the water out of the tray. All the valves and pumps were interfaced with RPi via 16-Channel, 9-36V USB Relay Module (Sainsmart, Lenexa, Kansas, USA). The flow meter communicated with RPi via GPIOs.

Since there were 54 trays in the system, 54 such modules would be needed. But for the ease of management and minimizing costs, only one pump per treatment per bench was used, solenoid manifolds having multiple valves were chosen, and flow meter was positioned such that it could be used with all the valves in its line. For e.g., two 1-in 4-out normally closed DC12V solenoid valves (FPD-270A, Yanmis, Guangdong, China) were used for inflow control of a treatment that can feed up to 8 trays. The flow meter was between the pump and the valves and at any instance, only one of the valves was opened to control inflow of the corresponding tray. A typical setup for one bench is shown in Figure 3.6.

Three different nutrient solutions were used per cycle for each of the experiment to establish optimal combinations. EC ranges 1.2 – 2 milli Siemens(mS)/cm of the solution were tested with pH values 5.5-6.5 to ensure availability of nutrients to the plants. To prevent the escape of oxygen from reservoirs, an array of air stones with a timer were used.



(a) Central frame for holding irrigation motors, relays, and other circuit components

(b) Solenoid valve unit for one treatment

Figure 3.6. Typical setup of irrigation control placed under corresponding bench

Flow sensor calibration

Five different sensors with price varying from $7 - $270 were gravimetrically calibrated and contrasted. The number of pulses resulting in 1 liter of water was established based on the datasheet and iteratively validated in place, recalibrated for flow conditions of the system. The setup for calibrating and analyzing pulses, flows is shown in Figure 3.7. A liquid level sensor (PN-12110215TC-12, Milone Technologies, Sewell, New Jersey, USA) was used to assist in automatically capturing the volume of water, thus facilitating collection of more data points for future anomaly detection algorithms. A snapshot of the data captured from the experiment to arrive at a calibration table is shown in Figure 3.8

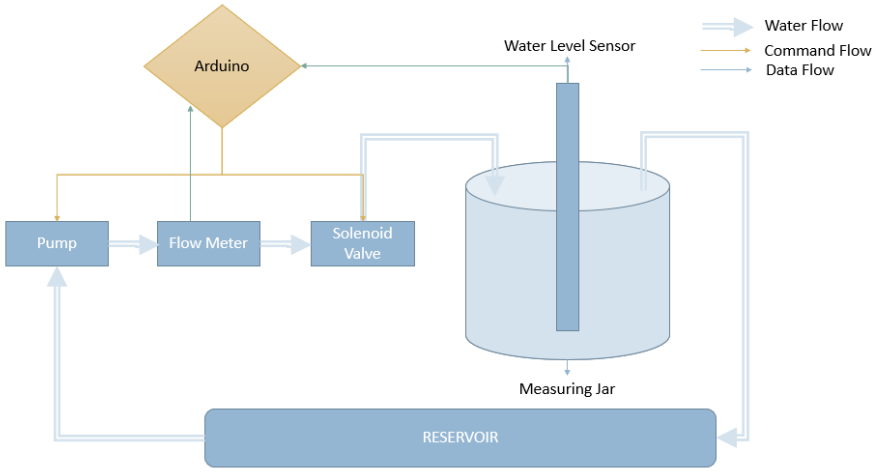


Figure 3.7. Flow meter calibration setup

Time	Sensor	Command Val	No. of pulses	Run Num	Sensor value	Vol of water (in ml)
1/26/2021 17:53	1AL7	Off		1	2104.56	0
1/26/2021 17:58	1AL7	On	6600	1	1877.79	1000

For each kind 1AL7, 1AL8, Digiten 3/8", Digiten ½" etc.

On/ off for corresponding number of pulses

Raw Sensor value

Mapping obtained from tape sensor calibration

a) Snippet of data captured and processed in the calibration experiment

Sensor	Table	Treatment	No. of pulses/ l
Digiten 3/8"	1	A	2100
Digiten 3/8"	1	B	1950

b) Snippet of final calibration table – When lights are at a height of 18" from table

Figure 3.8. Snippets of data tables obtained from flow meter calibration experiment

Water quality module

Nutrient usage of the plants was monitored using these sensors from Atlas Scientific (Long Island City, NY, USA) – EC probe (Conductivity Probe K 0.1) + circuit (EZO Conductivity circuit), pH probe (Lab Grade pH Probe) + circuit (EZO pH circuit), DO probe (Lab Grade Dissolved Oxygen Probe) + circuit (EZO Dissolved Oxygen circuit). These sensor circuits were interfaced via an isolator (tentacle shield for Arduino, Whitebox labs, Switzerland) with Arduino. The sensors communicate with the MCU using I2C protocol (I2C, 2003) and were triggered to obtain readings once the solution from a growth tray was emptied into the measuring jug. This module worked very much in tandem with the irrigation module upon trigger from the user dashboard or command line interface. The process flow is shown in Figure 3.9.

Imaging module

A system containing an RPi RGB camera (Raspberry Pi Camera V2.1), Thermal imaging camera (Grove MLX90640 110°, Seeed Studio, Shenzhen, China) for capturing the canopy imagery, and infrared proximity sensor (GP2Y0E02A, Sharp Electronics Corporation, Montvale, NJ, USA) to estimate plant height was setup to monitor plant growth. At the given standard height of LEDs mount in an indoor farm, multiple cameras would be required to capture the images from a single table. In the best case of using a single camera for a single tray, there would be a need for 18 camera systems and this setup would block the light on the tray underneath throughout the cycle. Moreover, since the frequency of capturing images is low, once in 15 mins, the utilization of the cameras in a fixed setup was also infrequent. Hence, the camera system was mounted on a robotic gantry which allowed a highly customizable and scalable imaging.

The gantry was assembled in-house using components sourced from a local hardware store to operate over the tables and the assembly is shown in Figure 3.10. A setup like this could be useful and budget friendly to indoor farm operators where one such gantry could be used to cover an entire connected level effectively. Stepper motors (NEMA-17, Quimat) controlled by Arduino drive the motion of the module. The current setup had two parallel gantry rails moving over the tracks using a single drive unit.

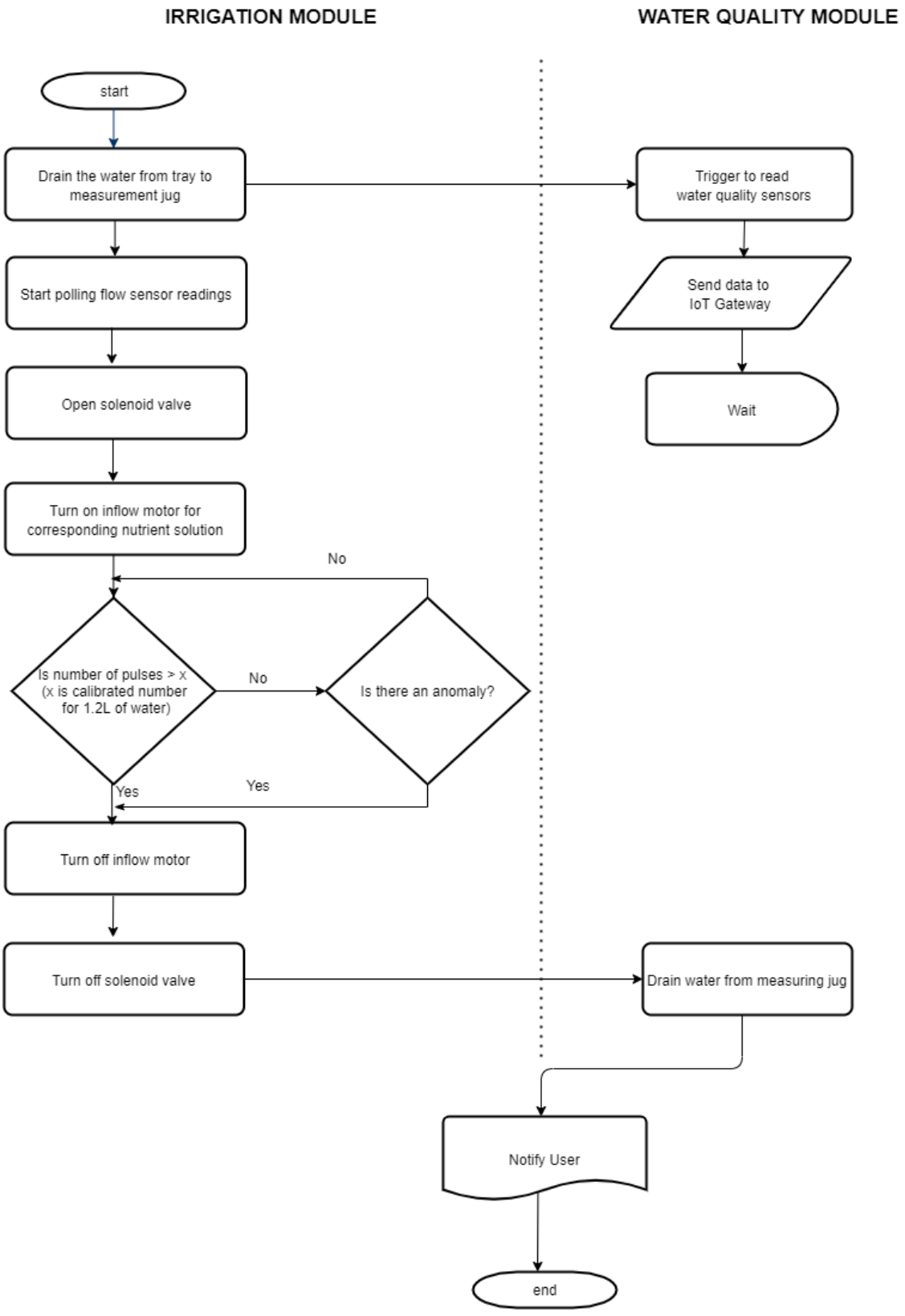


Figure 3.9. Process flow chart of irrigation and water quality measurement modules

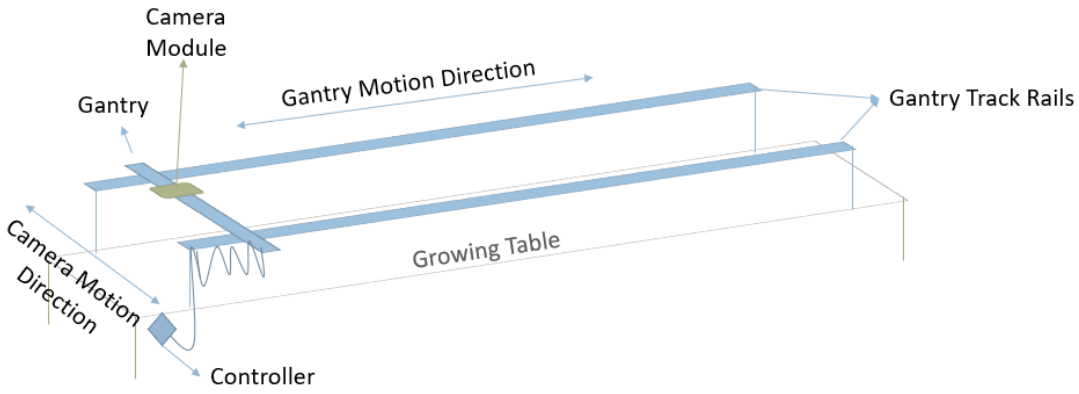
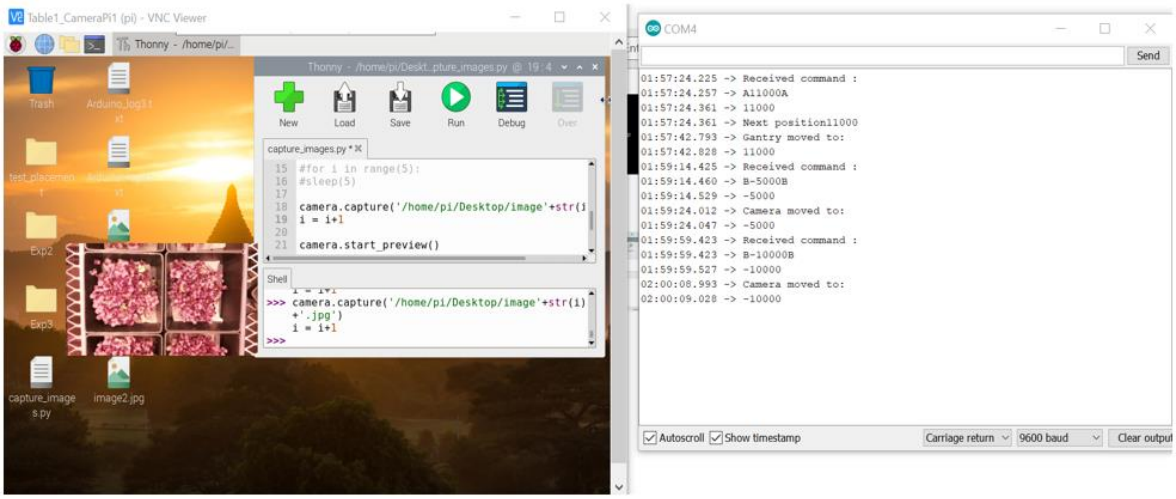


Figure 3.10. Robotic Gantry that carries imaging module over plant canopy

Rig calibration

Although the required number of rotations for stepper motors to go to a destination could be obtained by theoretical calculation, validation and adjustment was necessary because of the load, varying belt tension with length and camera position within a mount. The steppers of gantry and camera were moved to desired positions via Arduino serial interface and corresponding values were noted down when a satisfactory picture frame was observed in the camera preview in VNC viewer (RealVNC Ltd, UK) window. A snapshot of this is presented in Figure 3.11.



a) Snippet of iteratively moving the gantry to desired position using Arduino serial interface

GantryNo	GantryRailPositionNo	GantryCamPositionNo	X_GantryCamPos	Y_GantryRailPos	TrayNo	Subtray No 1	Subtray No 2
2	8	0	-38000	11000	1	2	1

b) Snippet of the data table with final mapping between positions and trays

Figure 3.11. Snapshot of imaging system calibration process

Climate module and other similar environment sensors

The general climate of the indoor farm was monitored from a central location. Temperature (±1.0°C accuracy), Humidity (±3% accuracy) were taken from the measurements of a four-in-one sensor BME680 (Bosch Sensortec, Reutlingen, Germany) sending data to the Arduino MCU via I2C. For obtaining CO2 concentration from the environment, Gravity v1.1 sensor (SEN0219, DFRobot, China) was used, and it measures concentration in the range of 0-5000 parts per million (ppm) with an accuracy of ± (50ppm 3% reading).

During the experiment, to be able to test new sensors/ add more functionality to the system, the additional I/O pins on Arduino of Node 1 were utilized. Currently, to note the microclimate conditions at tray level, waterproof sensor DS18B20 (Maxim Integrated, San Jose, California) having ±1.0°C accuracy was immersed to the bottom in three trays, one under each LED of a table. To monitor at the top of the canopy, DHT11 (Adafruit, NYC, USA) temperature (±1.0°C accuracy) and humidity sensor (1% accuracy) was used.

Temp., RH sensor calibration

All the environmental sensors used were connected to Arduino. The temperature, RH, and atmospheric pressure data readings were recorded for seven days by keeping the sensors side-by-side in a central location in the indoor farm as shown in Figure 3.12. A comparison was also made between an expensive research grade sensor, ATMOS 14 (Meter Environment, WA, USA).

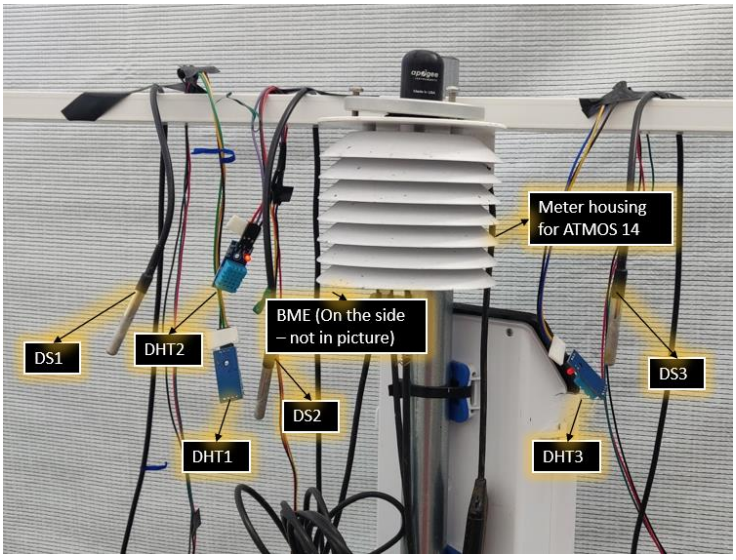


Figure 3.12. Set of environment sensors placed side by side for calibration

3.4.2 Communication Layer

Communication layer included both physical networks and protocols used in the ecosystem. All the devices in the indoor farm were connected over Wi-Fi network and located behind a secure firewall. Wi-Fi was chosen as transmission mode because of the reliable internet connections in indoor farms, given their proximity to urban areas, requirement to establish a two-way communication with remote devices and ease of availability of MCUs with inbuilt Wi-Fi functionality. A hybrid topology was used. A local server (running on RPi) was used as an IoT gateway and to receive data from all the nodes in the indoor farm via message queuing telemetry transport (MQTT), hypertext transfer protocol (HTTP) protocols. This data was periodically synced with a cloud database (DB) via secure shell protocol (SSH) due to the restrictions on greenhouse Wi-Fi network. Since the devices were connected to the internet in themselves, the edge computing nodes could communicate with cloud directly. But they were allowed to do so only in limited capacity to receive pre-defined control commands from dashboard and send corresponding progress back to minimize traffic at server in terms of number of devices connecting to cloud. All the devices thus ran MQTT clients. Additionally, all the nodes and IoT gateway were enabled remote access via VNC viewer within the network in addition to secure shell access for effortless application development/ deployment at node.

3.4.3 Service Layer

Cloud-based MySQL database synchronized periodically with local MySQL DB and facilitated multiple reads, write, and no overwrite functionality. This data was sent to dashboard for visualization or pulled via python for further analysis from backend. A secure MQTT broker was also hosted on cloud and relayed communications between all the clients.

Data Management

Tables for storing data from each node, functionality were created in the local and cloud servers. For e.g., WaterSensors_1 is a database table that stored data sent from water quality node from experiment table 1. PHP scripts received data annotated with client id via HTTP and each entry was added with a timestamp at the local server and later synced with cloud Database. A PhpMyAdmin was installed on this local server for quick administration in the farm.

Sensor sent data periodically to a local DB. However, actuators sent a few structured logs during operation directly to cloud DB for access from dashboard. Other than that, most of the actuator related data were stored as detailed log texts on the edge node that were processed in-situ for anomaly detection or high-quality storage and could be retrieved later from command line interface (CLI). For images, an MQTT client and auxiliary scripts dedicated to receiving them from the remote farm and sending them to relevant folders for dashboard access or storage ran on the server. Folders were organized by run number/ duration, with name of the file containing the sub-tray number and timestamp.

The intervals at which data was received varied. Sensor data was usually received once every 5-15 min based on the sensor, actuator data was on-the-go basis based or when the action was performed, and image data was retrieved only as per user's request. A summary of the database structure at the end of the experiment is shown in Figure 3.13.

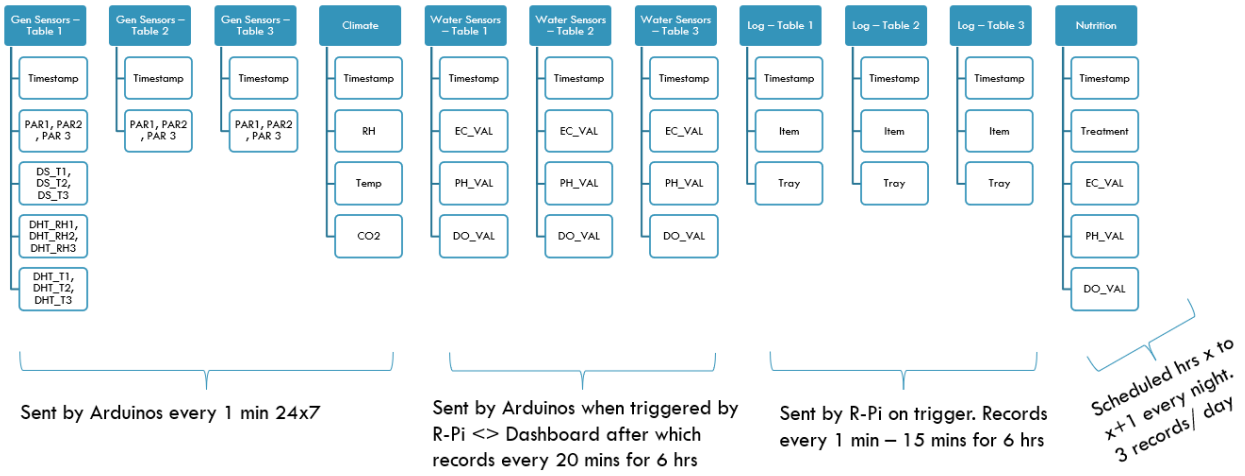


Figure 3.13. Database structure for storing sensor data with expected frequency of update

3.4.4 Application Layer

A client-side web application was developed to visualize sensor data and control actuators remotely. Dash python framework was used for building powerful visualizations with the data and enable quick deployment. Connection to cloud database was achieved using PyMySQL whereas climate, treatment monitoring, camera data was continuously retrieved by frequent callbacks facilitated by Dash. Multiple MQTT clients ran at the backend of the application to provide control functions in real-time. A sample software pipeline for triggering an irrigation cycle is shown in

Figure 3.14. Since the tasks were not instantaneous to keep the user in loop, the progress/ status of the actuator function was streamed till the task was completed.

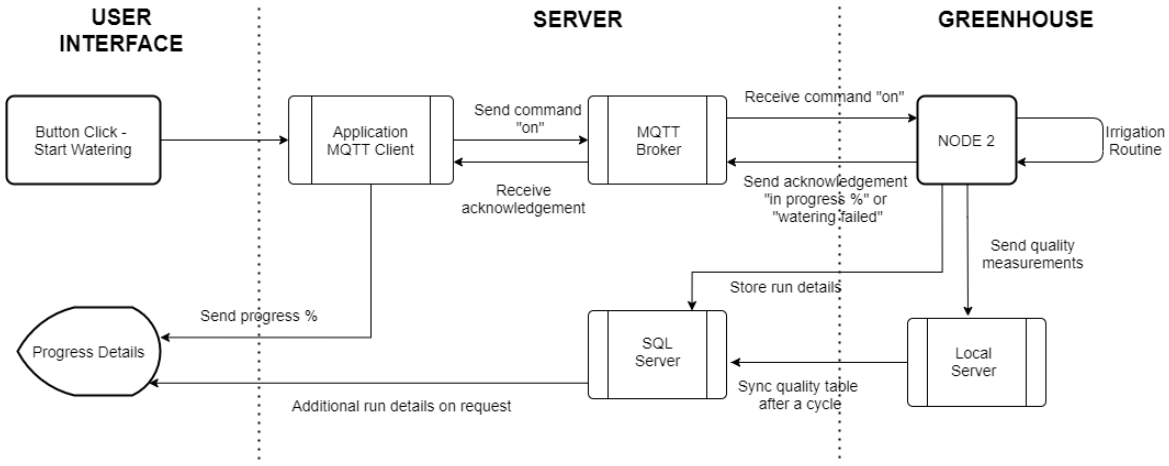


Figure 3.14. Example of a control operation pipeline (irrigation)

3.5 Edge Computing

3.5.1 Use cases

The value proposition of edge/ fog computing lies in leveraging the processing power of multiple local devices to enable enough quality of service (QoS) for some computationally intensive tasks as well as reduce latency. The testbed was designed under such paradigm and edge computing was used in the following manner:

- Reducing latency - When the user triggered irrigation of a specific tray, the flow sensor values were polled at the RPi upon receiving the user signal, sensor data was locally assessed for anomalies for action and periodic progress notifications were sent to the user as opposed to sending every pulse.
- Computation offloading - Images were processed locally on the RPi edge and a summary of the information obtained from them was sent over to the cloud.
- Reducing data traffic - Since the edge has in-built data storage capability, instead of sending data as soon as it was captured, the data points were bundled at the local server and synchronized with the cloud every 15 mins. The data that was captured from continuous reading of sensors like flow meters was stored locally at the node and only sent over in case of anomaly or upon request by the user.

3.5.2 Experiments

- To establish the necessity of edge computing for latency sensitive applications, one way latency tests were conducted for different QoS levels over 3 days each. An MQTT Client (Partner A) ran on RPi node in the greenhouse while another client (Partner B) ran on the cloud server. Latency was measured at the receiving client as the time elapsed between message origin at sending client to arrival at the receiving client. The measurements of average latency and observations from the experiment are provided in results section.
- To strategize the reduction of traffic at the local network or cloud, an analysis of a typical operation at the local server was performed under two scenarios – A) Data is uploaded from local DB into cloud every time a sensor sends data, B) Periodic update every 15 mins

3.6 Anomaly Detection

3.6.1 Use cases

A mapping of use cases was created between the types of anomalies that can arise from different sensors on the farm and techniques to handle them and is shown in Table 3.2. Numerous experiments were conducted on data from climate sensors and different techniques were explored. The data preparation and details of the process are discussed in the following sections.

3.6.2 Exploratory Data Analysis and Pre-processing

For time-series based algorithms, it is essential that the data is in continuous blocks. However, the system had significant chunks of missing data owing to issues faced because of MQTT, internet protocol (IP) address changes and Wi-Fi. The longest possible interval of data with no issues or least issues of discontinuity is considered as training set and the rest of them are used for different test sets, validation set. In case of a discontinuity, if the data was missing for less than 1 hour, it was filled with previous known value automatically by the code. If the data was missing between 1 to 3 hours, it was filled with values from a similar interval on the day before or later after manual verification. If more than 3 hours were missing, the day was deleted.

Data cleaning required identification of normal and abnormal points. Empirically, anomalies rarely appear in the dataset, but time series plots, box plots were drawn to observe any potential

ones. Literature also suggested that climate parameters follow binormal distribution, so density plots were drawn to verify this and further obtain thresholds for eliminating extreme points. Z-scores were used to detect outliers and points beyond threshold $\mu - 3\sigma$ and $\mu + 3\sigma$ are removed iteratively until there is a clean dataset. A manual inspection was also done to remove any further deviant points.

Table 3.2. Different types of anomalies, their use cases, and techniques to handle them

Anomaly Type	Use cases	Techniques to handle
Point anomalies	• Occasional spurious values by sensors because of interference or noise or software error • Missing data chunks which might point to network connectivity issues at the node	• Standard quantile-based outlier detection methods, prediction-based methods like regression, NNs • Threshold on data receiving intervals at the server
Contextual anomalies	• Deviation from expected correlation between multiple climate sensors in use at the same time • Deviation from expected correlation between water quality sensors of similar treatments • PAR sensor values not reflecting the 18h photoperiod condition which might mean that LED is not working or software error with the controlling code.	• Pattern detection methods like auto-encoder, LSTM • Validation with expected patterns from previous data
Collective anomalies	• Gradual failing of an actuator, for e. g. reduced flow rate in water sensors indicating a blockage at the valve, pump failing or power supply issues. • Water flow sensor failure, if designated number of pulses exhibit deviant pattern from data captured at calibration. • Height of the microgreens in a tray not increasing over days calling for reactive operation	• Time series-based methods • Pattern detection methods like auto-encoder, LSTM

3.6.3 Anomalous Data Generation

For inducing anomalies into the clean training datasets, a contamination % of the training data was chosen and point, contextual, collective anomalies were imputed in a ratio of ~3:2:5. A lower overall number for contextual anomalies was chosen given the non-triviality of imputing these anomalies and the data size we have. Collective anomalies need to be in groups, so a higher fraction was allocated for them. Rest of the fraction were made point anomalies.

For point anomalies, random indices were chosen from the dataset and the corresponding points were replaced by anomalous values that were generated from a uniform distribution between

[theoretical minimum value, $\mu - 3\sigma$] and [$\mu + 3\sigma$, theoretical maximum value]. For contextual anomalies, random windows corresponding to 12 h length were chosen and either the maximum value was replaced by minimum, or the minimum was replaced by maximum value. For collective anomalies, several consecutive data points with variant length values between 3 to 6 were selected and replaced with values that were 10 to 15 hrs away from the selected points.

3.6.4 Modeling pipeline for the experiments

For ease of reference, the original dataset was referred to as D_{raw}, the clean dataset with outliers removed was referred to as D_{clean} and the dataset imputed with anomalies was denoted as D_{ano}. A subscript of train/ test data like $D_{clean,train}$ or $D_{clean,test}$ was used to depict the training and testing sets as used. The property of interest was shown in the brackets following the dataset like $D_{clean,train}(Temp)$.

Unsupervised models were trained to output two classes (anomaly/ not) from the data fed into them. Unlabeled $D_{ano,train}$ was used for training and $D_{ano,test}$ for testing. Supervised learning models as classification problems were not implemented due to the known pitfalls with imbalanced datasets and their lesser popularity for anomaly detection. Also, the dataset we have becomes very small to explore these in attempts to balance it. For supervised prediction-based algorithms $D_{clean,train}$ was used for training and $D_{clean,test}$ for predicting the future values. $D_{ano,test}$ was compared with this set and based on a quartile or fixed threshold a point is flagged as anomaly/not. The working pipeline of data processing, modeling is shown in Figure 3.15.

3.6.5 Models used in experiments

Numerous unsupervised learning algorithms that were discussed in literature were explored. A general summary, with the strategy to extract the best performance from a particular model and comments are mentioned in Table 3.3. Numbers supporting the comments follow in the results section. In prediction-based models, to start from the simplest, linear regression was considered. Taking advantage of the Scikit-learn (Pedregosa et al., 2011) linear model library different regression methods were evaluated. ARIMA model was chosen initially, it performed well on the training data but generating a rolling forecast for the testing data took long hours (at least 6 hrs for 2000 points) and parameter tuning was more time consuming. Hence ARIMA model

was not chosen for further consideration. A tree-based model Gradient boosting regressor (GBR) was also tested out. Moving to neural networks, a simple artificial neural network (ANN) with two dense layers was evaluated. Similarly, LSTM and Bi-LSTM with one LSTM/ Bi-LSTM layer followed by one dense layer were evaluated. All the neural networks were trained to 10 epochs, although in most cases they converged within 2-3 epochs. A point is flagged as anomaly if the relative error between predicted value and actual value is greater than threshold. The threshold was chosen from receiver operator characteristic (ROC) curve or precision-recall (PRC) curve.

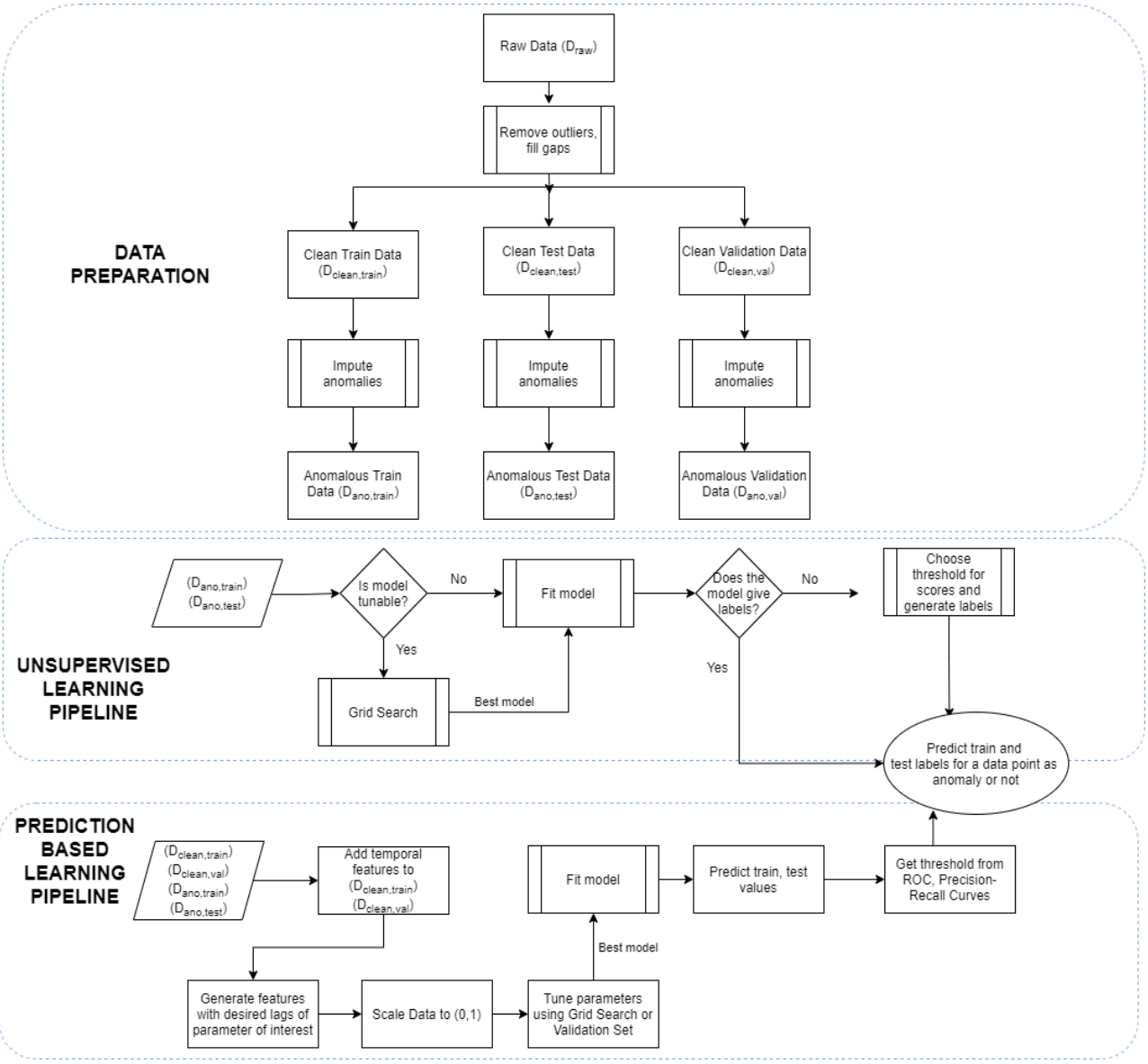


Figure 3.15. Data preparation and modeling pipeline for experimenting with different anomaly detection algorithms

Table 3.3. Summary of hyperparameter strategy and general performance of unsupervised models for anomaly detection

Model	Nature	Hyperparameter strategy	Comments
Histogram-based outlier score (HBOS)	Statistical, Non-Parametric, Proximity-based	Fixed no. bins = $\sqrt{No.\ of\ training\ samples}$	Proximity based methods usually only work well with global outliers
Kernel density estimation (KDE)	Statistical, Non-Parametric, Proximity-based	GridSearch over different kernels and bandwidths. Cutoff quantile for threshold = 0.01	
K-nearest neighbors (KNN)	Clustering, Proximity-based	Iteratively choose num_neighbors. Cutoff distance quantile = 0.01	
Local outlier factor (LOF)	Clustering, Proximity-based	GridSearch over num_neighbors	The model is either sensitive to only one of local or global outliers, but not both at any time
Elliptic envelope (EE)	Probabilistic	Contamination = 0.01	Expects underlying data to be Gaussian and good for global outliers
Cumulative sum control chart (CUSUM)	Probabilistic	Choose h, k iteratively	Sensitive to small changes but fails to detect larger ones. So does not work well when the base data is erratic
GMM	Probabilistic	Num. clusters set to 2 assuming normal points & outliers fall under different Gaussians	If underlying distribution is multi-Gaussian like RH, it fails
Isolation forest (iForest)	Ensemble, Tree-based	GridSearch over different max tree size	Does not work for contextual anomaly

3.7 Yield Prediction

3.7.1 Problem Formulation

The goal was to predict yield based on canopy images captured at certain time intervals pre- harvest. This was formulated as a regression problem for deep learning model where an RGB image collected ‘t’ time ago from the harvest was tagged as ‘x’ variable with the weight at harvest as ‘y’ variable. Height was also considered one of the ‘x’ variables when available in one experiment.

3.7.2 Data Preparation

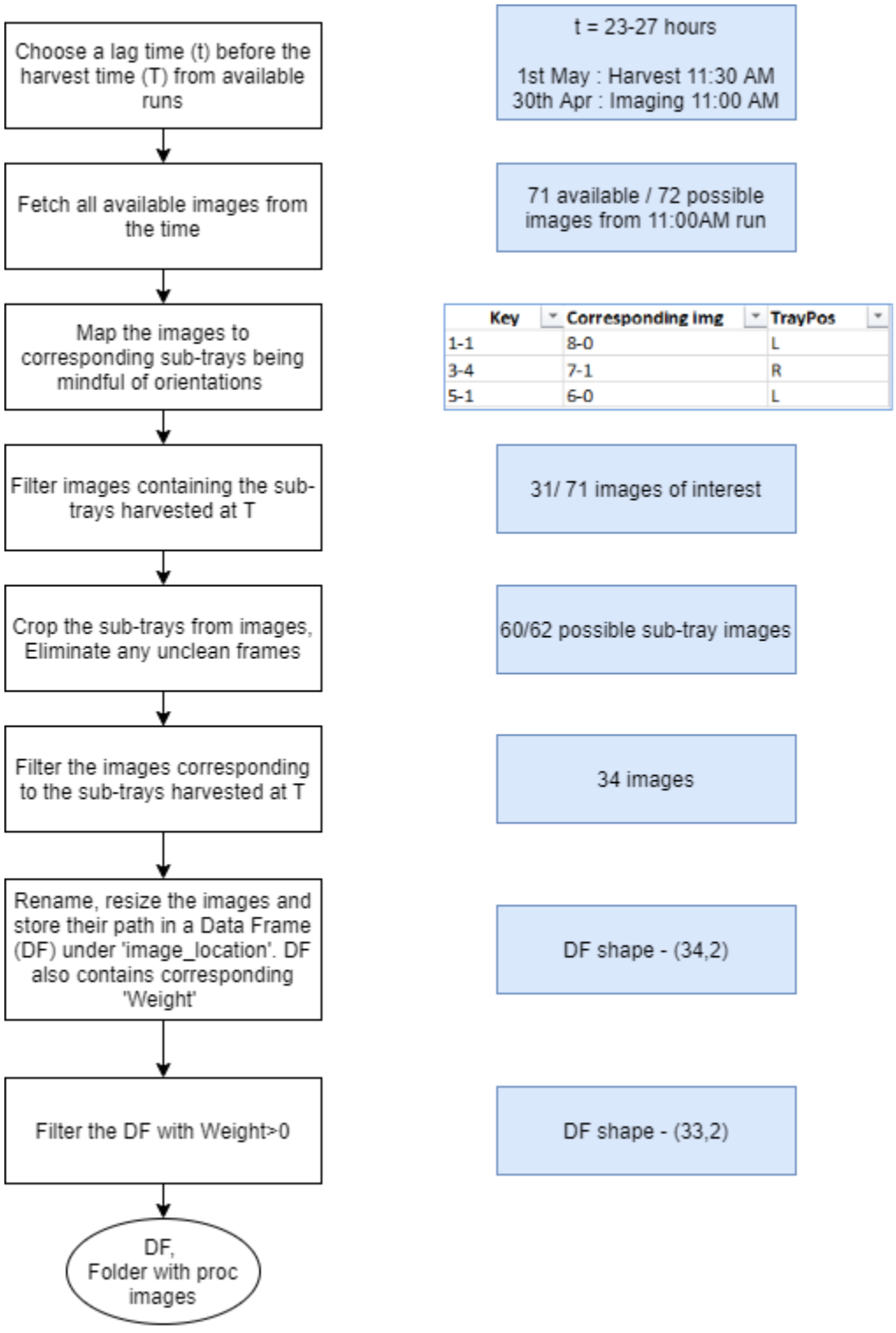


Figure 3.16. Flow chart showing data frame preparation from images captured by the gantry for feeding into deep learning models

Images acquired of microgreen canopy from gantry system were processed as shown in Figure 3.16 to feed the deep learning model. Trays that do not have any growth (Weight 0) were removed as it was raising issues in calculating the error metrics mean absolute percent error (MAPE) along with not having any significance. The images when acquired were 1024x768 pixels which were cropped roughly into half, i.e., brought down to 512x768 pixels per sub-tray which in turn were resized to 224x224 pixels for being fed to the deep learning architectures.

3.7.3 Code flow

Models were built in python using Keras and TensorFlow backend on Google Colab. A typical workflow of the code is shown in Figure 3.17. All fitting and callbacks were written in a generic function which can take a specific model as input and keep rest of the setup same for comparison. Each model gets its own separate function and experiments were performed with a custom CNN (referred to as SmallCNN further), ResNet50 (He et al., 2015) with no top layer, custom layers added (referred to as ResNet50 further) and Efficient Net B0 (Tan and Le, 2019) with no top layer, custom layers added for regression (referred to as EffNetB0).

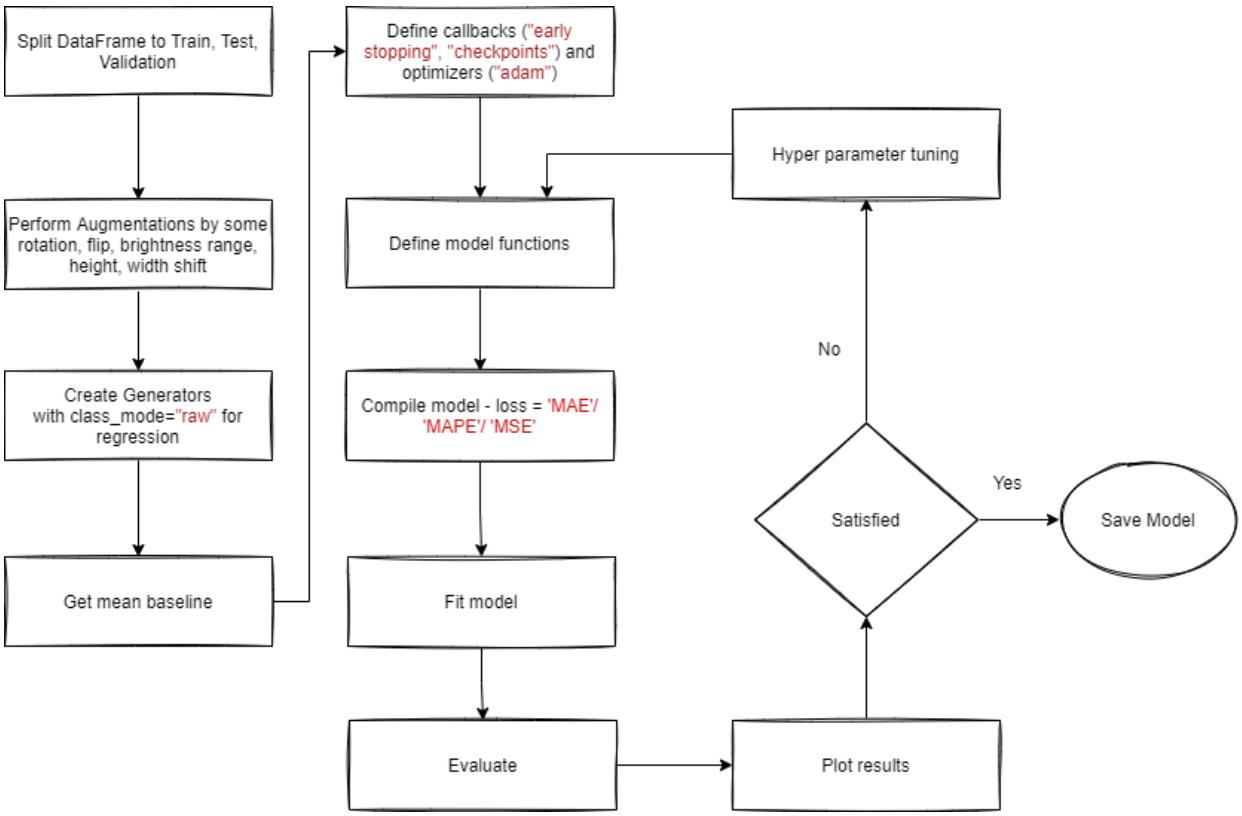


Figure 3.17. Code flow in Keras for yield prediction problem

3.8 Plant Growth

A growth cycle lasted between 10-14 days for the crops grown in this research. Light duration, nutrient solution concentration, and seeding density were varied based on the experimental design. The timeline of a typical cycle is shown in Figure 3.18. Seeds were sown in the growth trays lined with Biostrate substrate and kept for germination in dark for 2-3 days. The trays were later moved to the indoor farm wherein irrigation and light duration were handled by scheduled automations throughout the growth cycle. Two sub trays per tray were sampled to harvest on the last four days of the cycle (referred to as day n-3, n-2, n-1, and n with n being the length of the growth cycle for the corresponding crop). The height and weight measurements of the plants were carried out manually using a hand-held digital roller electric ruler (DZT1968, Guang Dong, China) least count 0.1cm and an electronic portable balance (SPX6201, Ohaus, Parsippany, New Jersey) least count 0.1gm. Once the data was captured, a statistical analysis to study factor effects on weight was performed according to the method displayed in Figure 3.19

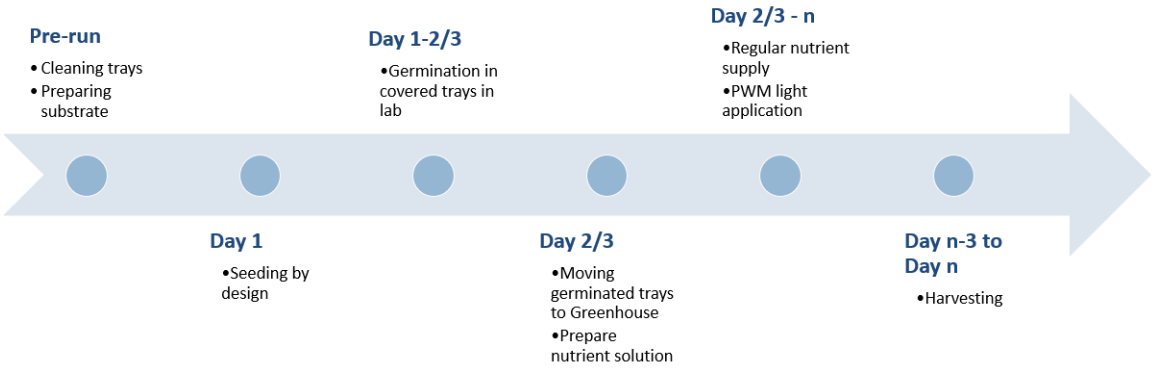


Figure 3.18. Timeline of a typical growth cycle of microgreens

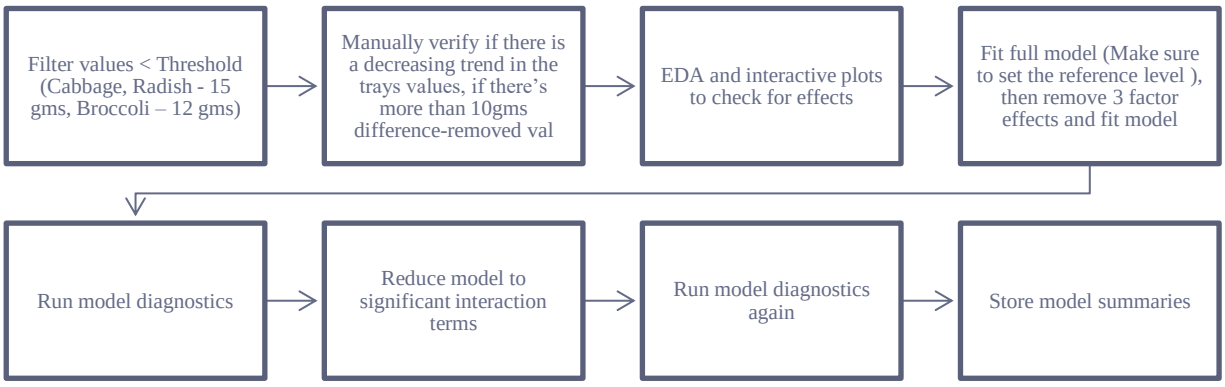


Figure 3.19 Analysis method for establishing factor-effects on microgreen weight

4. RESULTS AND DISCUSSION

4.1 Design and Implementation of the system

Numerous challenges were encountered during the system development. Emphasis had been placed on the safety, reliability, and reusability while design. The number of components had also been kept minimal and isolated enough to facilitate scaling. I have listed out a few aspects taken into consideration and challenges faced so that they can serve as a reference to practitioners.

4.1.1 Power supply

- There had been several system crashes of RPis triggered by power fluctuations despite using the specific rated adapters or sometimes even from the approved vendors. Hence, it was essential to choose a stable AC to DC adapter that could provide enough current for powering the sensors and the controller. Power supplies that have a slightly higher wattage than the required, but within the specified limits of the micro-processor were used for reliable operation.
- It was essential to plan for power supply provision for the whole setup of sensors. The current system required us to expand supply lines multiple times to accommodate more devices in a greenhouse that originally only had provision for powering LEDs.

4.1.2 Lighting module

- Individual dimmers were replaced with digital PWM dimming from a central controller which helped reduce hardware and achieve synchronized control of all the LEDs.
- To accurately measure PPFD incident on a horizontal surface, the sensor must be level. Hence a leveling plate was also bought from Apogee, spirit level was checked before each run to prevent inaccurate measurements.
- Calibration factors for the quantum sensor varied with LEDs due to non-ideal spectral response of the sensor in photosynthetically active radiation range. Thus, a co-ordination with the LED manufacturers was required to obtain detailed spectral intensity characteristics. This data was in turn uploaded into Apogee calibration tool available

publicly to obtain correction factors. For instance, LED B which had emphasis on blue and red wavelengths had an error of 11.8% in the reported PPFD without a correction factor (which was determined as 1.13 from the tool to bring the error within 5%)

- To prevent sudden fluctuation of voltages and tripping of mains due to sudden shutting down/ turning on of LEDs, as a good practice with switching of high voltages, a delay of 30s was maintained between the shutting off LEDs in the greenhouse. Additionally, the LEDs were gradually turned up to the desired intensity (soft start to minimize start-up surge current)
- Although the LED intensity has a linear response to control voltage, the variability in components and application makes it imperfect, especially with PWM. Hence, the voltage was adjusted iteratively by using the monitoring sensor to attain desired PPFD.
- Specifications of LEDs vary based on coverage area, the height at which LEDs are mounted and the standard used by the manufacturer. For instance, one of the LEDs had a light distribution angle of 120°. At a height on 45 cm and maximum voltage of 10V at the controller, a tray of microgreens receives only 2/3 of what it receives when at 30 cm.

4.1.3 Irrigation and water quality modules

- Lab-grade water quality sensors were expensive with the setup of three sensor circuits+ probes+ tentacle costing ~$750. Hence only one such sensor setup was built at each table and the irrigation module was programed to use it sequentially i.e., a tray was emptied to the measurement jug where the readings were taken, and this jug was cleaned out before draining of next tray begins. This added about 10 mins to the time it took to finish the irrigation cycle of each tray. Clean water was used to rinse the measuring jug between treatments to prevent contamination from earlier readings.
- Physical compatibility of components on the irrigation pipeline had been a huge challenge, especially because of the imposed budget constraints and ready availability. The current pipeline for input water supply is shown in Figure 4.1. Pump was chosen based on delivery height, distance, and voltage compatibility. Solenoid manifold was used to facilitate convenient splitting of the inflow to supply to 6 trays and voltage compatibility. Flow meter size was chosen based on the tubing size upstream. There was also the issue of hose, barbed or threaded connectors. Choice of tubing followed the choice of hardware

compatibility and flexibility required to reach from solenoid to tray. Thus, a carefully deliberated irrigation system planning is essential.

- No significant differences were observed between accuracy of flows between the cheap and expensive sensors in a constant pressure setup. However, the expensive flow sensors were truer to their specifications sheet and followed the number of pulses/liter number very closely, while the cheap sensors required heavy recalibration. Additionally, cheap sensors were more sensitive to the placement in the water line, flow rate (even within the range of the sensor) with occasional spurious values.
- To facilitate automated inflow and outflow, holding the tubing in place was imperative. So, the standard 1020 trays were modified by drilling a hole in the sidewall and securing with grommets to insert tubing. A safety hole was also drilled to prevent excess watering in case of flow sensor/ relay failure.
- Since the irrigation system carried nutrient solution, to steer clear of algae in the system it was considered ideal to block light from all the water flow routes by means of using black tubing or covering the solenoids.
- Water level sensors were connected to Arduino pins in two runs to test if they could be used to make irrigation decisions. But fixing them in position was challenging and their accuracy in sensing level, response speed was questionable. They could be used to detect the presence of water but might not be reliable enough for turning the motors on/ off.

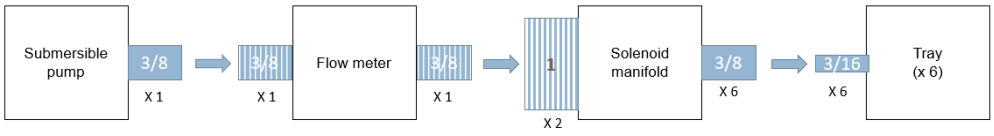


Figure 4.1. Inflow part of the irrigation pipeline

4.1.4 Climate modules

- BME680 was connected via I2C and was sensitive to length. So, sensor needed to be close to the MCU.
- From the calibration setup, it was found that the differences in the parameters between local climate sensors vs BME680 increased sharply when the temperature is high. BME680 showed sharp ups and downs in daytime temperatures as compared to ATMOS. One potential reason could be that ATMOS was housed in a radiation shield which might

smooth out changes in microclimate around the measurement area of the sensor. Pairs of DS18B20 and DHT11 that were physically located close by behaved similarly, with DHTs averaging a 0.6° higher measure than DSs. However there had been a difference between DS1 and DS3, which could be attributed to the distance in space between the sensors in the setup. Thus, it was observed that sensors of similar kind behave similarly and can be compared with each other, but comparison between sensors of different kinds measuring the same parameters should be made with caution. A few plots demonstrating these observations are shown in Figure 4.2.

Figure 4.2. Readings and differences between some sets of microclimate sensors

- It was observed that environment conditions around the central sensors were different from those around the trays. This was attributed to heating from LEDs, plant transpiration, difference in air circulation over the plants given they are more closed. To gauge this difference, sensors were placed on some trays. Regions below sheet LEDs turned out to be 1-3° C warmer than outside, while the ones below bar LEDs did not exceed 0.5° C.

4.1.5 Data and Connectivity

- To prevent loss of data, it was stored at multiple locations. It was captured and stored temporarily at the edge to account for network discontinuities and retrieved at the end of the cycle. It was also stored in a local DB which was periodically synced with the cloud and both the copies were retained.

- The frequency at which different data points were captured was chosen discretely. For e.g., BME680 can provide a reading at few second intervals (1-3s), but this application only read it every 5 mins as the indoor settings imply no sudden changes in environment conditions. By taking such measures, power consumed at the nodes was reduced. This also helped avoid data traffic, storage overflow and post-processing efforts to down-sample.
- The system in the greenhouse zone was behind a secure firewall and the only mode of two-way communication with devices was via MQTT over transport layer security (TLS). This MQTT broker was hosted on another secure server that also hosted dashboard, DBs.
- A static IP could not be obtained for the devices which created troubles with access via VNC as well as sending the data over HTTP. Hence a script to notify changes in IP was deployed on the nodes (RPis).

4.2 Real-Time Control and Monitoring

4.2.1 Experiment to measure latency in the system

Average latency measurements of raw and filtered measurements (removing >10s values) are provided in Table 4.1. The high values of average and standard deviation of latency in QoS 1,2 were observed around client reconnect period or occasional traffic at the broker and these values occur <1% of over 10,000 measurements. However, from a safety and precision standpoint, it is essential that our application handles these cases reliably. For e.g., with a cloud-based anomaly detection in the event of flow meter failure, a delay of 100 secs would imply 5l of water being sent to the tray which leads to heavy overflow. Hence, it is essential to have such functions run at edge.

Table 4.1. Raw and filtered measurements of one-way latency observed over 3 days

Case	Client	Subscribe QoS	Publish QoS	Raw measurements (ms)		Filtered measurements (ms)	
				Avg. Latency	Std. Dev. of Latency	Avg. Latency	Std. Dev. of Latency
1	Partner A	2	2	46	302	46	228
	Partner B	2	2	940	9784	62	342
2	Partner A	1	1	148	1758	67	470
	Partner B	1	1	6836	56535	49	446
3	Partner A	0	0	12	87	12	87
	Partner B	0	0	38	827	20	190
4	Partner A	2	1	17	155	16	104
	Partner B	0	0	279	5090	19	213

4.2.2 Analysis of local-cloud sync operation

A typical code block to sync a data table from local server to cloud along with the time taken for each step is shown in Figure 4.3 (averaged over 100 runs). It goes from establishing a secure shell connection to cloud to finally committing the changes in the corresponding cloud database. This profiling revealed that 80% of the time was consumed in operations related to establishing connectivity. Thus, minimizing the number of times this code is run helps reduce the resource utilization at the local DB. It in turn also reduces the data traffic both at the local gateway and cloud by reducing the number of times a connection is initiated to synchronize both DBs. An analysis of the two scenarios over 4 data tables in the DB is presented in Table 4.2. Scenario A) Data is uploaded from local DB into cloud every time a sensor sends data, B) Periodic update every 15 mins. Hence for sensors that do not relate to control operations and are not latency sensitive, a periodic synchronization with the cloud was carried out once every 15 mins.

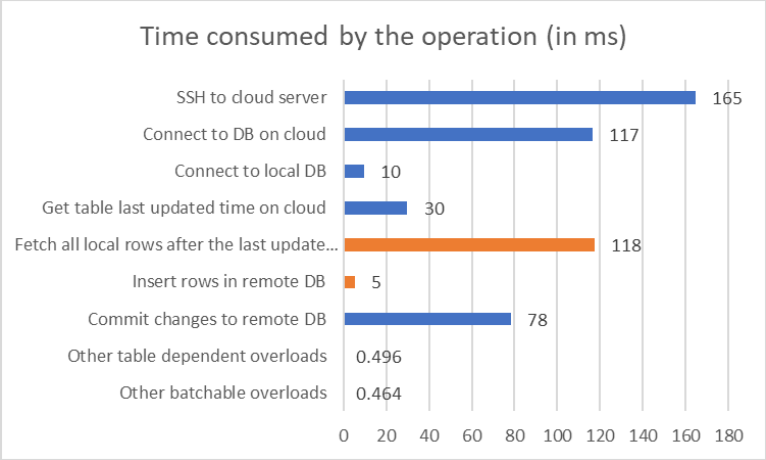


Figure 4.3 Time consumption distribution in a typical local<->cloud sync operation

Table 4.2. Analysis of time and memory consumption in scenarios A and B

	Scenario A	**Scenario B**
Num update operations	12	4
Num of times cronjob is run	45	1
Avg memory usage by cronjob/ program run	55 MiB	55 MiB
Amount of time Pi this memory would be occupied *{(Total connection times * Num of connections made) +* *(Total fetching time * Num of fetches) +* *(Total updating times * Num of updates)}*	34116 ms *{292 ms * 45 +* *148 ms * 45 * 3 +* *83 ms * 12}*	985 ms *{292 ms * 1 +* *148 ms * 3 +* *83 ms * 3}*
Bandwidth	100 kB/s *12	100 kB/s *4
Data Latency (max)	20 sec	15 mins

4.2.3 Adoption of results

A sample strategy of connectivity of nodes along with the functionalities that can be performed by them in-situ is shown in Figure 4.4. This framework was formulated after deliberating potential alternative paths to achieve minimization of latency in real-time applications, trade-off with latency and data traffic for passive sensor data and utilization of storage, computational capacity at the nodes.

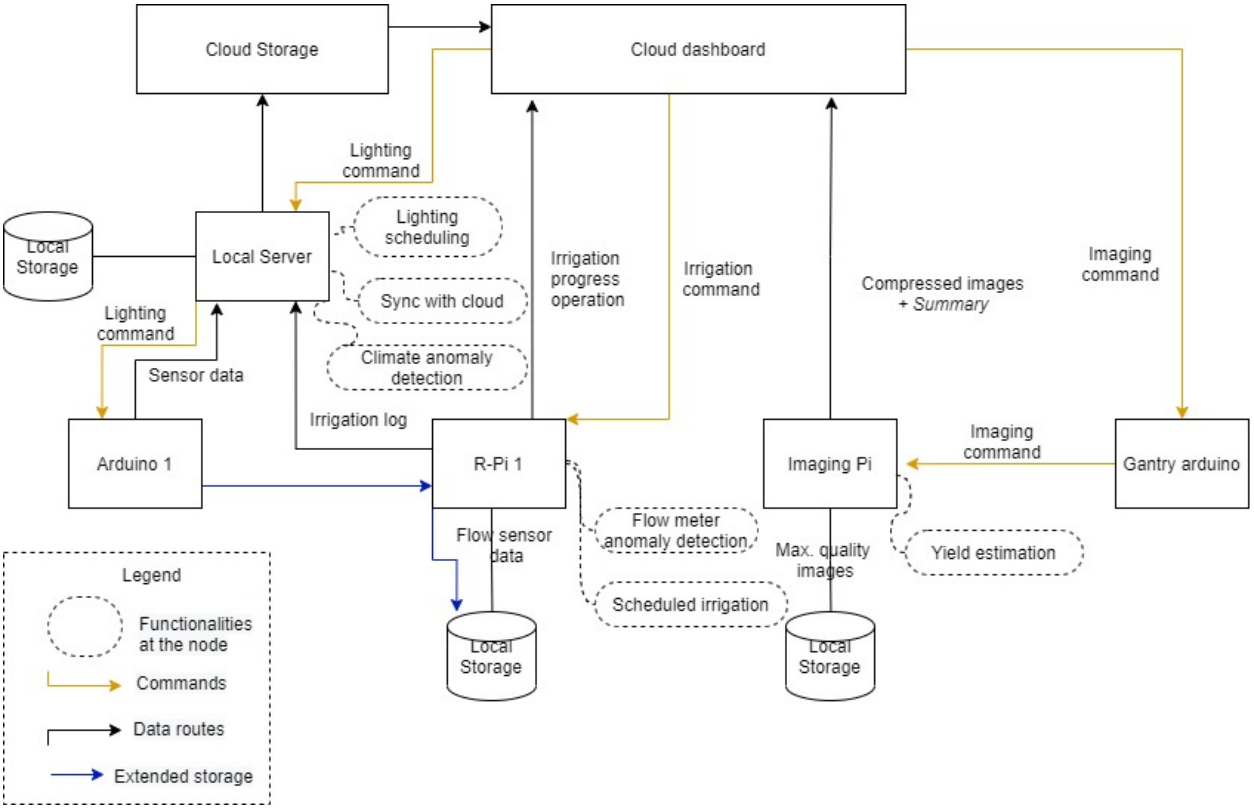


Figure 4.4. A typical edge-fog-cloud pipeline. Connectivity and functionalities at node devices

4.3 Anomaly Detection Algorithms

The data captured by BME680 sensor was used for anomaly detection experiments. The period of '4/29/2021' to '5/29/2021' had been the longest and most reliable after all the identified issues were fixed. So, this data is used for training. In the training set, 8196 data points were captured in 5 mins intervals with 5 hours of data missing on '5/5/2021' due to internet downtime. Hence, the data from this day was deleted to reduce impact on further machine learning tasks and maintain continuity. The rest of the data was resampled to 5 mins intervals which resulted in a total of 8479 points. Similarly, other chunks of data were used for testing, validation and their

summary are given in Table 4.3 with Test1 being the primary testing set. From this data outliers were removed to obtain clean datasets. Figure 4.5 which has density and box plots reveals that temperature, RH did not have any outliers, but CO2 data had a considerable number. So those points were removed and replaced with previous correct values. Once a clean data was obtained, anomalies were imputed based on the process described in 3.6.3. For instance, at 1% contamination rate, out of 8479 datapoints, 85 points were modified to introduce anomalies property by property. A plot of sensor values in period of interests in training and Test1 datasets is shown in Figure 4.6.

Table 4.3. Different periods of data considered for experiments of anomaly detection

Data	Number of samples	Number of anomalies (by contamination)		Period
		1%	5%	
Train	8479	85	422	04/29/2021 - 05/29/2021
Test1	2818	43	138	04/06/2021 - 04/15/2021
Test2	3315	34		10/16/2021 - 27/10/2021
Test3	1401	15		11/24/2021 - 11/29/2021
Validation	1186			03/25/2021 - 03/29/2021

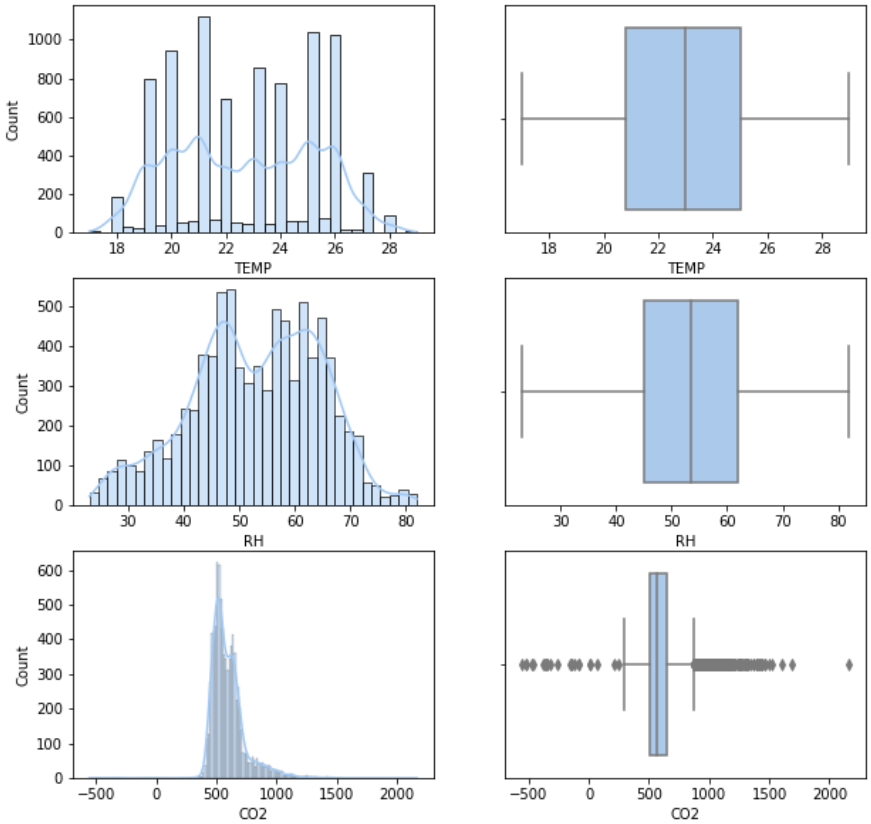


Figure 4.5. Density and box plots of BME680, Gravity v1.1 data for identification of outliers

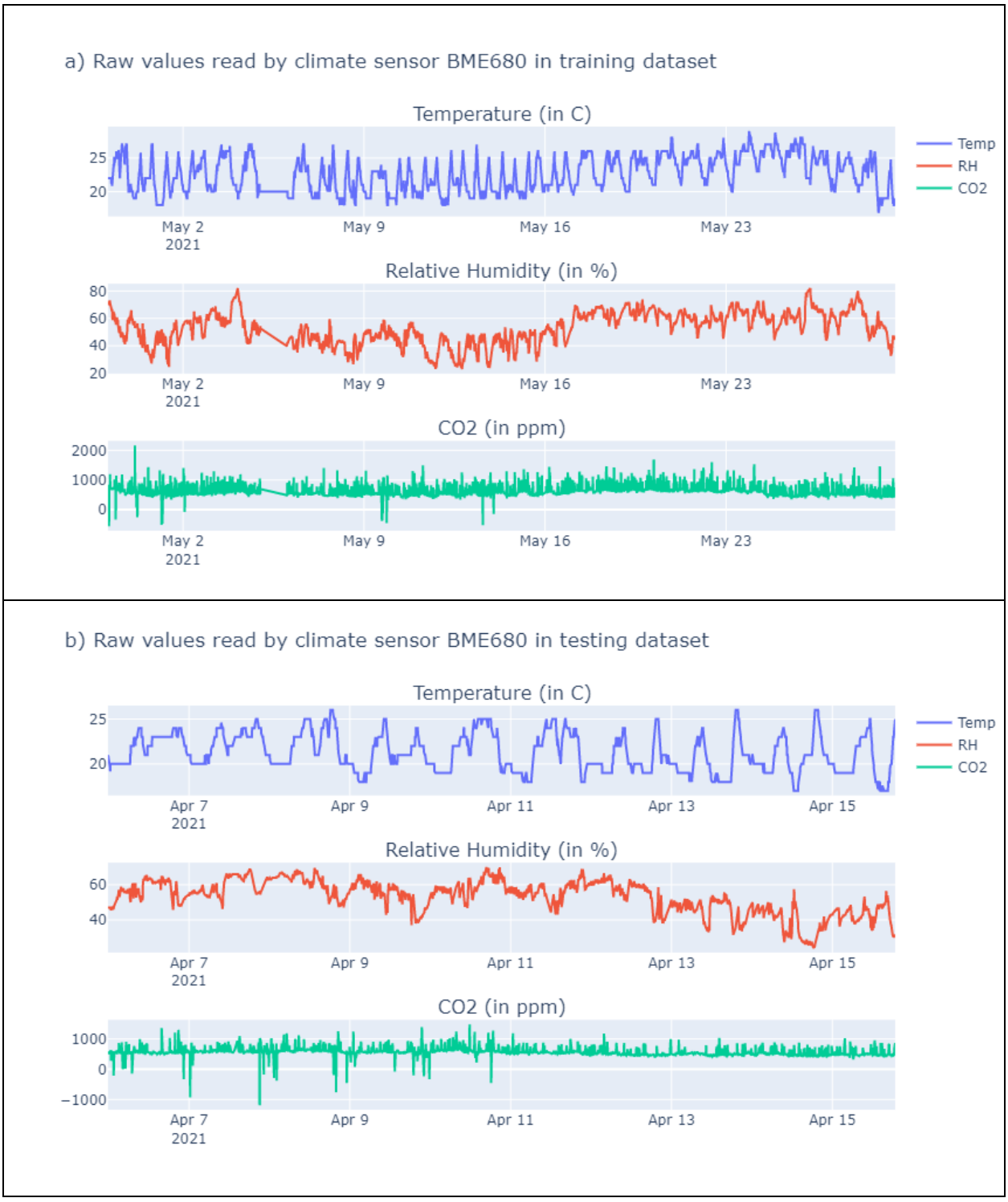


Figure 4.6. Time series plots of data captured by BME680 and Gravity 1.1 sensors by training and testing sets as well as different measured parameters

4.3.1 Experiments on temperature dataset

Comparison of all identified models

A comparison of performance between different models from an overall anomaly detection perspective in temperature dataset is shown in Table 4.4. It can clearly be seen that prediction-based methods are superior to unsupervised learning methods as they were able to capture more anomalies correctly with fewer false positives. Among the unsupervised models, density-based methods like KDE, HBOS, EE performed better albeit capturing only point anomalies most of the time. In prediction-based models, regressions perform surprisingly well, with performance at par with neural networks (Note that for prediction-based methods, these results correspond to a dataset processed at lag 3 value, i.e., past 3 values of the parameter were treated as features to predict the next interval). All-in-all prediction-based models were deemed better and chosen for further experiments and fine-tuning to derive better performance.

Table 4.4. A comparison of models used for detecting anomalies in temperature data in terms of the number of anomalies identified correctly based on the type.

Model	Train				Test			
	false positives	collective (47)	context (10)	point (30)	false positives	collective (16)	context (3)	point (9)
ANN	3	26	9	30	3	16	3	9
BayesianRidge	3	26	9	30	3	16	3	9
BiLSTM	3	26	9	30	3	16	3	9
ElasticNetCV	27	37	10	30	13	16	3	9
HuberRegressor	3	25	9	30	3	16	3	9
LSTM	3	26	9	30	3	16	3	9
LassoCV	3	26	9	30	3	16	3	9
LassoLarsCV	3	26	9	30	3	16	3	9
LinearRegression	3	26	9	30	3	16	3	9
PassiveAggressive Regressor	4	25	10	30	3	16	3	9
RidgeCV	3	26	9	30	3	16	3	9
KNN	0	0	0	23	0	0	0	9
iForest	24	0	1	20	0	0	0	9
LOF	55	0	0	0	26	0	0	0
GMM	0	0	0	28	0	0	0	8
CUSUM	22	0	0	22	12	0	0	7
EE	24	0	1	30	47	0	0	9
HBOS	16	0	1	30	45	0	0	9
KDE	52	5	1	30	53	0	0	9

Comparison of prediction-based models at different lags

Lag values 1 to 7 that correspond to 5 mins to 35 mins were experimented with as predictors for the next interval. False positive rates (FPR) are close to 0 for all the cases, given the nature of the problem. True positive rate (TPR), precision and root mean square error (RMSE) for the models under consideration on test data are shown in Table 4.5. RMSE remains constant with the increasing lag value implying that temperature values are best predicted just using a single previous interval excepting for gradient boosted regression (GBR). RMSE of GBR at a lag value of 2 is less than rest of the models for both train and test datasets which makes it the best predictor. However, precision and TPR are still the same as others which might be attributed to the fact that thresholds were incremented only in steps of 0.025 i.e., 2.5% relative error. Further experimentation with thresholds might be required to see if a better classification performance can be derived. Lag values of up to 4 are only shown in the table due to space constraints and the trend revealing no new information.

Table 4.5 Performance of prediction-based methods for different lag values on test data

	RMSE				PRECISION				TPR			
Model	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
ANN	0.27	0.27	0.27	0.27	96.55	93.33	90.32	87.5	65.12	65.12	65.12	65.12
BayesianRidge	0.27	0.27	0.27	0.26	96.55	93.33	90.32	87.5	65.12	65.12	65.12	65.12
BiLSTM	0.27	0.27	0.27	0.27	71.79	93.33	90.32	87.5	65.12	65.12	65.12	65.12
ElasticNetCV	0.27	0.27	0.27	0.27	71.79	70	68.29	87.5	65.12	65.12	65.12	65.12
GBR	0.27	0.23	0.23	0.23	71.79	93.33	71.79	66.67	65.12	65.12	65.12	65.12
HuberRegressor	0.27	0.27	0.28	0.3	96.55	93.33	90.32	87.5	65.12	65.12	65.12	65.12
LassoCV	0.27	0.27	0.27	0.27	71.79	70	90.32	87.5	65.12	65.12	65.12	65.12
LassoLarsCV	0.27	0.27	0.27	0.26	96.55	93.33	90.32	87.5	65.12	65.12	65.12	65.12
LinearRegression	0.27	0.27	0.27	0.26	96.55	93.33	90.32	87.5	65.12	65.12	65.12	65.12
LSTM	0.27	0.27	0.27	0.27	96.55	93.33	90.32	87.5	65.12	65.12	65.12	65.12
PassiveAggressive Regressor	0.32	0.31	0.31	0.37	96.55	93.33	90.32	87.5	65.12	65.12	65.12	65.12
RidgeCV	0.27	0.27	0.27	0.26	96.55	93.33	90.32	87.5	65.12	65.12	65.12	65.12

Comparison of prediction-based models with different temporal, windowing factors

A common practice is to add temporal features while dealing with time series. Hence, a feature ‘time of the day’ (calculated as hour*60 + minutes) is added as a predictor variable and performance was tested (Scenario T). Similarly, window related features like difference, mean,

standard deviation were added as predictors (Scenario W). Finally, both were tested together (Scenario T+W). LSTM, Bi-LSTM, GBR showed very slight improvement in RMSE for a lag value of 2. Higher lag values showed an improved performance, but still less than lag 2. Thus, finally a lag value of 2, with temporal and window features were chosen as predictor variables (referred to as WT2 and used in experiments further). The details of the experiment are given in Table 4.6.

Table 4.6 Performance of prediction-based methods with window, temporal features on test data

	RMSE			PRECISION			TPR		
Model	**T**	**W**	**T+W**	**T**	**W**	**T+W**	**T**	**W**	**T+W**
ANN	0.27	0.27	0.27	93.33	93.33	93.33	65.12	65.12	65.12
BayesianRidge	0.27	0.27	0.27	93.33	93.33	93.33	65.12	65.12	65.12
BiLSTM	0.27	0.25	0.25	93.33	93.33	93.33	65.12	65.12	65.12
ElasticNetCV	0.27	0.27	0.27	93.33	93.33	93.33	65.12	65.12	65.12
GBR	0.22	0.23	0.22	80	70	77.78	65.12	65.12	65.12
HuberRegressor	0.33	0.27	0.27	93.33	93.33	93.33	65.12	65.12	65.12
LassoCV	0.27	0.27	0.27	93.33	93.33	93.33	65.12	65.12	65.12
LassoLarsCV	0.27	0.27	0.27	93.33	93.33	93.33	65.12	65.12	65.12
LinearRegression	0.27	0.27	0.27	93.33	93.33	93.33	65.12	65.12	65.12
LSTM	0.27	0.26	0.26	93.33	68.29	93.33	65.12	65.12	65.12
PassiveAggressiveRegressor	0.33	0.29	0.29	93.33	93.33	93.33	65.12	65.12	65.12
RidgeCV	0.27	0.27	0.27	93.33	93.33	93.33	65.12	65.12	65.12

Comparison of prediction-based models for different test datasets

To verify if the models become obsolete after some time and to establish a frequency for re-training, models trained on the training period are tested on data from different time periods in the past (refer Table 4.3 for details of the time periods). Results are shown in Table 4.7. The generalization of GBR seems very poor, which again might be a thresholding problem. Rest of the models behave very similarly with linear regression outperforming rest in precision on Test1 dataset. This experiment revealed that in the context of indoor climate, frequent updating of models might not be necessary as testing with data from 6 months away did not deteriorate the performance achieved.

Table 4.7 Performance of prediction-based methods on different test sets

Model	PRECISION				TPR			
	Train	Test1	Test2	Test3	Train	Test1	Test2	Test3
ANN	97.06	93.33	84.38	87.5	75.86	65.12	79.41	93.33
BayesianRidge	97.01	93.33	87.5	87.5	74.71	65.12	82.35	93.33
BiLSTM	97.01	93.33	82.35	87.5	74.71	65.12	82.35	93.33
ElasticNetCV	97.01	93.33	87.5	87.5	74.71	65.12	82.35	93.33
GBR	88.37	77.78	17.65	58.33	87.36	65.12	97.06	93.33
HuberRegressor	97.01	93.33	84.85	87.5	74.71	65.12	82.35	93.33
LassoCV	97.01	93.33	87.5	87.5	74.71	65.12	82.35	93.33
LassoLarsCV	97.01	93.33	87.5	87.5	74.71	65.12	82.35	93.33
LinearRegression	97.01	93.33	90.32	87.5	74.71	65.12	82.35	93.33
LSTM	97.01	93.33	87.5	87.5	74.71	65.12	82.35	93.33
PassiveAggressiveRegressor	97.01	93.33	84.85	87.5	74.71	65.12	82.35	93.33
RidgeCV	97.01	93.33	87.5	87.5	74.71	65.12	82.35	93.33

Comparison of prediction-based models for different contamination rates in test data

The original train and test data were imputed with 1% anomalies, i.e., contamination rate = 0.01. Thus, the threshold was also chosen based on classification performance with respect to this rate. To check what happens in a real scenario if more anomalies start to appear, the same test data is made up to contamination rates of 0.05 and 0.1. It can be seen from Table 4.8 that point anomalies are identified correctly and there are close to 0 false positives. Collective and contextual anomalies exhibited poor performance. Upon deep dive into the data, it was observed that this was more because of the way the data was prepared for the experiment. For e.g., out of the collective anomalies that were not captured in most cases, 37 instances were <= 1° C from the original clean values that were replaced. GBR was able to capture a few of them between 0.8°C - 1° C because of its better prediction capability and tighter threshold. Therefore, larger dataset would be needed to impute anomalies discrete from actual values and test the performance. Theoretically however, it should not matter as the prediction is made for clean data and if 100% of the data deviates from the expected value, it should be reported.

4.3.2 Experiments on CO2 dataset

CO2 dataset provided a peculiar case. It was a noisy signal with too many fluctuations from interval to interval. All the prediction models failed with very poor precision-recall curves that

suggested thresholds over 0.7. This implies to label data that is more than 70% away from predicted value as anomaly. Still, an estimate of around 0.1- 0.15 was chosen based on ROC curve to check the trade-offs which explain the massive number of anomalies predicted, low precision in Table 4.9. Modifications of deep-learning models by introducing an extra layer and dropouts did not yield any results. KNN, HBOS managed to capture most of the point anomalies despite the noise, with KNN keeping up the status in test set as well. Noise removal or sensor fusion could be explored to reliably detect anomalies in this data.

Table 4.8 Performance of prediction-based methods at different contamination rates

Model	Contamination = 0.1				Contamination = 0.05			
	false positives	collective (151)	context (33)	point (99)	false positives	collective (73)	context (16)	point (49)
ANN	2	90	22	99	2	33	9	49
BayesianRidge	2	91	22	99	2	34	9	49
BiLSTM	2	91	22	99	2	33	9	49
ElasticNetCV	2	91	22	99	2	34	9	49
GBR	8	107	24	99	8	43	9	49
HuberRegressor	2	91	22	99	2	34	9	49
LSTM	2	91	22	99	2	33	9	49
LassoCV	2	91	22	99	2	34	9	49
LassoLarsCV	2	91	22	99	2	34	9	49
LinearRegression	2	91	22	99	2	34	9	49
PassiveAggressive Regressor	2	90	22	99	2	33	9	49
RidgeCV	2	91	22	99	2	34	9	49

Table 4.9. A comparison of models used for detecting anomalies in CO2 data in terms of false positive rate, true positive rate, precision and RMSE when applicable.

	Training dataset (N_ano = 87)					Test dataset (N_ano = 43)				
Model	**N_ano**	**FPR**	**TPR**	**Prec**	**RMSE**	**N_ano**	**FPR**	**TPR**	**Prec**	**RMSE**
CUMSUM	23	0	0.18	0.7		4	0	0.09	1	
EE	84	0.01	0.36	0.37		9	0	0.21	1	
GMM		0	0	0			0	0	0	
HBOS	38	0	0.36	0.82		18	0	0.19	0.44	
KDE	85	0.01	0.36	0.36		42	0.01	0.23	0.24	
KNN	30	0	0.34	1		11	0	0.19	0.73	
LOF	61	0.01	0	0		25	0.01	0.02	0.04	
iForest	84	0.01	0.32	0.33		28	0.01	0.21	0.32	
LSTM	1640	0.19	0.71	0.04	86.06	549	0.19	0.44	0.03	71.08
PassiveAggressive Regressor	1094	0.13	0.48	0.04	82.42	292	0.1	0.35	0.05	73.08

Table 4.9 continued

HuberRegressor	1670	0.19	0.75	0.04	71.95	550	0.19	0.58	0.05	61.88
BiLSTM	2144	0.25	0.75	0.03	71.46	739	0.26	0.53	0.03	64.93
ANN	1713	0.2	0.74	0.04	71.08	553	0.19	0.56	0.04	61.15
LassoLarsCV	1803	0.21	0.67	0.03	70.54	595	0.21	0.53	0.04	60.76
LassoCV	1803	0.21	0.68	0.03	70.54	596	0.21	0.53	0.04	60.77
ElasticNetCV	1798	0.21	0.68	0.03	70.53	594	0.21	0.53	0.04	60.78
BayesianRidge	1798	0.21	0.68	0.03	70.52	592	0.21	0.53	0.04	60.79
RidgeCV	1798	0.21	0.68	0.03	70.52	592	0.21	0.53	0.04	60.79
LinearRegression	1769	0.2	0.69	0.03	70.51	579	0.2	0.53	0.04	60.72

4.3.3 Adoption of results

With a comprehensive consideration of the modeling experiments and real-world scenarios, the following anomaly detection framework is suggested.

1. Script running on cloud server to check for updates in database and point suggestive actions to the user (Figure 4.7. Algorithm for system monitoring at cloudFigure 4.7) based on fault localization. The actions and operating procedures should be learnt, standardized over time for a robust system with updating of the faults database when a new issue is noted.
2. Deployment of algorithms at the edge node (Figure 4.8). In practice, point anomalies do not usually reflect any control failures, but they lead to biased analysis later. Hence, a suitable algorithm can be used to flag anomalies (which is also a regression model in the case of temperature data) and later a regression model can be used to predict, replace the anomalous point before updating to database.
3. Periodic model update at the node given the seasonality and change in conditions/ crop cycles to keep the model relevant.

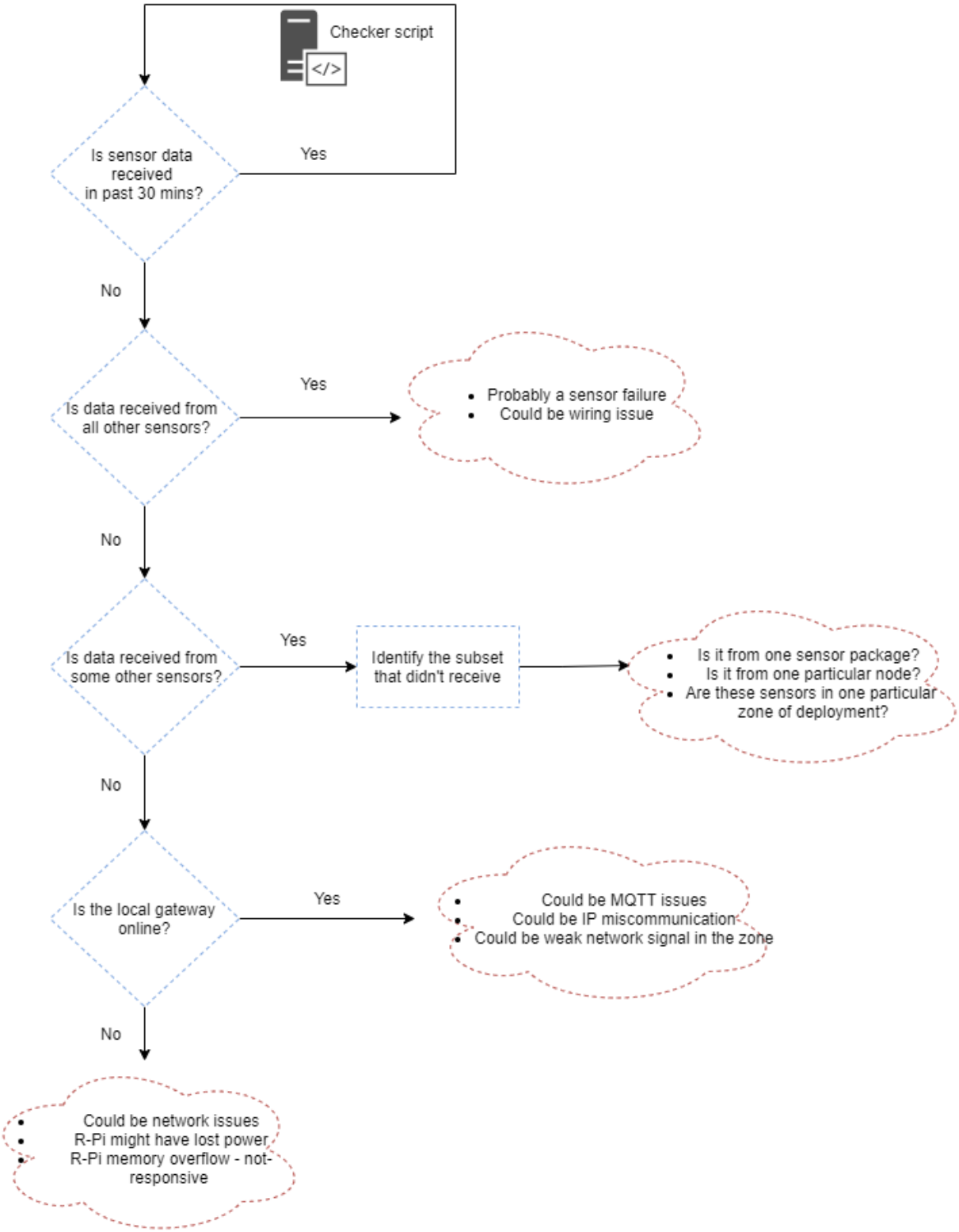


Figure 4.7. Algorithm for system monitoring at cloud

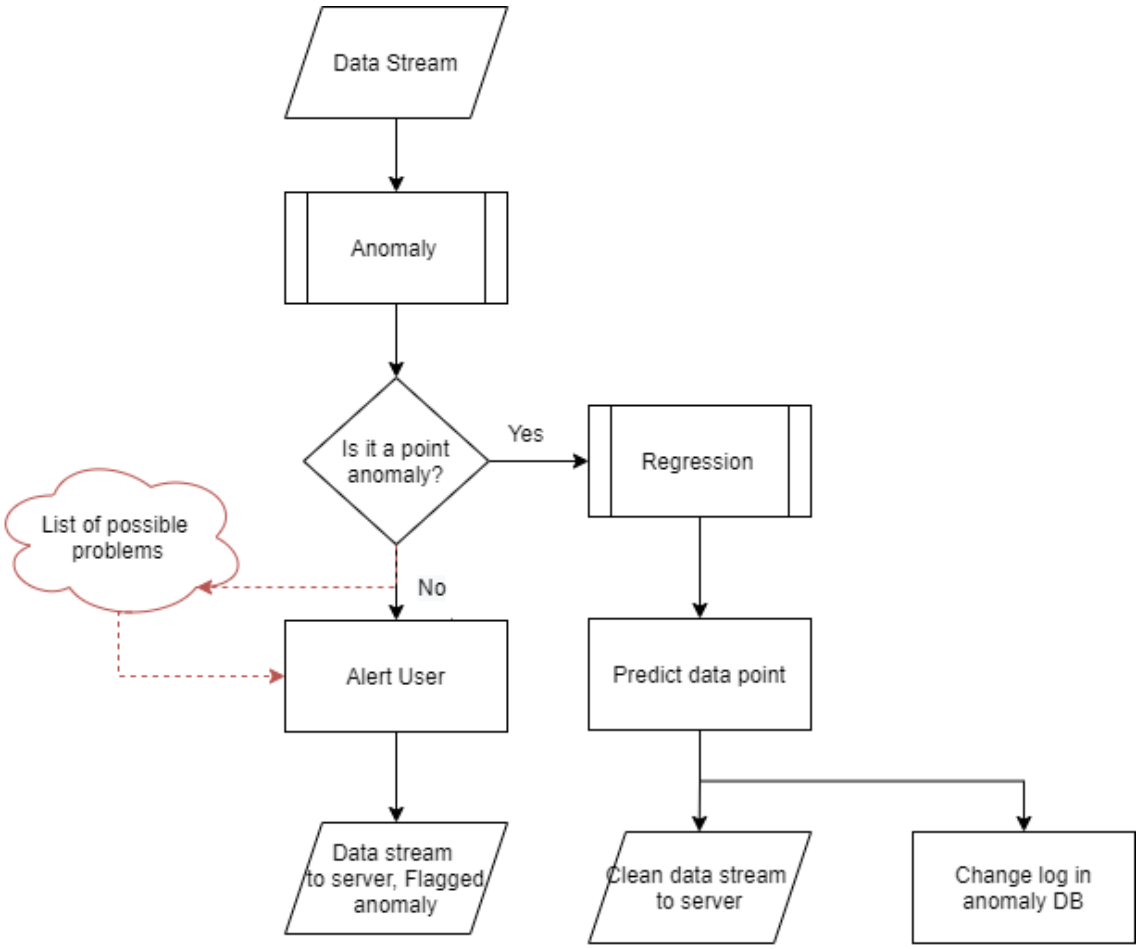


Figure 4.8. Flow of events in an anomaly detection pipeline at edge node

4.4 Yield Prediction Algorithms

Over two runs of Broccoli, six days of data was available which made up to a total of 170 images. Histogram of the filtered harvested weights is shown in Figure 4.9. With these weights as target variable and images as predictors, deep learning models were run.

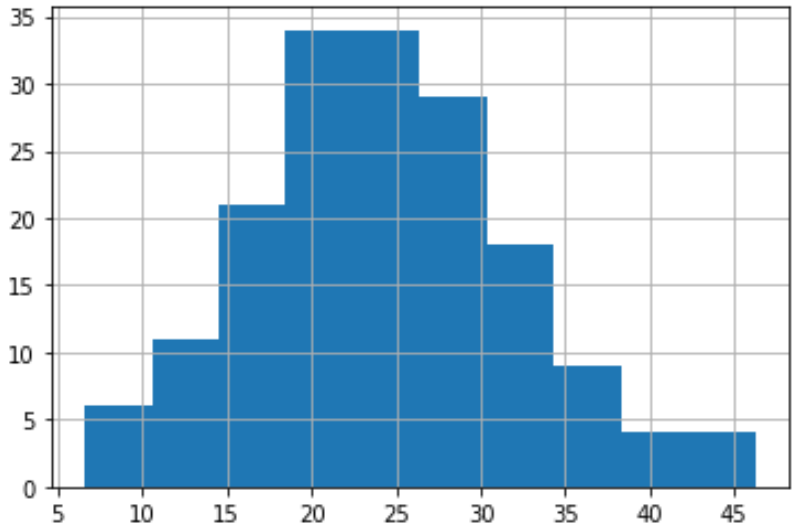


Figure 4.9. Distribution of weights of Broccoli microgreens in the Data Frame

4.4.1 Results

The train, validation, test datapoints were 137,17,16 in number, respectively. A baseline for the metrics mean absolute error (MAE), MAPE is created by assuming that training mean is the predicted value for all validation data. Rest of the models were compared against this naïve baseline which is 23% for this dataset and is shown by the red horizontal line in Figure 4.10. A freedom to train to 100 epochs was given to the model, but they stop at different points due to the implementation of early stopping callback which was a very generous 0.25 improvement of MAE over 10 epochs. The overall performance on validation and testing data, number of epochs the models ran to are shown in Table 4.10. Transfer learning models converged faster than the SmallCNN as anticipated. However, the confidence intervals of these predictions are questionable for two reasons. 1) The size of the dataset is too small to generalize. The model would result in very different values when run with different seeds. 2) The images are very similar to each other, especially at later stages of growth and in real-world too, yield is a function of leaf cover and height. Including a height data and other relevant parameters with respect to differential nutrient solutions etc. might help predictions better. Other CNN architectures that are discussed in literature for leaf-counting applications can be explored.

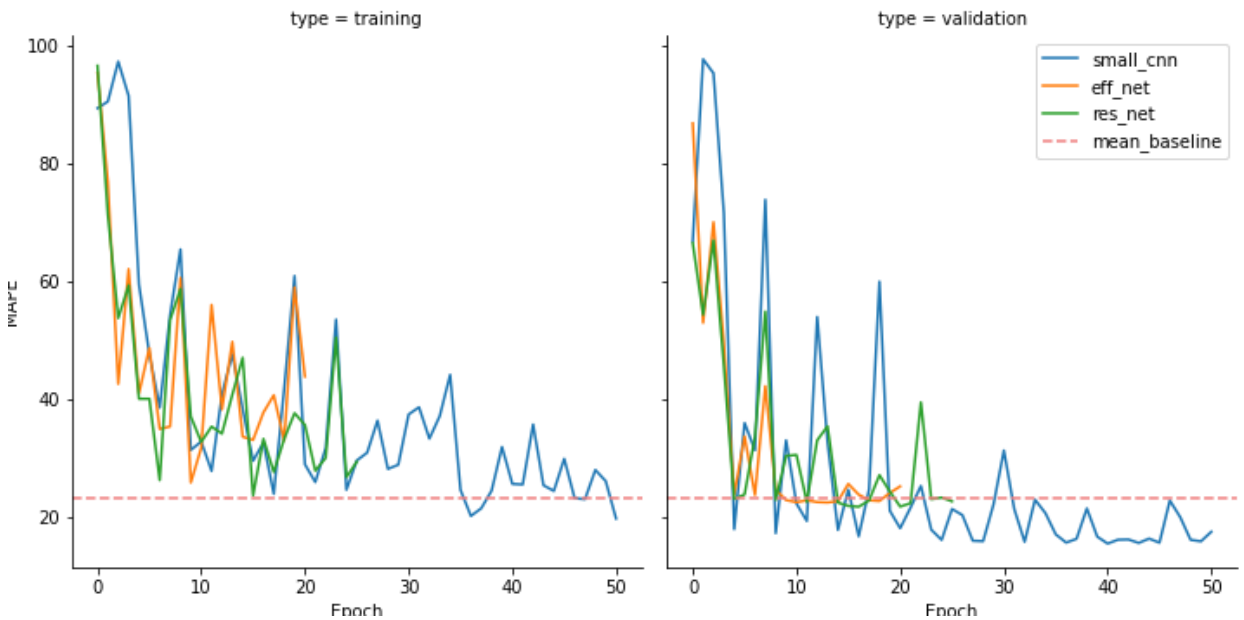


Figure 4.10. Training and validation MAPE plotted against number of epochs

Table 4.10. Performance metrics of neural networks on predicting yield

Model	No. of epochs	Val MAE	Val MAPE	Test MAE	Test MAPE
ResNet50	24	6.25	22.30%	6.92	32.20%
EffNetB0	20	6.96	22.49%	7.13	30.41%
SmallCNN	50	5.82	12.21%	6.76	29.79%

4.5 Evaluation using plant growth

The system was deployed 12 months ago in the greenhouse and functionalities were added to it on a gradual basis to achieve the configuration discussed in the paper. It had been used to collect over 300k sensor measurements and 5000 images of various experiments. It was used to successfully grow 5 batches of different microgreens (+2 test runs) under variable application of nutrition, light intensity, and seed density.

The yields of these batches per tray by harvest day are shown in Figure 4.11. Cabbage was the first experiment at a scale of 3 benches and its yield suffered due to erratic irrigation. Radish run 2 was the cleanest of all runs, with new grommets installed and sealed in place, safety hole to prevent overwatering which is very different from radish run 1 where there were issues due to trays over watering. Both broccolis run 1 and 2 showed similar pattern of growth, but broccoli 2

yield was slightly lower which might have been a result of rising temperature in the greenhouse or difference in nutrient solutions or few hours difference in harvesting times.

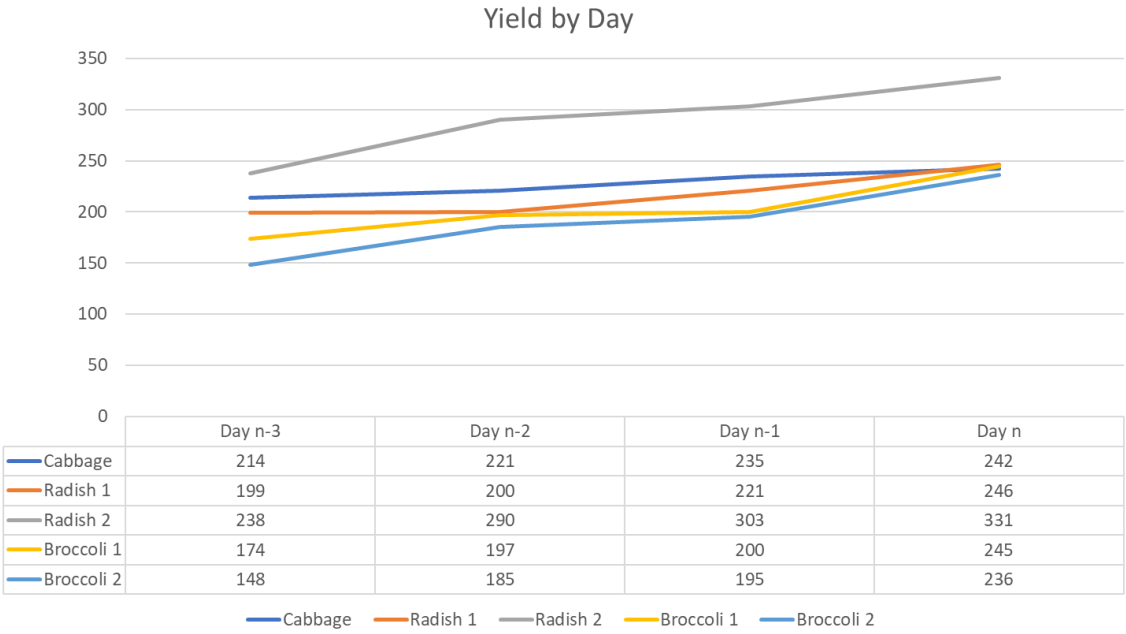


Figure 4.11 Microgreen yield (weight in grams/ tray) by crop by day of harvest

Based on the method discussed in Figure 3.19, analysis was performed on the yield data obtained from the previously mentioned runs. The significance of factor effects is shown in Table 4.11. Density as expected had a significant effect on all the crops, but the impact varied from crop to crop. For instance, for radish, yields in density 1 oz are on an average 29% higher as opposed to density 0.75 oz across all nutrient solutions and lights whereas for broccoli it was 23% higher, cabbage 5% higher. It can be observed that nutrient solution was a significant effect for cabbage and radish, but not broccoli. The EC values tested for the former were 1.2-1.6 mS/cm as opposed to for the latter which were 1.4-1.8 mS/cm, hence the greens might have received abundant nutrients already to show any significant difference. The density-nutrient solution interaction is more pronounced in radish with density 1 oz at concentration 1.2 mS/cm being 15% lower than that at 1.6 mS/cm.

It is evident from the graph earlier that weight increased by day and hence, day would have a significant effect. An interesting point to note in radish was that increase in day n-2 to day n-1 is not significant with a mere 5% change in average yield, but it picks up again by day n with a

difference of 23% between day 12 and day 14. This could be a considerable factor in performing cost-benefit analysis.

Although light did not have any significant impact standalone, there were significant interaction effects with density or nutrient solution in radish. Higher concentration of nutrient solution under LED B did not show any significant impact but had 13-20% improvement over the lowest concentration under LEDs A, C. Within a given light density 1 oz is better than 0.75 oz, but by the light, density interaction, density 1 oz under LED C is 9% higher than the same density under LED B. For cabbage, density 0.75 oz under LED B had no difference compared to density 1 oz under LED A and C and 1 oz under LED B is at least 12% greater than all the other LED-density combinations.

Table 4.11 Results of analysis of variance for factor effects on weight for different microgreens

Factor	**Cabbage**	**Radish (Run 2)**	**Broccoli (Run 1,2)**
Light			
Density	.	***	***
Nutrient Sol	**	***	
Day	***	***	***
Light*Density	**	**	
Light* Nutrient Sol		*	
Density* Nutrient Sol		***	
Light*Day			
Nutrient Sol*Day			
Density*Day			
Light* Nutrient Sol*Day			
Density* Nutrient Sol*Day			
Light*Density*Day	.		
Light* Nutrient Sol*Density			

Significance codes : 0 ‘***’; 0.001 ‘**’; 0.01 ‘*’; 0.05 ‘.’

5. CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

An automated IoT-based testbed was developed by utilizing low-cost off-the-shelf sensors and components. This is one of the few studies that had implemented a system at a laboratory scale, conducted long duration experiments to report on anomalies with low-cost sensors and actuators, and reported the outcome in a detailed manner. A discussion on implementation challenges related to sensor calibrations in an indoor farm, design of irrigation systems, and providing reliable power supplies to electronics, is expected to help during future enhancements to the current design.

In establishing a reliable control remotely and setting up the dashboard, challenges with communication protocols, latency in operations were overcome. A user dashboard was developed from scratch. Edge computing was utilized to mobilize resources and realize various functionalities.

To ensure reliable operations and cleanliness of data, anomaly detection was implemented. Many techniques and experiments were performed to identify models that best detects anomalies. A practical and novel framework for fault localization in the sensor data capture pipeline was developed.

A novel imaging system was built to explore yield prediction from images. A well performing model could not be identified with the data at hand and this aspect will require further exploration.

Effects of lighting, nutrition solution concentration, seed density, and day of harvest on the growth of microgreens was evaluated using a split-plot design. Different microgreens showed different results. Light did not have a significant effect by itself in all cases but had some significant two-factor interaction effects. Nutrient concentration of 1.2 mS/cm had significantly lower yields than higher concentrations but increasing it beyond 1.4 mS/cm did not make any difference to the growth. Seeding density of 1oz had more yield than 0.75oz, but the percentage gain varied with nutrient concentration or lighting in cabbage and radish. Yield increased by the day of harvest, but sharp increases were noticed between first two days of harvest whereas, the rate of growth was slower beyond the second harvest day.

This study can be seen as a start point of bringing in more automation, data-centric approach into microgreen growth or any hydroponic system and there are several aspects worthy of exploration.

5.2 Future Scope

The overall goal of the study was to develop a system with low-cost, IoT devices driven indoor farm with reliable real-time remote control and move towards an intelligent farm management system. Some aspects that can be pursued in the future to advance these goals are mentioned under different headers as follows:

5.2.1 System

- In terms of choice of MCUs, cheaper alternatives like Pi-zero or particle photons or ESP8266 can be chosen. These choices were not explored due to a large inventory of RPis available in the lab and flexibility to experiment with sensors and actuators was a major consideration. Now that major features of the system have been designed, all these features could be implemented with the suggested alternatives in a more compact way. Cheaper chips could also facilitate for trade-offs between using more nodes, less wiring or more wiring originating from fewer nodes while keeping the cost of the system at a similar level.
- Weather-proof packaging of the solution developed for commercial use and deploying in farm without interfering with operations will be a ground level challenge. This challenge is furthermore a motivation to explore vision-based applications using something like the imaging system designed in this study.
- Advances are being made in the field of IoT at a rapid pace and the capabilities, costs of commercial cloud platforms are becoming competitive. Different IoT platforms like Thingspeak, Google cloud platform, IBM Watson IoT, Azure IoT can be contrasted as opposed to developing a dashboard from scratch for mid-scale operation, if the data ownership policies are not an issue.

5.2.2 Algorithms

- Developing a decision support system with prescriptive actions can help bridge the skill-gap between field workers and technology. This requires capturing sufficient data across multiple runs and exploring further applications like nutrient requirement, predictive maintenance suggestions.
- Opportunities with thermal images captured by the system need further exploration.

- Different models need to be explored for improving yield prediction and a reliable height measurement system is to be experimented with before integrating with the current design.
- Flow meter anomaly detection requires further exploration to improve precision in irrigation and reduce the response time required in flagging of erroneous events.

5.2.3 Databases

- Better data management strategies to facilitate easy integration of new sensors into the system should be investigated. NoSQL could be one option to explore.
- In the current setup, more images make the application very heavy as they are stored in a docker-container that is packaged along with the user application. The storage system as such is very primitive and optimizations can be brought in by making choices for ‘where and when to store what’.

5.2.4 Miscellaneous

- Further research is needed to develop affordable and precise moisture sensors for thin hydroponic substrates like Biostrate. Alternate methods for making irrigation decisions by replacing moisture sensors with a vision-based or nutrient-uptake based method could be explored.
- Stronger growth trays with provision for in and outflows are to be considered for reusability and for reducing material movement in indoor farms.
- Harvesting and seeding are the next labor-intensive tasks where precision is required for seeding and time management for harvesting. Automation for these operations is to be explored.

APPENDIX A. COST OF COMPONENTS IN THE SYSTEM

Component	Cost/ component	Num in this Experiment	Total Cost
IRRIGATION SYSTEM			
Solenoid valves	$28	18	$504
Flow Meters	$160	9	$1,440
Sensor setup	$750	3	$2,250
Inflow Pumps	$50	9	$450
Outflow pumps	$12	63	$756
Relays	$33	9	$297
Power Supply	$10	12	$120
Tubing			$300
LIGHTING SYSTEM			
PWM	$50	3	$150
PAR Sensors	$269	9	$2,421
Power Supply	$20	3	$60
IMAGING SYSTEM			
Gantry Rails	$27	6	$162
Thermal Sensors	$110	2	$220
IR Proximity Sensors	$15	2	$30
Pi Cameras	$30	2	$60
RPis + kit + SD Card	$70	2	$140
Steppers + Belt + Pulleys + Drivers	$46	4	$184
Arduino	$40	1	$40
Wiring + Power Outlets + PCB + Power Adapters	$82	1	$82
CLIMATE			
Gravity CO2 + BME680	$82	1	$82
DHT + DS local climate sensors	$6	3	$18
CONTROLLERS			
RPi	$70	4	$280
Arduino	$40	3	$120
OTHERS			
Power Supplies (General 5V,9V)	$20	3	$60
Router	$130	1	$130
TOTAL*			$10,356

*These costs are added costs from the experiment over the basic infrastructure needed for growing i.e., benches, LEDs, reservoirs for water/ nutrient solution, growth trays

REFERENCES

Abdallah, M., Lee, W. J., Raghunathan, N., Mousoulis, C., Sutherland, J. W., & Bagchi, S. (2021). Anomaly detection through transfer learning in agriculture and manufacturing IoT systems. Retrieved from http://arxiv.org/abs/2102.05814

Adenaeuer, L. (2014). Up, up and away! The economics of vertical farming. *J. of Agric. Stud.*, 2, 40–60. https://doi.org/10.5296/jas.v2i1.4526

Ahmad, M., Ishtiaq, A., Habib, M. A., & Ahmed, S. H. (2019). A review of internet of things (IoT) connectivity techniques. In M. A. Jan, F. Khan, & M. Alam (Eds.), *Recent Trends and Adv. in Wireless and IoT-Enabled Netw.* (pp. 25–36). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-99966-1_3

Aich, S., & Stavness, I. (2017). Leaf counting with deep convolutional and deconvolutional networks. *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, pp. 2080-2089. Piscataway, NJ: IEEE.

AlShrouf, A. (2017). Hydroponics, aeroponic and aquaponic as compared with conventional farming. *Am. Scientific Res. J. Eng., Technol., Sci. (ASRJETS)*, 27(1), 247–255. https://asrjetsjournal.org/index.php/American_Scientific_Journal/article/view/2543

Astill, G., Perez, A., & Thornsbury, S. (2020). Developing automation and mechanization for specialty crops: A review of U.S. department of agriculture programs. Administrative publication 082. Washington, DC: USDA-ERS. Retrieved October 30, 2020 from https://www.ers.usda.gov/webdocs/publications/95828/ap-082.pdf?v=561

Ayadi, A., Ghorbel, O., Obeid, A. M., & Abid, M. (2017). Outlier detection approaches for wireless sensor networks: A survey. *Comput. Netw.*, 129, 319–333. https://doi.org/10.1016/J.COMNET.2017.10.007

Basso, B., Cammarano, D., & Carfagna, E. (2013). Review of crop yield forecasting methods and early warning systems. *Proc. First Meet. of the Sci. Advis. Comm. of the Glob. Strateg. to Improv. Agric. and Rural Stat.* FAO Headquarters, Rome, Italy.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Comput. Surv. (CSUR)*, 41(3), 1-58. https://doi.org/10.1145/1541880.1541882

Chang, C.-L., Tian, J.-Y., Fu, W.-L., & Chang, K.-P. (2019). Integrated monitoring platform of plant growth based on IoT edge computing in greenhouse. ASABE Paper No. FL19-01. St. Joseph, MI: ASABE. https://doi.org/10.13031/FL201901

Chaudhary, P., Godara, S., Cheeran, A. N., & Chaudhari, A. (2012). Fast and accurate method for leaf area measurement. *Int. J. of Comput. Appl.*, 49(9), 22–25. https://doi.org/10.5120/7655-0757

Danita, M., Mathew, B., Shereen, N., Sharon, N., & Paul, J. J. (2019). IoT based automated greenhouse monitoring system. *Proc. 2nd Int. Conf. on Intell. Comput. and Control Syst., ICICCS 2018*, pp. 1933–1937. Piscataway, NJ: IEEE https://doi.org/10.1109/ICCONS.2018.8662911

de Souza, P. S. S., Rubin, F. P., Hohemberger, R., Ferreto, T. C., Lorenzon, A. F., Luizelli, M. C., & Rossi, F. D. (2020). Detecting abnormal sensors via machine learning: An IoT farming WSN-based architecture case study. *Measurement*, 164, 108042. https://doi.org/10.1016/j.measurement.2020.108042

Elijah, O., Rahman, T. A., Orikumhi, I., Leow, C. Y., & Hindia, M. N. (2018). An overview of internet of things (IoT) and data analytics in agriculture: Benefits and challenges. *IEEE Internet of Things J.*, 5(5), 3758–3773. https://doi.org/10.1109/JIOT.2018.2844296

Foorthuis, R. (2020). On the nature and types of anomalies: a review of deviations in data. Retrieved from https://arxiv.org/abs/2007.15634v3

Franchetti, B., Ntouskos, V., Giuliani, P., Herman, T., Barnes, L., & Pirri, F. (2019). Vision based modeling of plants phenotyping in vertical farming under artificial lighting. *MDPI Sensors*, 19(20), 4378. https://doi.org/10.3390/s19204378

Gaddam, A., Wilkin, T., Angelova, M., & Gaddam, J. (2020). Detecting sensor faults, anomalies and outliers in the internet of things: A survey on the challenges and solutions. *Electronics*, 9(3), 511. https://doi.org/10.3390/ELECTRONICS9030511

Gertphol, S., Chulaka, P., & Changmai, T. (2018). Predictive models for lettuce quality from internet of things-based hydroponic farm. *Proc. 22nd Int. Comput. Sci. and Eng. Conf., (ICSEC), pp. 1-5* . Piscataway, NJ: IEEE https://doi.org/10.1109/ICSEC.2018.8712676

Glaroudis, D., Iossifides, A., & Chatzimisios, P. (2020). Survey, comparison and research challenges of IoT application protocols for smart farming. *Comput. Netw.*, 168, 107037. https://doi.org/10.1016/j.comnet.2019.107037

Gómez, C., Currey, C. J., Dickson, R. W., Kim, H.-J., Hernández, R., Sabeh, N. C., … Burnett, S. E. (2019). Controlled environment food production for urban agriculture. *HortScience*, 54(9), 1448–1458. https://doi.org/10.21273/HORTSCI14073-19

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. and Pattern Recognit.*, pp. 770–778. Piscataway, NJ: IEEE. https://arxiv.org/abs/1512.03385v1

Hemming, S., Zwart, F. de, Elings, A., Righini, I., & Petropoulou, A. (2019). Remote control of greenhouse vegetable production with artificial intelligence—greenhouse climate, irrigation, and crop production. *Sensors*, 19(8), 1807. https://doi.org/10.3390/s19081807

Huang, Y., & Niu, J. (2016). A review of the advance of HVAC technologies as witnessed in ENB publications in the period from 1987 to 2014. *Energy and Build.*, 130, 33–45. https://doi.org/10.1016/j.enbuild.2016.08.036

Itzhaky, Y., Farjon, G., Khoroshevsky, F., Shpigler, A., & Bar-Hillel, A. (2018). Leaf counting: Multiple scale regression and detection using deep CNNs. *Proc. British Mach. Vis. Conf.*, pp. 328. Durham, UK: BMVA

Jaiswal, H., Singuluri, R., & Sampson, S. A. (2019). IoT and machine learning based approach for fully automated greenhouse. *Proc. 2019 IEEE Bombay Sect. Signat. Conf., (IBSSC),* pp. 1-6. Piscataway, NJ: IEEE. https://doi.org/10.1109/IBSSC47189.2019.8973086

Karimanzira, D., Na, C., Hong, M., & Wei, Y. (2021). Intelligent information management in aquaponics to increase mutual benefits. *Intell. Inf. Manag.*, 13(1), 50–69. https://doi.org/10.4236/iim.2021.131003

Khanna, A., & Kaur, S. (2019). Evolution of internet of things (IoT) and its significant impact in the field of precision agriculture. *Comput. Electron. Agric.*, 157, 218–231. https://doi.org/10.1016/j.compag.2018.12.039

Li, Z., Guo, R., Li, M., Chen, Y., & Li, G. (2020). A review of computer vision technologies for plant phenotyping. *Comput. Electron. Agric.*, *176*, 105672. https://doi.org/10.1016/j.compag.2020.105672

Liakos, K., Busato, P., Moshou, D., Pearson, S., & Bochtis, D. (2018). Machine learning in agriculture: A review. *Sensors*, 18(8), 2674. https://doi.org/10.3390/s18082674

Lin, K., Wu, J., Chen, J., & Si, H. (2013). A real time image segmentation approach for crop leaf. *Proc. 5th Conf. Meas. Technol. and Mechatron. Autom., ICMTMA*, pp. 74–77. Washington, DC: IEEE Computer Society. https://doi.org/10.1109/ICMTMA.2013.30

Lin, Y.-B., Lin, Y.-W., Lin, J.-Y., & Hung, H.-N. (2019). Sensortalk: An IoT device failure detection and calibration mechanism for smart farming. *Sensors*, 19(21), 4788. https://doi.org/10.3390/s19214788

Lin, Y.-W., Lin, Y.-B., & Hung, H.-N. (2020). CalibrationTalk: A farming sensor failure detection and calibration technique. *IEEE Internet of Things J.*, 1-1. https://doi.org/10.1109/JIOT.2020.3036859

Liu, Y., Pang, Z., Karlsson, M., & Gong, S. (2020). Anomaly detection based on machine learning in IoT-based vertical plant wall for indoor climate control. *Build. and Environ.*, 183, 107212. https://doi.org/10.1016/j.buildenv.2020.107212

Long, Y. (2019). Agricultural internet of things system based on cloud computing and machine learning. *Proc. 12th Int. Conf. Intell. Comput. Technol. and Autom., (ICICTA)*, pp. 364–367. Piscataway, NJ: IEEE. https://doi.org/10.1109/ICICTA49267.2019.00084

Ummesalma, M., Subbaiah, R., & Narasegouda, S. (2020). A decade survey on internet of things in agriculture. In *Internet of Things (IoT)* (pp. 351-370). Cham: Springer. https://doi.org/10.1007/978-3-030-37468-6_19

Madushanki, A. A. R., Halgamuge, M. N., Wirasagoda, W. A. H. S., & Syed, A. (2019). Adoption of the internet of things (IoT) in agriculture and smart farming towards urban greening: A review. *Int. J. Adv. Comput. Sci. Appl.*, 10(4), 11–28. https://doi.org/10.14569/ijacsa.2019.0100402

Min, K. T., & Hwang, I.-C. (2021). Tomato farm environment forecasting system using machine learning. *Proc. Int. Conf. on Electron., Inf., and Commun. (ICEIC)*, pp. 1–2. Piscataway, NJ: IEEE. https://doi.org/10.1109/ICEIC51217.2021.9369753

Mochida, K., Koda, S., Inoue, K., Hirayama, T., Tanaka, S., Nishii, R., & Melgani, F. (2018). Computer vision-based phenotyping for improvement of plant productivity: A machine learning perspective. *GigaScience*, 8(1), 1–12. https://doi.org/10.1093/gigascience/giy153

Nagano, S., Moriyuki, S., Wakamori, K., Mineno, H., & Fukuda, H. (2019). Leaf-movement-based growth prediction model using optical flow analysis and machine learning in plant factory. *Front. Plant Sci.*, 10, 227. https://doi.org/10.3389/fpls.2019.00227

O'Grady, M. J., Langton, D., & O'Hare, G. M. P. (2019). Edge computing: A tractable model for smart agriculture? *Artif. Intell. Agric.*, 3, 42–51. https://doi.org/10.1016/j.aiia.2019.12.001

Ou, C., Chen, Y., Huang, T., & Huang, N. (2020). Design and implementation of anomaly condition detection in agricultural IoT platform system. *Proc. Int. Conf. Inf. Netw. (ICOIN)*, pp. 184–189. Piscataway, NJ: IEEE. https://doi.org/10.1109/ICOIN48656.2020.9016618

Paucek, I., Appolloni, E., Pennisi, G., Quaini, S., Gianquinto, G., & Orsini, F. (2020). LED lighting systems for horticulture: Business growth and global distribution. *Sustainability*, 12(18), 7516. https://doi.org/10.3390/su12187516

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, É. (2011). Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12(85), 2825–2830. Retrieved from http://jmlr.org/papers/v12/pedregosa11a.html

Praveen Kumar, J., & Domnic, S. (2019). Image based leaf segmentation and counting in rosette plants. *Inf. Process. Agric.*, 6(2), 233–246. https://doi.org/10.1016/J.INPA.2018.09.005

Ray, P. P. (2018). A survey on internet of things architectures. *J. King Saud Univ. - Comput. and Inf. Sci.*, 30(3), 291-319. https://doi.org/10.1016/j.jksuci.2016.10.003

Revanth. (2019, November). Towards future farming : how artificial intelligence is transforming the agriculture industry. Retrieved October 30, 2020, from https://www.wipro.com/holmes/towards-future-farming-how-artificial-intelligence-is-transforming-the-agriculture-industry/

Ruengittinun, S., Phongsamsuan, S., & Sureeratanakorn, P. (2017). Applied internet of thing for smart hydroponic farming ecosystem (HFE). *10th Int. Conf. Ubi-media Comput. and Workshops (Ubi-Media)*, pp. 1-4. Piscataway, NJ: IEEE. https://doi.org/10.1109/UMEDIA.2017.8074148

Schimmelpfennig, D. (2016, October). Farm profits and adoption of precision agriculture. Economic research report 217. Washington, DC: USDA-ERS. Retrieved October 30, 2020 from www.ers.usda.gov/publications/err-economic-research-report/err217

I2C (2003). Application Note - I2C Bus. Retrieved July 23, 2021 from https://www.nxp.com/docs/en/application-note/AN10216.pdf

Sharma, S., & Jain, R. (2018). Outlier detection in agriculture domain: application and techniques. In V. B. Aggarwal, V. Bhatnagar, & D. K. Mishra (Eds.), *Big Data Analytics* (pp. 283–296). Singapore: Springer. https://doi.org/10.1007/978-981-10-6620-7_28

Shi, W., Pallis, G., & Xu, Z. (2019). Edge computing [scanning the issue]. *Proc. the IEEE*, 107(8), 1474–1481. https://doi.org/10.1109/JPROC.2019.2928287

Sishodia, R. P., Ray, R. L., & Singh, S. K. (2020). Applications of remote sensing in precision agriculture: A review. *Remote Sens.*, 12(19), 3136. https://doi.org/10.3390/rs12193136

Talavera, J. M., Tobón, L. E., Gómez, J. A., Culman, M. A., Aranda, J. M., Parra, D. T., … Garreta, L. E. (2017). Review of IoT applications in agro-industrial and environmental fields. *Comput. Electron. Agric.*, 142, 283–297. https://doi.org/10.1016/j.compag.2017.09.015

Tan, M., & Le, Q. v. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *Proc. 36th Int. Conf. Mach. Learn., ICML*, pp. 10691–10700. JMLR. https://arxiv.org/abs/1905.11946v5

Torres, A., Adriano Filho, J., da Rocha, A., Gondim, R., & de Souza, J. (2017). Outlier detection methods and sensor data fusion for precision agriculture. *Proc. IX Brazilian Symp. Ubiquitous and Pervasive Comput*. https://doi.org/10.5753/sbcup.2017.3316

van Klompenburg, T., Kassahun, A., & Catal, C. (2020). Crop yield prediction using machine learning: A systematic literature review. *Comput. Electron. Agric.*, 177, 105709. Elsevier B.V. https://doi.org/10.1016/j.compag.2020.105709

Vannieuwenborg, F., Verbrugge, S., & Colle, D. (2018). Choosing IoT-connectivity? A guiding methodology based on functional characteristics and economic considerations. *Trans. Emerg. Telecommun. Tech.*, 29(5), e3308. https://doi.org/10.1002/ett.3308

Varghese, B., Wang, N., Barbhuiya, S., Kilpatrick, P., & Nikolopoulos, D. S. (2016). Challenges and opportunities in edge computing. *Proc. 2016 IEEE Int. Conf. Smart Cloud (SmartCloud)*, pp. 20–26. Piscataway, NJ: IEEE. https://doi.org/10.1109/SMARTCLOUD.2016.18

Verdouw, C., Sundmaeker, H., Tekinerdogan, B., Conzon, D., & Montanaro, T. (2019). Architecture framework of IoT-based food and farm systems: A multiple case study. *Comput. Electron. Agric., 165*, 104939. https://doi.org/10.1016/j.compag.2019.104939

Wang, L., Yu, Y., Deng, L., & Pang, H. (2017). A two-stage agriculture environmental anomaly detection method. In *Adv. Comput. Methods in Energy, Power, Electr. Vehicles, and Their*

Integration (pp. 779-789). Singapore: Springer. https://doi.org/10.1007/978-981-10-6364-0_77

Wang, Z., Wang, K., Yang, F., Pan, S., & Han, Y. (2018). Image segmentation of overlapping leaves based on Chan–Vese model and Sobel operator. *Inf. Process. Agric.*, 5(1), 1–10. https://doi.org/10.1016/J.INPA.2017.09.005

Ward, D., Moghadam, P., & Hudson, N. (2018). Deep Leaf Segmentation Using Synthetic Data. Retrieved from https://arxiv.org/abs/1807.10931v3

Yin, X., Wang, L., Jia, W., & Jin, C. (2020). Semi-supervised transformation and deep embedding-based anomaly identification for agricultural internet of things. *IEEE Sensors J.*, https://doi.org/10.1109/JSEN.2020.3047841

Zahniser, S., Taylor, J. E., Hertz, T., & Charlton, D. (2018). Farm labor markets in the United States and Mexico pose challenges for U.S. agriculture. Bulletin 201. Washington, DC: USDA-ERS. Retrieved from https://www.ers.usda.gov/webdocs/publications/90832/eib-201.pdf?v=5940

Zamora-Izquierdo, M. A., Santa, J., Martínez, J. A., Martínez, V., & Skarmeta, A. F. (2019). Smart farming IoT platform based on edge and cloud computing. *BioSyst. Eng.*, 177, 4–17. https://doi.org/10.1016/J.BIOSYSTEMSENG.2018.10.014

PUBLICATION

Guntaka, M. L., Saraswat, D., & Langenhoven, P. (2021). IoT based low-cost testbed for precision indoor farming. ASABE Paper No. 202100617. St. Joseph, MI: ASABE. https://doi.org/10.13031/AIM.2021006