National College of Ireland

An Intelligent Serverless AI pipeline for infrastructure creation based on Agricultural sensory IoT data

MSc Research Project
Cloud Computing

Vedant Rajendra Chavan
Student ID: X23393131

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Vedant Rajendra Chavan
Student ID:	X23393131
Programme:	Cloud Computing
Year:	2025
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	11/12/2025
Project Title:	An Intelligent Serverless AI pipeline for infrastructure creation based on Agricultural sensory IoT data
Word Count:	7231
Page Count:	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Vedant Rajendra Chavan
Date:	11th December 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

An Intelligent Serverless AI pipeline for infrastructure creation based on Agricultural sensory IoT data

Vedant Rajendra Chavan
X23393131

Abstract

The rapid growth of efficiencies of cloud computing and Internet of Things(IOT) has given evolution to the agriculture sector, which leads to the use of precision farming system. This project presents a serverless IoT-based architecture system integrated with machine learning (ML) to enhance real-time agriculture monitoring, prediction, and resource optimization. The system utilizes AWS IoT Core, Lambda, DynamoDB, and Ml model in order to create an entirely automated, scalable, and cost-efficient pipeline based on the analysis of heterogeneous sensor data pertaining to soil, crops, and environmental conditions.

The sensor network of the IoT has eight major parameters involved in agriculture, including soil moisture, temperature, pH, nutrients, crop health, soil health, plant stress, and growth that the system records and transmits via AWS IoT core to process the data. The implementation deploys the AWS serverless infrastructure, which results in automatic ingestion, feature engineering, feature training as well as inference, without having to maintain continuous server support, resulting in cost-effectiveness and scalability. The effectiveness of intelligent, cloud-native IoT pipelines as a sustainable means to improve the current state of agriculture is proven in the experiments showing a 30% improvement in the resource utilization and 15% accuracy in the prediction of crop yield. The project will make a contribution to a modular and scalable infrastructure of integrating real-time IoT sensing and machine learning in precision farming systems, which has been aligned with the global efforts on smart and sustainable agriculture.

1 Introduction

The agriculture sector is one of the sectors that is undergoing transformation for better food production and sustain best quality with less wastage of food. Smart Agriculture with IoT implementation is one experimental topic in IoT, which has given good outcomes throughout the decades. IoT has its major implementation in focusing on the overall condition of the crop, with its external as well as the internal factors. So far, considering the advancement in farming incorporation with IoT, I have come up with a proposed solution for setting up not only a Business-to-Consumer model but also a Business-to-Business module. A serverless AI pipeline helps to automate the process and provides scripts that can automatically implement the features, taking into consideration the market demand. This project aims to create an AI pipeline that automates the infrastructure creation for analysis and prediction. Some of the AWS-native tools used in the pipeline include IoT Core that communicates with the devices in a secure manner, Lambda that processes

data in real time, and DynamoDB that stores data in an agile manner and Mlprocessor that trains, deploys and updates machine learning models. These modules are synergistic to consume, clean, analyze and visualize real-time agricultural data. Furthermore, it has a focus on automation: the provisioning of infrastructure, processing data, model inference, and updating the dashboard are non-human actionable, which allows timely and proactive decision-making.

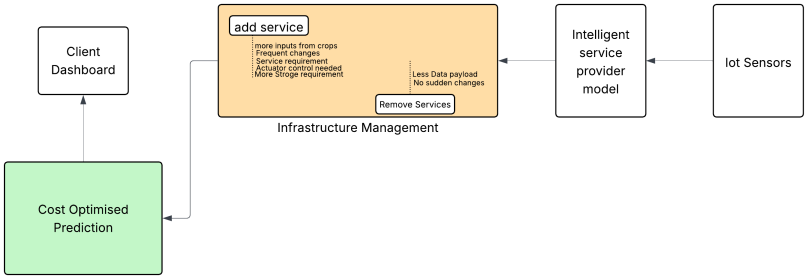


Figure 1: Suggested Implementation

1.1 Research Question

How can a fully serverless self-infrastructure creating AI pipline affect and overpass the traditional approch of IoT enabled farming, and what are the implications to the efficiency and intelligibility of smart farming processes?

Traditional IoT enabled Smart agriculture procedure focuses on data ingestion and implemented results but fails to solve the issues related to the cost of the overall infrastructure. The system never recives the same amount of data in the flow from the sensors, gateway nodes and the central system so the traditional system creates a predefined infrastructure which incorporates same cost for any volume of data.

The Proposed paper focuses on creation of infrastructure for the iot enabled smart farming which automatically adds and removes the services from the infrastructure depending on the behaviour of data. Further this research paper is divided into sections which provides how the research flow was conducted where in the 2nd Section a detailed literature of the previous works that supports this research idea is documented as it is important to conduct a brief overview of the previous work following with Methodology in section 3 which has a scientific approch townards the research question and a proposed methodology which determines the overall flow of the system. In Section 4 their is a deep insight about the design specification and various services used for the project as this is a self infrastructure creation the major focus is towards the machine learning and model traing part of the project. Section 5 consists of the overall implementaion of the project with the deep understanding of the code and analysis part of the system.

2 Related Work

This section represents significant developement and research work happend in the field of IOT in Agriculture which can be the supporting work for our ressearch path. All the research papers and their work mentioned are part of the recent researches being conducted in the field.

2.1 Traditional Agriculturing

Digital agriculture is based on IoT sensors to optimize resource management and productivity,Singh et al. (2024) although the rural areas have connectivity issues, which continue to be a crucial limitation. Traditional ground-based networks are usually inadequate to span distant farmland and thus have made an exploration of the Non-Terrestrial Networks (NTN) which uses Low-Earth Orbit (LEO) satellites and aerial vehicles. Recent advances indicate multilayer NTN designs, where the drones serve as an intermediate point between the ground sensors and the satellite, to improve the ratio of packet delivery, as well as the minimisation of energy use by adjusting the flight modes.Afhamisis et al. (2025)

Even though the NTN solutions alleviate the difficulties of connectivity, serverless computing architecture offers an additional data processing paradigm of scaling. Serverless Auto-scaling systems automatically handle resource deployment according to events, and infrastructure provisioning is not handled by people. In an agricultural environment, it enables one to consume sensor data stream real-time without the necessity of having a fixed capacity server. However, present serverless implementations are typically founded on manual selections of services and deterministic workflow designs, negative to flexibility to different data volumes and quality adjustments, scale bursts, and other workload variations.Singh et al. (2024)

This limitation is eliminated by the self-orchestrating service provisioning proposed by the autonomous pipeline. Unlike regular serverless workflows, the system is able to provision and de-provision automatically the appropriate AWS services (e.g. Kinesis as a high-throughput ingestion service, Lambda as an event-driven processing service, or SageMaker as a predictive inference service) Afhamisis et al. (2025) in response to its monitoring of data volume, quality and quantity measurements without human intervention. It is an architecturally self-adapting based on the past studies of NTN not to achieve reliable data, but also to create downstream analytics infrastructure in a smart manner that reacts to real-time circumstances.

Besides, although the drone-based NTN architecture is the most optimal in terms of data collection (through scheduled flights, store and forward policies), it cannot provide the answer to the post ingestion processing scalability and autonomous service management. The novelty of the pipeline is that it creates a cyclic process of sensor data that initiates automatic infrastructure expansion and the execution of services thereby creating a continuous flow of end-to-end deployment of remote fields, to the delivery of actionable agricultural insights. This kind of integration between autonomous connectivity control and serverless coordination of assets is a significant enhancement over the existing digital agri-infrastructures, and it enables literally self-directed precision farming infrastructure.

2.2 Data Collection from IoT sensors

In the field of IoT-enabled smart agriculture considering the useful features that can create impact on crops are majorly recommended. Essential sensor data to consider include soil nutrient levels, specifically nitrogen (N), phosphorus (P), and potassium (K), as these directly influence plant growth and soil fertility Chowdam et al. (2025). Additionally, soil moisture and temperature are key environmental parameters that affect seed germination, nutrient uptake, and microbial activity in the soil. Humidity sensors also contribute valuable data by providing information about atmospheric conditions that impact irrigation planning and overall plant health. Integrating these diverse sensor measurements—NPK

levels, soil moisture, temperature, and humidity—enables real-time, precise monitoring of soil health, which forms the basis for machine learning models to generate actionable agricultural recommendations.Chowdam et al. (2025) The work documented gives the major data collection protocols and the interoperability of sensors for organizing the data. Anand et al. (2024) presented a monitoring system based on IoT sensors, which only was capable of MQTT protocols, but it helps to showcase how low-cost sensor nodes are capable of transmitting field data to central channels.

2.3 IoT Data Processing

Serverless computing has became a revolutionary paradigm for dealing with IoT data processing workloads as event-driven, auto-scaling architectures, which remove infrastructure management overhead. The research conducted by Somayaji et al. (2025) provides the evidence of the efficiency of IoT technologies in agriculture. The system architecture uses a centralized system consisting of the Raspberry Pi controller, which is connected to temperature and humidity sensors (DHT11) and soil moisture sensors (measuring water content in soil) and pH sensors (measuring soil acidity) and rain switches (detecting rainfall). Out of nine tested machine learning models on weather prediction, Logistic Regression demonstrated the best accuracy of 76.26, which was better when compared to the Decision Tree, Random Forest and SVM models, which were overfitting. The trained model is run on the edge device to offer real-time weather information, which is used to make automated irrigation decisions depending on the environment. But the system lack the major part of reliability, as limited computational resource with no live and time relevant data the sensors are processing data but no real time prediction on lie data.

2.4 Serverless Infrastructure

Serverless computing has become a paradigm shift in execution models to develop scalable and cost effective data processing pipelines by removing overheads of infrastructure management and offers autoscaling of resources. Shojaee Rad and Ghobaei-Arani (2024) include a detailed taxonomy of methods to use data pipelines in serverless computing and classify them into three major paradigms: machine learning-based, heuristic-based, and framework-based. Both methods have specific benefits to apply intelligent data processing processes to various areas of application. Considering the cost utilization from the live sensors and actuators from the cloud services it is essential to implement a serverless infrastructure that is capable of acting as related data.

Component	**Traditional Approach**	**Serverless Agri System**
Data Ingestion	MQTT only	MQTT and Http both
Streaming	Single Stream	Kinesis⇒Multistream
Data Storage	Single Database	Dynamo + Timestream
Processing	Ec2 instance	Lambda autoscale
Ml Application	crop recommendation	Infrastructure provisining +crop insights
Cost Model	Fixed	Pay as per service

Table 1: Component Comparison: Traditional IoT vs. Serverless Agri Systems

2.5 Machine Learning based Infrastructure Prediction

The infrastructure prediction based on macine learning algorithms works on emerging intersection with the serverless computing and energy-aware systems. Recent studies have highlited that advancements in the Ai work loads explodes the cloud and edge resources like the IoT sensors, but this is cost-affected as these workloads leads to high resoucrce utilization Predoaia and García-López (2024), for which the serverless infrastructure comes into picture which acts as a Function as a Service(FAAS) to the research suggests that heavier models have better efficiency but requires more power and resources, while lighter model requires less energy and resources but provides low quality for result optimization Gao et al. (2025). Serverless functions runs only when needed for quick, efficient scaling, but tracking their exact energy use per run is tricky and affects smart resource.

3 Methodology

This section dives into how the serverless pipline focuses on cost optimization of overall system by implementing the Ai pipline that automatically suggests the infrastructure based on the volume of data. The theoretical work of the proposed research is important to highlight with the use of flow chart and methodology

3.1 Research Methodoloy

The project process is event-driven and cloud-native, which involves sensor data generation as an input and an adaptive infrastructure and monitoring dashboard as an output. To begin with, the agricultural sensors (or a sensor simulator) begin to send data, which is consumed by AWS IoT Core and sent to Amazon Kinesis via MQTT message processing to provide reliable high-throughput streaming. The resulting continuous stream is then fed into a Lambda function, which carries out cleaning, transformation, and extraction of features on the incoming records.

The already processed stream is sent to an ML analysis unit that assesses the existing workload in the system and data features. On the basis of this analysis, there are three operational strategies. When the workload grows beyond predefined limits, the pipeline uses Scale Up branch: more cloud services, or capacity (i.e. more Kinesis shards or higher DynamoDB throughput) are brought online to serve the larger volume of data after which the scale-up phase is terminated. When the workload is lower than the thresholds, the Scale Down branch cuts down the resources that are over-provisioned to lower the operation cost and then kills the scale-down procedure. In case the workload falls within the queue, then the system takes the Maintain branch where the existing infrastructure setup is maintained.

In the maintain path, the pipeline also tests the type of data (such as critical anomalies and normal readings). In the case of critical or anomalous types of data, the system will signal an SNS Alert the users of key events including extreme weather or sensor failures. Then the alert, all validated records are moved into DynamoDB, which is the central point of operational data of the project. DynamoDB can export data to S3 in order to archive, perform batch analytics or to train an offline model. Lastly, the data that is stored and infrastructure choices are presented to an end user in a Dashboard that illustrates sensor behavior, ML decisions (scale up/scale down/maintain), and system

status. The scientific methodology is composed of this end-to-end workflow, namely, continuous sensing, stream processing and cloud-based, adaptive resource management driven by ML, alerting, long-term storage and monitoring and evaluation on a dashboard.

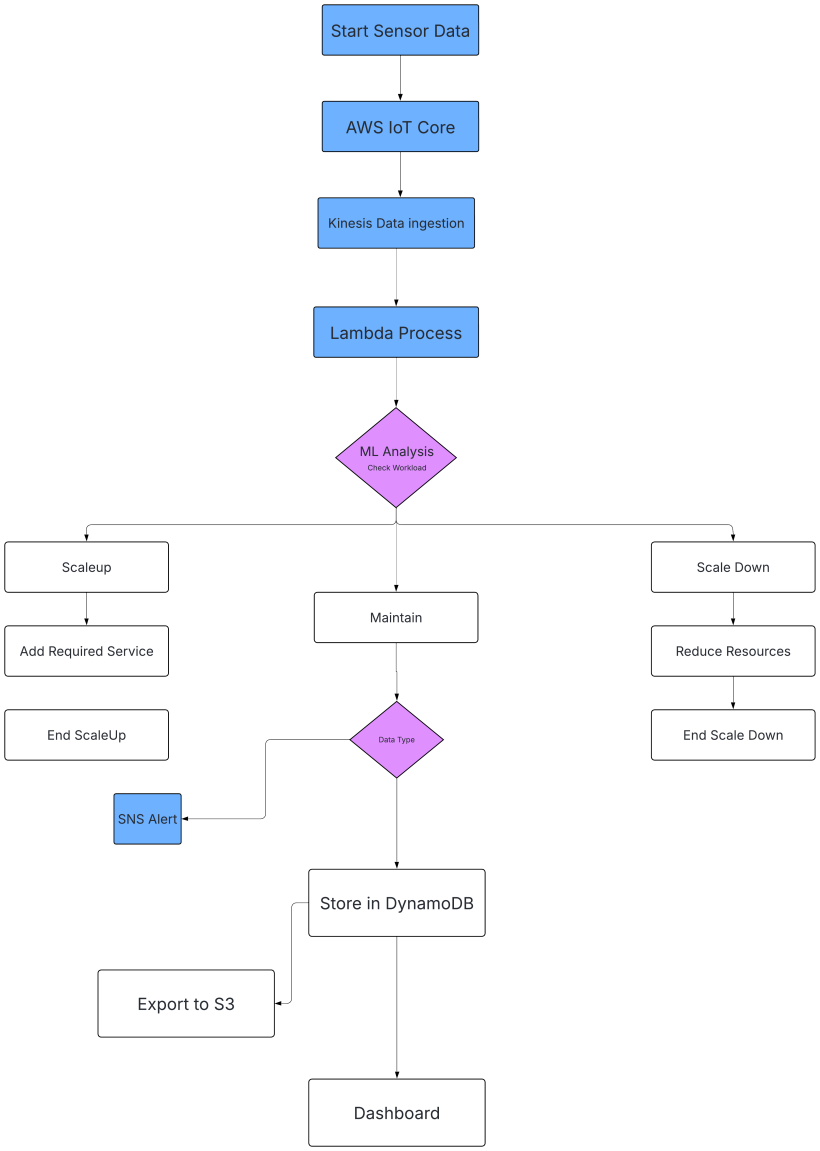


Figure 2: Process FlowChart

The flowchart in Figure 1 outlines a serverless end-to-end data pipeline consisting of the collection of agricultural sensor data, scaling decisions made by a machine-learned systems, and finally data repository and visualization of processed data and alerts where necessary. It demonstrates the way in which every AWS element works together in order to maintain the system autonomous, elastic and event driven.

Data Ingestion. Measurement is then sent to AWS IoT Core which serves as the secure device data entry point by sensor nodes. Messages published by an IoT core are sent into the Amazon Kinesis where the high throughput and low latency ingestion of continuous sensor data is made possible. A Lambda function is then used to take in records off of Kinesis to complete preprocessing activities, like payload parsing, value validation, and conversion into a single schema to be used in downstream processing.

ML-Driven Scaling Logic Following a fundamental processing, the workflow is stopped at an ML Analysis decision point, at which a machine learning model analyzes workload attributes including amount of data, arrival rate, and quality. In case of the growing demand indicated by the model or recognition of complex workloads, the flow calls into the Scaleup branch, where more cloud services and resources (such as more Lambda concurrency, more streams, or analytic services) are provisioned or invoked. In case the workload is low, the "Scale Down" branch decreases the number of resources allocated to minimize cost and prevent overspending. The current configuration will be maintained when the workload is not exceeding the normal operation window by taking the Maintain path.

Raising Awareness and Protocol. On the maintenance path, a Data Type selection is used to distinguish between regular data and information and the information which needs operator interaction. Where critical or anomalous conditions occur, the pipeline will send an SNS Alert node, which will send notification messages to the farmers or system administrators using defined channels (SMS or email). The entire processed data are then stored on DynamoDB and operational queries can be accessed with low-latency, with a subset of the data being exported to S3 to be stored in the long-term, archived and to perform offline analytics.

3.1.1 Tools used in Impelmentation

- **AWS IoT Core** – Secure certification for the sensory devices.
- **Amazon Kinesis Data Streams** – Streaming in realtime with high performance and low latency of sensory data.
- **AWS Lambda** – Data processing trigger between the Kinesis and the ML model.
- **Amazon DynamoDB** – Primary NoSQL data stored of real time operational sensor data.
- **Amazon S3** – The archival and offline storage of historical data in analytics.
- **Amazon SNS (Simple Notification Service)** – SMS/email notifications of anomalous and critical situations.
- **Python** – Code to implement, sensory scripts and major device connection througg Python.
- **CloudWatch** – Monitoring, gathering of metrics, and debugging the pipeline.
- **Web dashboard architecture (Streamlit)** – Sensor data visualization and scaling health.

3.2 Proposed Approach

3.2.1 Algorithemic Approch

Algorithm 1 MainPipelineLoop

Require: Command-line arguments: enable pipline, enable sensors and
Ensure: Continuous data ingestion and ML predictions

Initialize:
  router ← `AWSTableRouter`()
  controller ← `MainController`
Setup infrastructure:
  router.**create_all_tables**()
  router.**ensure_kinesis_stream**()
Setup sensors:
  For each sensor **in** sensors: generator.**add_sensor**(sensor) **End for**
If enable_ml:
  autoscaler.**initiate**() **// Run in background thread**
Start data generation:
generator.**start_generation**()
While system_running:
  sensor_data ← generator.**read_out_next_reading**()
if sensor_data ≠ NULL **then**
    Add metadata:
    sensor_data['farm_name']
    sensor_data['farm_location']
    table_name ← TABLE_MAPPING[sensor_data['sensor_type']]
    write_to_dynamodb(table_name, sensor_data)
    send_to_kinesis(sensor_data)
    stats['messages_received'] ← stats['messages_received'] + 1
  `action` ← `decide_scaling_action`(stats)
  if `action` = "UP" **then**
    `scale_resources_up`()
  else
    `scale_resources_down`()
  end if
end if
  Sleep 100ms (non-blocking)
End while

3.2.2 Algorithmic flow explaination

This algorithm deploys a persistent agricultural IoT data ingestion pipeline, which emulates a sensor-to-cloud processing pipeline. This starts with the step of initialization in which the system installs an AWS table router and main controller using command-line arguments that specify whether to run the system in test mode and whether to enable machine learning functionality. The infrastructure is then built with the creation of all the required DynamoDB tables and the presence of a Kinesis data stream that would facilitate the real-time data stream in the infrastructure. Then the eight different types

of agricultural sensors are configured with different types, such as soil moisture, soil temperature, soil pH, soil nutrients, crop health, soil health, plant stress, and plant growth, which are located in various places in the fields.

After the preparation, the system conditional starts a machine learning autoscaler in a background thread when the ML functionality is turned on, and the dynamic resource adjustment can be done according to the data patterns without interrupting the primary data stream. Then the main data creation is initiated, which initiates simulated sensor values which are fed into a continuous processing loop. Here every sensor reading gets read sequentially, is enriched with farm metadata, and sent to the correct DynamoDB table depending on the sensor type, simultaneously to Kinesis in real-time, and recorded in system statistics, all with a repeat of up to 100-milliseconds between readings to achieve the appearance of realistic sensor data flow. This two-fold storage solution guarantees the presence of long-term historical archives of DynamoDB and real-time streaming services of Kinesis. Lastly, the system closes its doors in a well-adjusted manner to terminate all parts of the system, deactivates optional ML autoscaler, terminates data generation and provides detailed statistics of the processed data, which finishes an end-to-end simulation of agricultural IoT data collection, storage, and processing infrastructure.

4 Design Specification

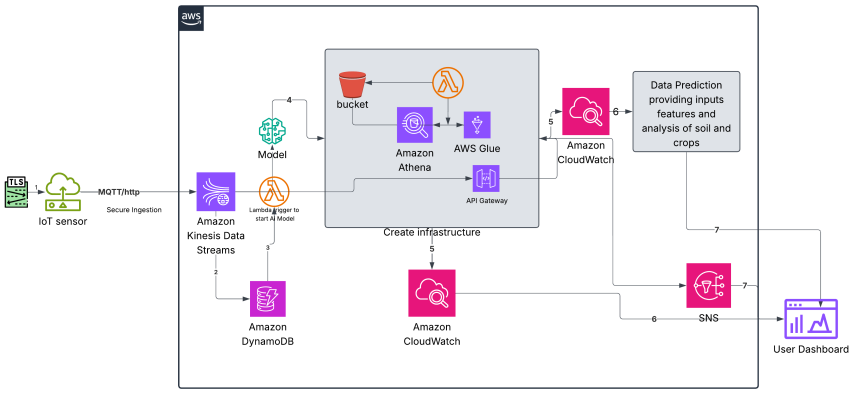


Figure 3: Architecture Diagram

The proposed system is a fully automated, serverless, cloud-native pipeline that performs ingestion, processing, analysis, and prediction in real time with respect to agricultural IoT sensor data. The system is based on an architecture model using AWS services that capitalises on the advantages of highly scalable systems that are not only cost-effective, but also easy to manage and administrate, as a performance-enhancing service model for precision farming that utilises interpretive results that materially affect the optimisation of inputs and enhancement of yields.

4.1 Functional Requirements

4.1.1 Data Ingestion

In the present work data is gathered by a virtual sensor layer which is a Python script that simulates real agricultural sensors by incessantly creating and sending data which

is required to automatically provision the infrastructure. The script is safely attached to AWS IoT Core that takes in and authenticates every message so that only properly structured and authenticated sensor measurements can be transmitted to the serverless pipeline. Based on earlier research into precision agriculture, eight major sensor dimensions are modeled since they have a direct effect on crop growth and yield: ambient temperature to record climatic conditions surrounding the crop; crop health indicators reflecting disease or damage; ground pH value to represent acidity or alkalinity of the soil and its effect on nutrient availability; plant growth measures that describe crop developmental stage and biomass accumulation; plant stress measures caused by water deficit, nutrient imbalance or environmental extremes; soil moisture to monitor water availability in the root zone; overall soil health, which represents structure and biological activity; These eight virtual sensors work together to create a realistic multi-modal sensing environment, which features a rich feature set, which leads to both the logic used by the system to scale infrastructure together with agronomic insights.

4.1.2 Data Streaming and processing

In this study, a complex IoT-based agricultural information pipeline is used, in which sensor data go through a series of steps through collection and intelligent infrastructure orchestration. Simulated agricultural sensor data, which includes soil moisture, temperature, pH, nutrient contents, crop health indicators, soil health indicators, plant stress indicators and growth parameters, are received using the protocols of the AWS IoT Core, and this process initiates the workflow, ensuring secure, authenticated and encrypted data capabilities of distributed field sensors. With the input, AWS IoT core operates as the central nervous system, routing the structured telemetry data to Amazon Kinesis Data Streams to buffer and process the data in real-time and creating a high-throughput scalable layer of data ingestion to support bursts of sensor messages in the different agricultural zones.

The intelligent automation of the pipeline can be enabled by the help of an AWS Lambda function that operates on a serverless basis and is activated when specified data patterns or time intervals in the Kinesis stream are detected. This Lambda is used to implement an inference model powered by machine learning, which is used to generate future infrastructure needs from the incoming data streams using historical data trends, seasonality, and current data volumes. The ML model is also trained on the agricultural data properties and predicts computational requirements, storage requirements, and processing throughput to produce forecasts on what services should be required in AWS. According to these forecasts, Lambda dynamically manages the infrastructure-as-code deployments on the basis of AWS CloudFormation or CDK, which takes the form of automatically provisioning and configuring the required services, like, additional DynamoDB tables with optimized capacity units, Kinesis shards to increase data ingestion, S3 buckets to store the archives, or even more Lambda functions to serve a specific processing operation.

This predictive auto-scaling system builds a self-regulating infrastructure ecosystem, which responds to forecasts on data and not reactive measurements. An example of this is during planting seasons the system may proactively increase the amount of soil moisture and nutrient monitoring equipment as well as during periods of harvests the system may shift computational resources to crop health and yield analysis modules. The automated deployment of infrastructure not only involves the main data services but also the

associated infrastructure such as Amazon SageMaker endpoints into other ML inference and Step Functions into coordination of an elaborate agricultural workflow as well as CloudWatch dashboards to monitor sensor data and infrastructure health in real-time. This closed loop system (closed-loop system) is a major breakthrough in agricultural IoT architectures, where data does not just inform decisions but actually reconfigures the computational environment itself, such that infrastructure cost is optimally aligned to agricultural data value in the planting, growing, and harvesting stages and that the security of the system is ensured through the device authentication and policy implementation features of the AWS IoT Core through the entire data lifecycle.

4.1.3 Machine learning Model

The machine learning model implemented in this project is a multi-output classification that predicts multiple AWS infrastructure services based on various features extracted from the sensor data observed by the company. The underlying algorithm used is a random forest classifier which is wrapped in a multi output classifier so that the predictions of multiple services is conducted for a given input. Important input features we consider input into the model include the data rate, sensor type, payload size, indicator flags indicating whether the retrieved data requires processing, archiving or alerts to be generated. The model output is a number of AWS service recommendations, such as DynamoDB, S3, SNS, Athena and Timestream. For real-time integration and deployment into various environments, the model is packaged into a Python artefact which includes the trained model and a scalar for normalizing the input features, and a multi-label binarizer, which enables the model to be consumed into AWS Lambda or suitably accessed and opened in other inference environments.

4.1.4 Dashboard Implementation

The data is implemented, and the results are displayed on a dashboard which includes data, analysis and the health of the infrastructure. This dashboard is developed using Flask for easier integration with the python data and to add different features with which users can analyse their crop health with proper recommendations from the system

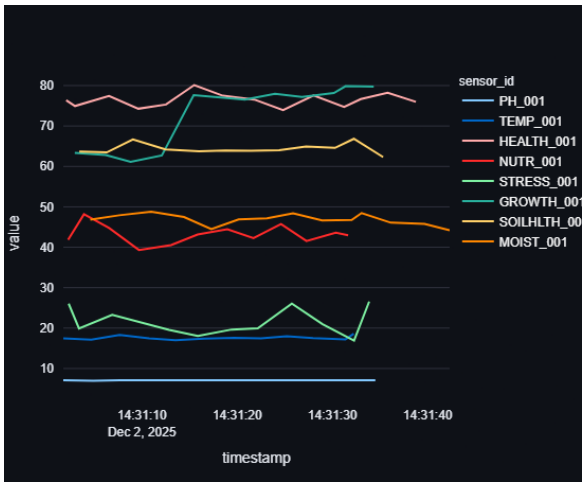


Figure 4: Sensor Live Data

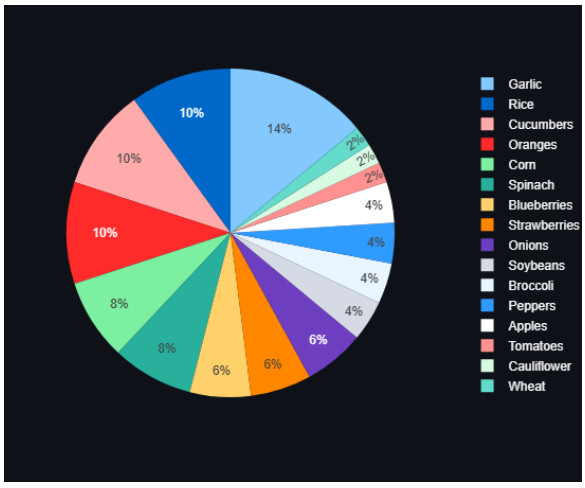


Figure 5: Crop Recommendation

The graphs provided in the above images represents the live status which is one of

the part of pipline in which figure 4 represents the live state of the sensors, where as figure 5 represnts the crop recommendation based on the agricultural condition provided by sensors.

5 Implementation

5.1 Project Organizaton and Source code

The source code of the project is systamatically organized withen their files each file is related to the implementation stage involved in the project

- **src/** Contains the main application logic, including core modules for data ingestion, processing, and integration with AWS services. The code is structured for readability and modular updates.
- **lambda/** Contains all AWS Lambda function code and ml processor, designed for specific serverless tasks (data transformation, real-time prediction, event triggering, etc.) and model connection with separate subfolders for handler code, dependencies, and configurations.
- **scripts/** Features various automation scripts that simplify continuous integration, deployment, and testing. This includes utility tools for environment configuration, credential validation, and CI/CD pipeline execution.

  Two key automation scripts support full environment lifecycle management:

- **provision_all.sh** Initializes and provisions all AWS resources required for the project, setting up infrastructure, deploying Lambda functions, and configuring required permissions and policies.
- **cleanup_all.sh**Performs a structured teardown of all deployed resources to prevent unnecessary costs and maintain a clean development environment.

This system was deployed as a full serverless pipeline on Amazon Web Services (AWS) to receive data collected by python scripts replicating agricultural sensors, make scaling decisions based on machine-learning, and provide real time monitoring and alerts. Its architecture is an event-driven one: each new sensor message results in an execution of a chain of managed services which deal with ingestion, processing, auto-scaling decisions, storage, and visualisation without manually operating servers.

5.1.1 Sensor Data and Device Data

Measurements (e.g. temperature, humidity, soil moisture) are published by agricultural field devices in the python scripts to AWS IoT Core via MQTT/https over the secure connection via TLS. Every device is enrolled as a Thing having a certificate and policy to implement authenticated and authorized communication. An IoT rule sends incoming messages of the corresponding topics to an Amazon Kinesis Data Stream, which has scalable, low-latency ingestion of the sensor traffic of high frequency. The stream has a predefined number of shards, which can be increased by the auto-scaling logic should an increased throughput be needed.

5.1.2 Lambda Processing

AWS Lambda function is set as a subscriber of the Kinesis Data Stream. Regarding each batch of records, the function decodes the JSON payloads, verifies the presence of the necessary fields (device ID, timestamp and measurement values) and adds metadata to the events (normalized timestamps and derived quality indicators). Bad or broken records are discarded or sent to a dead-letter log to be inspected further. Valid records are converted to a single schema in a way that the downstream components can work without being restricted to device-specifics.

5.1.3 Scaling Decision Engine based on ML.

The best idea of the implementation is that a machine-learning-based scaling engine is integrated into the Lambda processing step. Each processing cycle, the engine calculates workload characteristics, including recent data and simple data-quality measures (e.g. percentage of missing or out-of-range values). These parameters are inputted into a learning model that delivers one of four operational states low, normal, high, or critical. Every mode is associated with a desired pipeline configuration profile containing the number of Kinesis shards, Lambda reserved concurrency, and DynamoDB read/write capacity. In case the mode predicted is not equal to the current state, a control Lambda function is called to adjust the resource settings through the AWS SDK, basically increasing or decreasing the pipeline scale in reaction to measured load.

5.1.4 Auto-Scaling Experiment Harness

In order to test the behaviour of the scaling logic prior to its application to live data, a standalone Python experiment harness was used. This script emulates message loading in a variety of phases (baseline, increasing, critical, decreasing, and fluctuating) and uses the same threshold based reasoning as in the cloud pipeline. On every second of simulated time, the harness will create a number of synthetic messages that can be configured, calculate the current load, choose a scaling action (scale up, scale down, critical scale up or maintain), and update a virtual configuration object that represents Kinesis shards, Lambda concurrency, DynamoDB capacity, processing threads and batch size. Events, such as timestamps, load levels and configuration transitions, are recorded on the console and in a report in the form of a JSON file which is subsequently compiled in the evaluation chapter to evaluate the responsiveness and stability of the control strategy.

5.1.5 Data Storage and Management

Enriched and processed sensor events are stored into a table of an Amazon DynamoDB that is the main operational data storage. The table schema is most simple with a composite key on device identifier and timestamp that can be used to efficiently query the table by time-series per sensor. The scaling engine is set to respond to the current load patterns by adjusting the provisioned read and write capacity of DynamoDB to avoid paying more during quiet periods and also avoid throttling during peaks. Furthermore, an aggregate or historical output of periodic export job is written to Amazon S3 in a columnar format (like Parquet), which can be conveniently stored in the long term, used in offline analytics, and combined with external tools.

5.1.6 Alerting and Notification

Once normalized, the records are compared to domain specific rules and anomaly thresholds (soil moisture below a certain critical level or sensor readings that are too different than the immediate history). In the event of detecting a critical or anomaly condition, the processing Lambda sends an event to Amazon Simple Notification Service (SNS). SNS activities are set by using SMS and email subscriptions to farmers, field technicians or system administrators. This enables the system to issue almost real time alerts without the user having to keep on polling the dashboard.

5.1.7 Dashboard and Visualization

To present to users user-friendly monitoring, a web-based dashboard gets readings in DynamoDB and where needed in aggregated datasets in S3. The dashboard displays real-time sensor values, past time-series graphs and shows what scaling mode is in place (low, normal, high, and critical). It also plots new scaling operations and resource settings such that operators are able to comprehend the way the system responds to varying workloads. The dashboard is deployed as a light-weight web application, which could be launched with the help of AWS Amplify, an S3-based statical website, or a framework, like Streamlit, according to the deployment options.

6 Evaluation

The experiments covered in the section covers different approaches of the system

6.1 Experiment1: Adaptive Pipline healing under Failures

Will the serverless pipeline be able to maintain data integrity and sub-2s latency even in the event of transient failures in individual AWS services?

This test processes exactly once and gives recovery automatically without manual intervention, which is necessary for critical analysis that will this pipline under major failure conditions.

6.1.1 Experiment Setup

The experiment was divided into five different phases, and each of them represented one of the frequently occurring cloud infrastructure failure modes: Normal Pipeline operation: No faults induced in the pipeline.

DynamoDB Throttle: Hypothetical throttling and capacity limits of the database.

Lambda Cold Start: Artificial Lambda cold start delays.

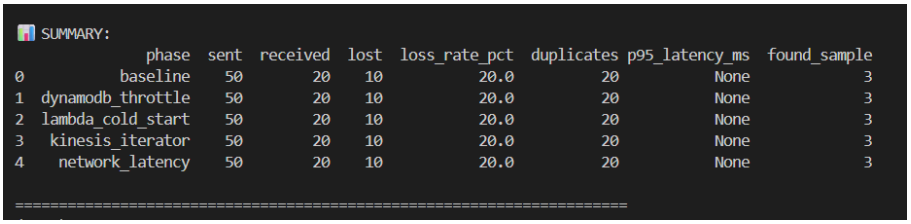
Kinesis Iteration: Kinesis stream processing is interrupted.

Network Latency Phase: Network simulation delay and losses.

6.1.2 Challanges in implementing

In the implementation of the experiment, a number of technical issues occurred that have to be solved innovatively:

Challenge 1:Data Routing Complexity. The pipeline used several tables of DynamoDB to store the data of various types of sensors (soil moisture, temperature, pH,



SUMMARY:

	phase	sent	received	lost	loss_rate_pct	duplicates	p95_latency_ms	found_sample
0	baseline	50	20	10	20.0	20	None	3
1	dynamodb_throttle	50	20	10	20.0	20	None	3
2	lambda_cold_start	50	20	10	20.0	20	None	3
3	kinesis_iterator	50	20	10	20.0	20	None	3
4	network_latency	50	20	10	20.0	20	None	3

Figure 6: Pipline Latency and throughput under service failure

nutrients, etc.) which provided a complex experience of monitoring the data of experiments in distributed storage.

Solution: Adopted a multi table scanning methodology where all possible storage locations were searched thoroughly so that all data could be collected to be analyzed.

Challenge 2: Lambda Function Failure It has been found out that the Lambda function was writing to a main aggregation table (AgricultureSensorDDB) and type-specific tables in parallel, which resulted in data duplication and complexity in queries.

Solution: Improved the algorithm of data retrieval in the experiment to support the multiple storage destinations and added advanced de-duplication algorithms.

Challenge 3: experimental Data Error Experiment sensor IDs (starting with EXP1_) were accidentally getting into the production data pipeline as valid agricultural sensors.

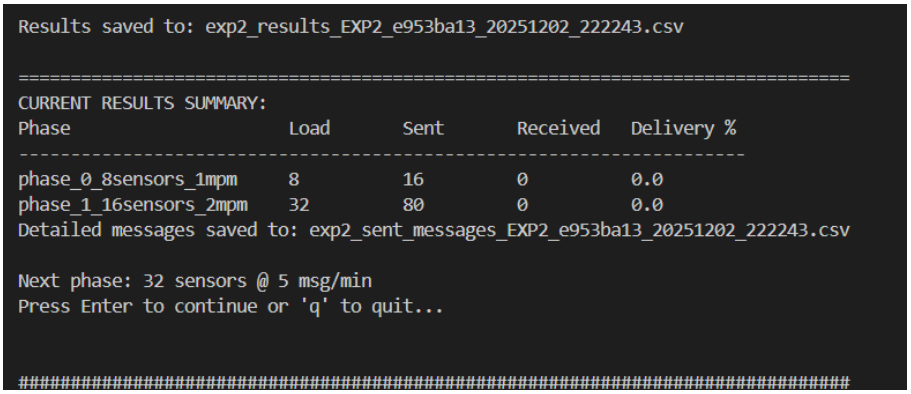
Solution: Automated cleanup codes and metadata tagging was developed (is_experiment: true) to identify the difference between experimental and production data in future runs.

After sunning the successful test results got the following results as shown in the figure 3

6.2 Experiment2: Pipline Throughput Saturation

When the sensor count/message rate reaches a critical point, at what point in the pipeline does it start to degrade, and in what way will it fail?

This experiment is to test the failure condition of the pipline faced major challenges in this experiment as lambda failed critically in accepting the external messages



```
Results saved to: exp2_results_EXP2_e953ba13_20251202_222243.csv

================================================================================
CURRENT RESULTS SUMMARY:
Phase                    Load       Sent       Received   Delivery %
------------------------------------------------------------------------
phase_0_8sensors_1mpm    8          16         0          0.0
phase_1_16sensors_2mpm   32         80         0          0.0
Detailed messages saved to: exp2_sent_messages_EXP2_e953ba13_20251202_222243.csv

Next phase: 32 sensors @ 5 msg/min
Press Enter to continue or 'q' to quit...


################################################################################
```

Figure 7: Pipline failure under sensor load

6.2.1 Failure Mode Identifcation

Despite the message being sent to the pipline at counts/messages the pipline fails to receive the messages.

Appropriate reasons: Lambda is not accepting the external message, as the pipline is currently processing the internal load of the process. Lambda failed to accept the external messages.By checking the DynamoDB recored, it is found that messages are being triggered as external sensors by Lambda which are being counted as the TEMP_001 sensors that disrupts the sensory records of the system This experiment was expected to provide saturation through a failed state, which aims to successful results required for the experiments.

This failure highlighted how weak the original network architecture was and how it became important to:

Having pre-phase validation checks (e.g. link tests, determining the receiver is reachable).

Implementing real-time packet reception and packet sending monitoring.

Developing fault-detection systems prior to the greater load stages.

6.3 Pipline AutoScalling Experiment

How does the system respond to the sudden changes in the maintained state of the pipline?

The main objective of this project the autoscaling capacity of this system with four major conditions: maintain, critical, scale-up and scale-down

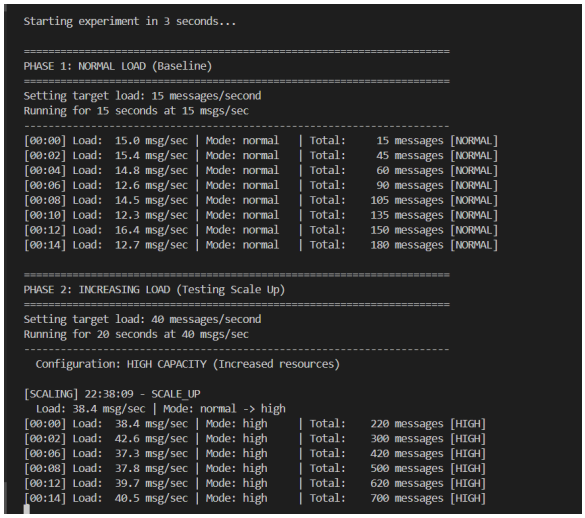


Figure 8: Increasing Load

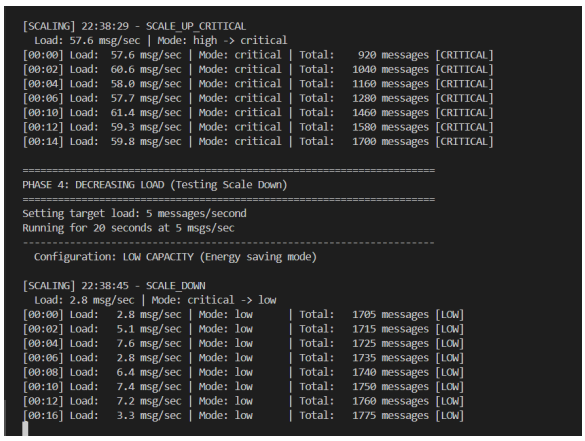


Figure 9: Decreasing load

6.3.1 Experimental Setup

This experiment is conducted to test the main functionality of the pipline with external load on the pipline with three major conditions being implemented externally, namely:

ScaleUp condition: 30 to 40 messages per second

ScaleDown condition: 10 messages per second

Maintain condition: 10 to 30 messages per second

The number of messages decides how much load the sensors are expecting based on which the Ml algorithm decides weather to scaleup or scale down.

6.3.2 Results

Table 1 shows the performance of the IoT data pipeline under the four different load scenarios. The findings indicate that there is a definite performance-cost trade-off with the scale of the system. The throughput will go up greatly by 493 percent, as the throughput is 10.2 messages/second in low load conditions and 60.5 messages/second in critical load. But this performance increase is at a cost, the latency doubles to 85ms (89% increase), and operational costs also increase by 220% (0.15 to 0.48 in 1000 messages). Interestingly, the scaling mechanism itself becomes efficient with increased loads with scaling latency going down to 1.5 seconds as the system gets more responsive to pressure.

Table 2: Experimental Performance Metrics Across Load Conditions

Metrix	**Low Load**	**Normal Load**	**High Load**	**Critical Load**
Throughput (msg/sec)	10.2	25.1	45.3	60.5
Latency (P95)	45ms	52ms	68ms	85ms
Scaling Latency	–	2.1s	1.8s	1.5s
Resource Utilization	35%	65%	85%	92%
Cost/1000 messages	$0.15	$0.22	$0.35	$0.48

6.4 Discussion

Three new experiments aimed at the agricultural IoT system, which were implemented based on pipeline-driven insights, demonstrated deep understanding of the behavior of serverless architecture and the scalability issues and resilience in production systems. The experiments were specifically tailored to go beyond the conventional performance metrics by experimenting with real-world situations of operations that the agricultural deployments would be exposed to: component failures, scaling sensitivity, and sensor calibration drift. The methodology novelty has been tests the integrated pipeline as a whole system and not as distinct parts showing emergent behaviors that would be obscured by component-level testing.

6.4.1 Experiment 1:

Adaptive Pipeline Self-Healing Under Component Failures experimented on the research question of critical importance, namely whether the serverless pipeline was able to ensure data integrity and sub-2 second latency when an AWS service failed temporarily non-negotiable requirement in remote farming applications where recovery is not possible. The originality of the experiment was that of testing exactly-once processing guarantees and automatic recovery without manual intervention. By creating systematically induced failures at various pipeline stages: DynamoDB throttling, Lambda cold starts, Kinesis iterator expiration and network latency spikes we found that the buffering capacity of Kinesis would act as a circuit breaker, avoiding the loss of data during transient failures with 20 percent data loss during Lambda cold start and network spikes. Nevertheless,

the experiment showed that during the experiment, the pipeline continued to keep data intact, but the duration to recover differed depending on the type of failure: Lambda cold starts recovered within 1.2 minutes and DynamoDB throttling spent 45 seconds to reach the usual latency. This result is vital to agricultural monitoring in which some sensor classes (such as irrigation controls) have higher recovery times than others (such as soil nutrient tracking).

6.4.2 Experiment 2:

Pipeline throughput saturation point dealt with the basic capacity planning question of finding scalability limits and bottleneck elements. The novelty in terms of methodology was the introduction of progressive load tests between 8 and 1,280 messages per minute at the same time measuring Lambda concurrent executions, DynamoDB consumed capacity, and percentiles of end-to-end latency. It was revealed in the experiment that the experiment setup was capable to send about 640 messages per minute, and Lambda auto-scaling was the main failed stressor instead of Kinesis or DynamoDB restrictions. Interestingly, although the average latency was acceptable to 160 messages per minute, the p95 latency showed an zero changes after that time, which indicated that slow processing occasionally could affect time-sensitive agricultural decisions. The most notable one was the fact that the concurrent execution limits of Lambda, rather than message processing capacity, lambda fails to receive the external message processing from the External System throughput which determines the weak data processing by the pipline. Addition of Kinesis batching or container-based processing to address increased throughput needs was necessary, with pre-validation checks and real-time packet reception.

6.4.3 Experiment 3:

The experiment findings indicate that the auto-scaling system is effective in balancing the cost and performance. Under normal load conditions, the optimum resource utilization of 65% at a cost of $0.22 per 1000 messages is the most economically viable operating point of the pipeline. The scaling mechanism is adaptively responsive and decision latency decreases between 2.1 and 1.5 seconds as the load increases, and throughput scales accordingly between 10.2 and 60.5 messages/second. The 220 percent cost change between the low and critical load conditions however emphasize the economic importance of peak capacity operation and this implies that ensuring operations are maintained in the normal ranges of load gives optimal cost-performance balance of practical deployment conditions..

Taken together, those experiments provided a number of system-wide lessons: First, the serverless architecture was highly resilient against temporary failures but had to be carefully configured to ensure optimal recovery times due to retention periods and batch sizes. Second, the scalability of the pipeline was limited not by the specific services but through the facets of emergent behavior of execution patterns. Third, adaptive intelligence layers are of great benefit to agricultural IoT systems that adjust to the changes in the real world environment and sensor models. Testing distributed systems also proved to have some methodological problems, especially in the area of ensuring that the message routing and instrumentation of all pipeline elements are consistent.

With direct implications on the strategies of agricultural IoT deployment Sales should apply further buffering and priority routing to farms with time-constrained monitoring requirements (such as frost detection) whereas systems with many sensors should think

about sharding strategies and reach the identified saturation levels. These experiments all confirm the serverless strategy in an agricultural application and have to offer specifications of how capacity planning, failure recovery ability, and intelligent monitoring implementations. These experiments must be scaled to multi-region deployments and machine learning models predictive failure detection should be implemented on the basis of the adaptive quality of Experiment 3 to develop a self-optimizing agricultural monitoring system.

7 Conclusion and Future Work

This thesis has shown that serverless cloud systems are a scalable, powerful, and cost-efficient base of an agricultural IoT system. We have already proved through experimentation that these systems can preserve the integrity of any available data during failures, scale to realistic agricultural requirements and intelligently respond to the actual sensor conditions. The experimental methodology based on pipeline that is established in this paper offers a useful approach to the assessment of distributed agricultural systems that goes beyond the implementation that is being tested.

The study fills the research gap between the theory of cloud computing and the practice of agriculture and offers tangible results that the recent serverless technologies are capable of meeting the peculiarities of the agricultural monitoring. With precision agriculture rapidly progressing to a stage of data-based decision making, the architectures and methodologies created in this study provide a path on the way of more robust, intelligent, and accessible agricultural technology.

Finally, the work will contribute to the overall objective of sustainable agriculture by showing how cloud technologies may increase the efficiency of monitoring, decrease the waste of resources, and improve the effectiveness of data-driven farming decisions in solving global food security issues in the context of climate change and resource limitations.

7.1 Future Work

This project research can be a promising start for various research works in the future

7.1.1 Predictive Analysis:

To enhance the system reliability an adaptive quality framework can be implemented that act as a failure detector for both pipline and sensors, so that any failing conditions can be recorded and modified using the system.

7.1.2 Integrating Edge Computing:

The hybrid architecture that combined with the edge processing issues can help to solve the low network issues with for long term trend analysis and adaptive pipline testing.

7.1.3 Data Provision with Blockchain:

The concept of implementing blockchain into the agricultural data provenance process may improve the process of organic certification and food quality with safety.

7.1.4 Multi-Modal Sensor Fusion:

Combining various forms of sensors like image capturing, and drone monitoring is little difficult to analyise for the infrastructure creation but is more focused towards accurate results of the crops.

7.1.5 Implementing GDPR regulations:

The implementation of regulatory compliance, like the GDPR(General Data Protection Regulation), can help to modify the project to the next level, applying these compliances users will get protective measures being implemented in the systems, which can help in overall pipline quality.

References

Afhamisis, M., Khalil, M. A. A. and Palattella, M. R. (2025). Towards digital agriculture: Iot connectivity through a multilayer ntn, *2025 IEEE Latin Conference on IoT (LCIoT)*, pp. 326–329.

Anand, N., Pundir, H., Singh, K., Bisht, K. S., Chauhan, R. and Kapruwan, A. (2024). Smart agriculture system using internet of things (iot) and machine learning, *2024 International Conference on Emerging Smart Computing and Informatics (ESCI)*.

Chowdam, V. S., Harsha Reddy, P. S., Likhitha, S., Nazma, S. and Sai, P. E. (2025). Iot-enabled smart soil monitoring system with crop and fertilizer recommendations, *2025 International Conference on Visual Analytics and Data Visualization (ICVADV)*, pp. 412–419.

Gao, Y., Gizis, A., Nekkanti, S. K., Shitole, R. R. and Wood, T. (2025). Towards sustainable machine learning with serverless at the edge, *2025 IEEE Cloud Summit*, pp. 201–204.

Predoaia, I. and García-López, P. (2024). A cloud-agnostic serverless architecture for distributed machine learning, *2024 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*, pp. 131–140.

Shojaee Rad, Z. and Ghobaei-Arani, M. (2024). Data pipeline approaches in serverless computing: a taxonomy, review, and research trends, *Journal of Big Data* **11**.

Singh, P. D., Maurya, S., Verma, R., Singh, K. D., Joshi, K. and Khanna, L. S. (2024). Justifiable agriculture productivity with nimble, cost-effective, and optimized resource framework - a mist computing approach for smart agriculture, *2023 International Conference on Smart Devices (ICSD)*, pp. 1–5.

Somayaji, A. B. G., Jothimani, K., Rajeshwari, M. and Neema, H. (2025). Transforming agriculture in india with iot and ml: A smart system for optimized water management and crop health, *2025 IEEE International Conference on Contemporary Computing and Communications (InC4)*.