National College of Ireland

Smart Farming IoT sensor data filtering using Pattern Analysis and Edge computing to reduce latency

MSc Research Project
MSc in Cloud Computing

Priya Patil
Student ID: 21121095

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Priya Patil
Student ID:	21121095
Programme:	MSc. Cloud Computing
Year:	2022
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	15/12/2022
Project Title:	Smart Farming IoT sensor data filtering using Pattern Analysis and Edge computing to reduce latency
Word Count:	8058
Page Count:	21

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:	Priya Patil
Date:	14th December 2022

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):	

Smart Farming IoT sensor data filtering using Pattern Analysis and Edge computing to reduce latency

Priya Patil
21121095

Abstract

From prehistoric times to the present, agriculture has always played an important role because of how crucial it is to the continued existence of humans. Given the world's expanding population, it is essential to increase agricultural output. The lack of available resources and severe weather has only made matters worse. By combining conventional farming practices with innovations like the Internet of Things (IoT), artificial intelligence (AI), and cloud computing, "smart farming" allows farmers to increase their harvest quality and longevity while still coming close to their crops' full yield potential. Because IoT devices generate so much data, it is not a good idea to transfer all redundant data from IoT sensors to the cloud and risk experiencing latency issues. Data filtering based on patterns can be done locally on the device or at the edge to conserve bandwidth. As a result, this study proposes that pattern recognition be used to eliminate redundancy before data enters the cloud rather than after it arrives. This will be demonstrated by developing a Farm Cloud Solution (FCS) method which focuses on removing the excess redundant data from the temperature and moisture readings gathered from four different farms within the vicinity. The FCS method involves pre-processing with repetition removal (RR), invalid data removal (IDR), and linear redundancy removal (LRR). A minimum 28.6 % improvement in the latency and more than 5700 bytes were saved from an average of 16136 bytes of data, roughly 35% of the data. The error is allowed within a reasonable 1% margin, allowing the farmer to forecast the upcoming harvest. Less data was sent to the cloud while retaining all vital information This FCS algorithm is a unique approach to two-tier architecture and could potentially prove to be the best option for smart farming technology.

1 Introduction

1.1 Background

Increased demand for agricultural production is a direct result of the rapid increase in the global population. Since agricultural growth is so critical to the entire progress of a nation, a significant amount of effort has been spent on the development of more advanced agricultural technologies. A farmer can come considerably closer to their crop's maximal productivity and quality when the temperatures and humidity are just right for it. Consequently, scientists and farmers have started employing a strategy known as "smart farming" to boost agricultural output, save energy, and decrease negative atmospheric consequences. (Ferehan et al. (2022))

1.2 Motivation

Due to time constraints, farmers frequently lack the information necessary to identify when and where their crops will grow, which makes it difficult for them to plan effectively. Hence a method called "Smart farming" has been accepted by both scientists and farmers to boost agricultural productivity while simultaneously lowering the number of wasted resources and negative effects on the environment. (Muangprathub et al. (2019))

1.3 Research Question

Most Internet of Things devices requires a connection to the cloud for the data they produce to be successfully stored and processed. Despite this, a significant portion of the information that is gathered by IoT agricultural sensors is redundant. Because of this, it is not suggested to immediately upload redundant sensor data to the cloud, as doing so consumes resources and slows down processing power. The conventional computing model is not well adapted to the transfer of the ever-growing flow of actual data. Bandwidth constraints, latency concerns, and unanticipated network outages can all impede such endeavors.

R.Q.- "To what extent can the repetition of data be removed to reduce latency in Smart farming IoT applications using pattern analysis and edge computing"

1.4 Solution

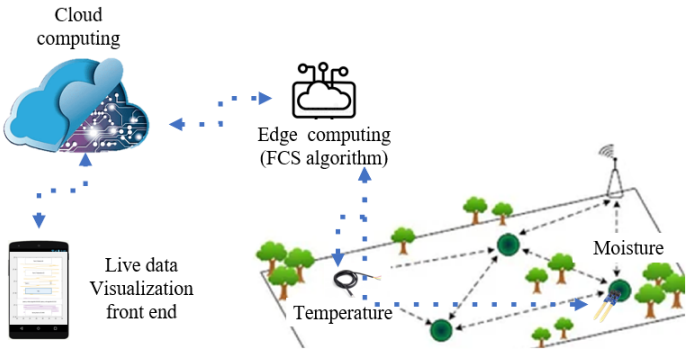

Figure 1: Semantic demonstration of redundant sensor data elimination at the edge computer

A unique approach was proposed as shown in figure 1, that can remove redundant data using pattern analysis at the edge computer itself and hence save bandwidth and latency in a two-tier architecture. The more actions performed close to the edge, the less data must be sent across vast distances. To address the issue of latency, huge amounts of data must be handled near their source, i.e., by means of edge computing, which allows enterprises to consume less internet traffic. In this study, the most commonly available dataset was used from the farming sensors, i.e., humidity and temperature. The data is pre-processed for invalid reading detection and removal. The repeating data is a major issue with slowly varying quantities like temperature and moisture. Hence, in the proposed algorithm, repeated values were removed and instead stored time points that

can let us reproduce the data at the receiver. For cases where a linear relationship could be established for the data points, just send the slope, and offset is sent. (Nagasubramanian et al. (2021))

2 Related Work

This chapter presents a discussion and analysis of the published research on smart farming by utilizing the Internet of Things and Edge computing in conjunction with pattern recognition. In addition to this, it offers a comprehensive analysis of the systems that have been described in the previous research and contrasts it critically. By doing a thorough literature review, we can evaluate the quality of the previous studies conducted on the subject, identify any knowledge gaps, and learn more about the field in general.

2.1 Integration of Edge Computing with IoT Devices

In the conventional design of the Internet of Things, sensors collect data and send it to a remote data centre or the cloud. There may be bottlenecks if a lot of data is being sent and received from a device, which would make this approach ineffective in any situation where latency is critical. IoT edge computing, which relocates data processing close to IoT devices, solves this issue. With this method, the system may do near-real-time data processing locally, shortening the data pipeline in the process. Using data processing at the network's edge, or "edge computing," is one way to improve the performance of an Internet of Things (IoT) system. Edge computing allows an intelligent device to process raw Internet of Things data at a nearby edge server instead of sending it to a central data center.

Izolan et al. (2020) designed a system that uses fog computing to do the management of the moisture in the soil. They also did an analysis of the collected data to provide the density map of the soil moisture. Based on the solid moisture level, the frequency of avigation is decided. They studied the performance of four different types of soil and found that field capacity is maximum (44%) for clay-type soil. The edge is sending the data every five minutes to fog, and fog was sending data every 30 minutes to the cloud. Their overall accuracy was 98%. and the testing period was about 48 hours.

A wireless sensor network can now collect data from different farms courtesy of an edge computer created by Li et al. (2020). Each WSN would be deployed in the farm, linked to edge computing, and then the user would receive data from the cloud, as proposed by them. As the data rate changed from 6 Mbps to 60 Mbps, they were able to cut the delay from 75 seconds to 25 seconds. In comparison to the other methods, which only managed 50% and 14% data quality, theirs was a perfect 100%. While they use a multi-tiered method, the proposed edge computer must have ample computational power to process data from many WSN sensors. Even at a speed of 60 Mbps, their latency is still above 25 seconds. Our project aims to cut down on this lag time. Additionally, get rid of the WSN connection if possible. As a result of the WSN connection, it is necessary to provide each node with its own source of power and with regular upkeep. In comparison to the higher upfront cost of WSN nodes and the ongoing cost of maintenance after deployment, wiring down the sensors may not be a major concern for a small-scale farmer. also want to fill the void in the literature about data quality, where some studies report a perfect score of 100% based on a sample size of ten sensors while others report rates of accuracy of 55% or lower.

The autonomous agricultural robot created by Khan et al. (2020) makes use of two-tier architecture, including the Internet of Things edge and the Internet of Things server. Raspberry Pi 3 was used in the creation of this network. Images were collected and processed to determine which ones had weeds and which did not. They used data from auxiliary sensors that measured things like ambient light, temperature, soil moisture, and relative humidity. With this data, they could determine whether the crop contained weeds. Similarly, the advantage of information about soil temperature and moisture was taken to guide our method. However, our research method is centered on employing 1D data, which has a lower computational cost than 2D data, rather than the camera as the primary data source. When just Wi-Fi would have done, they utilized both NRF and it was overkill. This lab studying many communication protocols will be managed by our research proposal, which will zero in on a single protocol.

2.2 Smart farming and Precision agriculture

Increased output lessened environmental impact, and optimal use of resources are all goals of "smart farming," which makes use of emerging technologies in agricultural and livestock production. As a result of incorporating technology into agricultural and livestock production, global food security can also be increased. In a future with overpopulation, automatic systems to water, fertilize, and fumigate each area according to its features; smart sensors that aid in the early identification of infestations and the weather forecast; drones to monitor hundreds of acres to assess the health of plants and animals; these are all examples of how "smart farming" may help to end world hunger.

Precision farming for saffron has been envisioned on a three-tiered scale by Kour et al. (2022). They utilized USP 30 WIFI modules, which collect and analyze data in real-time before uploading it to the cloud for big data research. They were able to implement smart decisions more easily in precision farming because they just had to worry about one crop of saffron. A potential goal of our suggested study is to improve upon an analogous single-crop Pattern Recognition technique. The development of a generalized two-tier system (without fog) to achieve the same goals is of interest to us, nevertheless. One area they concentrated on for improvement was reducing processing and networking lag. They also discovered that embedded operating systems benefit from the efficiency boost provided by fog-enabled gateway services while sending data. To mimic their gateway implementation, an edge gateway was used instead of a fog service that is native to an embedded OS. Thus, avoiding any delays caused by the network.

Kulbacki et al. (2018) utilizing a variety of computational methods, conducted a survey of drone technology for precision agriculture. From sowing to the gathering, they discussed every possible use of drones in agriculture. Drones, they said, will eventually become inexpensive and able to map difficult terrains. Due to their limited depth and breadth, 2D input data like drone photographs are mostly disregarded in our study proposal in favor of 1D sensors, which can provide in-depth information and are more accurate than images. The primary reason for favoring 1D sensor data over 2D hyperspectral image costs; a hyperspectral camera and drone together cost nearly 45 times as much as 1D sensors that can run on their own solar energy. While some farmers may be able to purchase a drone and hyperspectral imaging technology, others may not. To fill the knowledge gap in technology evaluations that is caused by a lack of cost data. Here, instead of taking a theoretical tack, a more grounded, application-oriented approach to our study proposal was taken.

A demonstration of the use of hyperspectral data in precision agriculture has been provided by Yao et al. (2018). Their research centered on analyzing the state of hyperspectral imaging technology such as satellite imagery, MAVs, GVs, and UAVs. Their research centered on such topics as nitrogen trace detection in crops, weed mapping, and fertility sensing. They designed a two-tier system that would use hyperspectral imaging data to analyze crops on a large scale. In the same vein as their approach, one of these values as Pattern Recognition's final objective is used. Hyper-spectrum images won't be applied, but efforts are towards applying their computational methods to more general datasets like temperature and humidity in a single dimension.

2.3 Implementation of pattern Recognition utilizing multiple approaches

Automatic recognition of patterns and regularities in data is the goal of Pattern Recognition, a data analysis technique that makes use of machine learning techniques. The term "pattern recognition" refers to the process of automatically spotting repeating structures within a dataset. It's a technique for automatically categorizing information as belonging to a particular mathematical form, which opens the door to the representation of more information. By identifying patterns in the data, pattern recognition makes it possible to remove less information during the filtering process. As a bonus to compression, pattern recognition provides forecasting. Recent advances in pattern recognition have led to the creation of a wide variety of methods, including categorical labeling, clustering, ensemble learning, and regression. Research in the field of pattern recognition technology includes the following.

An ensemble classification system for tracking crop diseases was developed by Nagasubramanian et al. (2021). By detecting crop illnesses early, their IoT-based Pattern Recognition technology helps farmers save a significant amount of money in the form of pesticides. Early discovery will aid in providing greater nourishment for the plants. They have been using nonlinear ensemble SVM for pattern recognition (Support Vector Machine). They also proposed additional methods for disease prediction, including the use of simple SVM, CNN, Naive Bayes, and K-nearest neighbors. An Arduino-based edge computer was used to collect data from a variety of sensors for their IoT project. LN35 for measuring temperature, TO-92-3 for measuring humidity, LN385 for measuring soil moisture, and DHT11 for measuring humidity in the air were their respective sensor modules. They utilized a BH1750 to gauge the brightness and a YF-S201 to gauge the velocity of the water. Data from sensors were combined with information from a hyperspectral camera (HySpex). In contrast to other publications, they relied primarily on data from sensors and secondarily on data from hyperspectral images.
The secondary data, farm photos, and hyperspectral data will be disregarded in our study. These images are typically disregarded because they are computationally expensive but contribute no information that can't be obtained with a standard 1D sensor. Temperature and humidity sensors from the many available 1D sensors were zeroed. Similarly, measurements of light, soil moisture, and water flow as alternative input data sources were used. The more sunlight there is, the higher the temperature will be; therefore, we've been ignoring it. The light was taken care of by monitoring the temperature and humidity because more sunlight means less humidity. Since the relative humidity in the air is precisely proportional to the soil moisture, don't need to enter that data as one of our major sensor inputs. That's because the soil is better able to retain water in

conditions of higher humidity. If temperatures continue to rise, it will be a sign that soil moisture is decreasing. That's because the soil loses moisture at a faster rate when the temperature rises. Consequently, there is a dependence between the parameters measured by their principal sensors. By focusing on the minimal minimum of sensors, our study will attempt to close this dependency research gap.

Di et al. (2019) built a systemic pattern recognition mechanism built will presumably keep an eye on 10 separate system patterns. They suggested a cognitive data analysis of machine and automobile data to keep an eye on seismic data. The researchers concentrated on Semitic data, but their methods are applicable to other types of sensor data, including humidity and temperature, and farming. The thinking behind this was to identify the most prevalent design principle from the set of 12, so that it may be optimized and made data-driven. Our suggested study aims to create a data-driven pattern recognition technique to get over the limitations of traditional pattern recognition.

Milking parameter Pattern Recognition is an area of research for Ebrahimie et al. (2018). Mastitis risk management in milk production was the focus of their efforts. Decision trees, random forests, and other machine-learning methods were the focus. They built 11 datasets with 527 models using 346,000 miking records. They might be 90Despite having access to such a large database, achieving just 90% accuracy with straightforward models like a tree was inefficient. A more sophisticated and intricate machine learning model may have been used for recognition. Investigating their proposed decision tree to arrive at the optimal pattern recognition policy is proposed here.

Kour et al. (2022) have developed a system for the precision farming of saffron. Data was gathered using USP 32 WIFI modules, which processed it in real-time before uploading it to the cloud for big data analysis. With only saffron as their primary crop, they were able to make more informed judgments and use more precise agricultural techniques. One possible goal of our suggested study is to improve upon an analogous single-crop Pattern Recognition technique. However, it is of interest to create a universal two-tier system (without fog) that can provide the same outcomes in the long run. One of the areas they concentrated on for improvement was reducing processing and networking lag. Additionally, they discovered that embedded operating systems benefit from the use of gateway services that are enabled by fog. To reduce network latency, a gateway strategy like theirs is used and employs an edge gateway instead of a fog service that runs directly on an embedded OS system.

2.4 Conclusion of Literature Review

Papers published by Nagasubramniam. (2021), Yao (2019), and Kulbacki et al. (2019) all concentrated on agricultural analysis utilizing multispectral photos and drone-captured images. Since this multi-spectrum imaging camera is quite expensive, we have opted out of this type of data analysis that is image-based. However, the licensing requirements and security concerns associated with drone technology significantly increase the complexity of its deployment and use. Instead of using Aerial measurements, we opted for sensors that are readily available and capable of doing actual measurements within the farm.

3 Methodology

This chapter delves into how our proposed FCS algorithm fills the gaps of previous literature reviews. This proposed effort aims to employ the FCS algorithm to devise a means of eliminating unnecessary data before it is uploaded to the cloud. Only information related to agriculture for temperature and moisture sensors is used in this proposal.

3.1 Research Approach

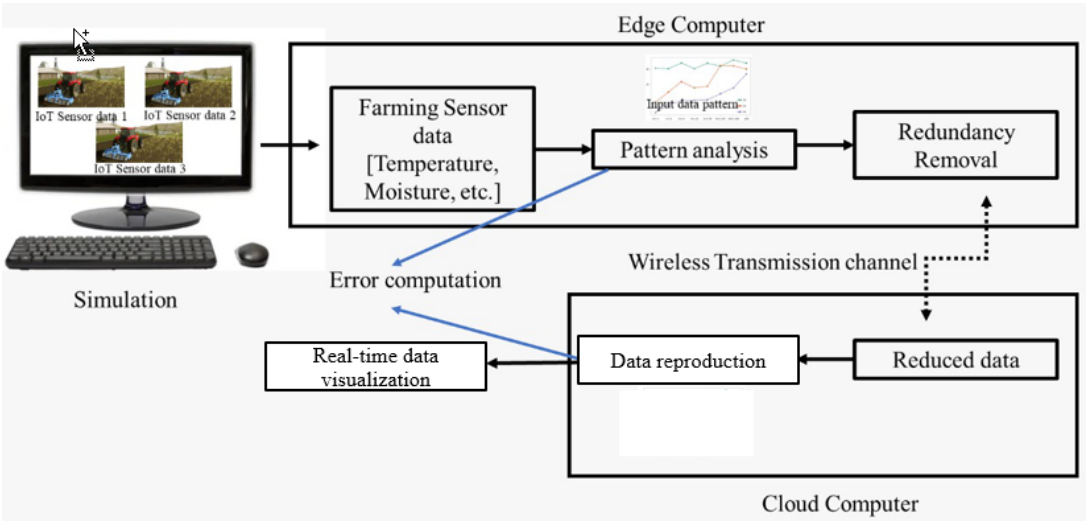

Figure 2: Block diagram of Farm Cloud Solution algorithm

Fig 2. above shows the proposed block diagram for the FCS algorithm. The data received from agricultural sensors such as moisture and temperature are collected over 84 days. This data was collected over 3 months and could be massive and will have a lot of redundancy. It is aimed to introduce data reducing FCS algorithm that performs pattern recognition data reduction tasks. Linear redundancy will be removed from this data and data is also checked for valid information. The linear pattern between the data is recognized. If the pattern is recognized with R-squared greater than or equal to 0.9 then only the slope and bias of the line fitting the data are sent for further processing.

3.2 Tools Used

MATLAB, Excel, Origin Lab, and PowerPoint were used for analysis and graphical plotting.

3.2.1 MATLAB

- MATLAB is a licensed cross-programming language and numerical programming environment built by MathWorks.
- MATLAB can compute matrices, visualize functions and data, execute algorithms, design UI, and communicate with other languages.
- MATLAB is used for programming the backend of edge and cloud-side functionality.

3.2.2 GUIDE

- The UI was developed with the help of GUIDE (Graphical User Interface Designer).
- The GUIDE is used for front-end development with drag-and-drop tools.
- A Fig window is created for which the guide generates the automated code that can be used by the developer to build the backend logic.

3.2.3 Excel

- Microsoft Excel allows spreadsheet formatting, organization, and calculation.
- Excel was used to read the CSV Dataset file before programming to know the number of rows, column, and their corresponding headings.

3.2.4 Origin lab

- Origin is computer software for data processing and dynamic scientific plotting.
- Origin lab is used to plot each experiment's outputs. Also, the statistical numbers are generated such as r opt, mean-variance, etc.

3.2.5 PowerPoint

- PowerPoint is used for creating slideshow Presentations.
- Software is used to make diagrams for the report and any kind of presentation.

3.3 Research Procedural Flow

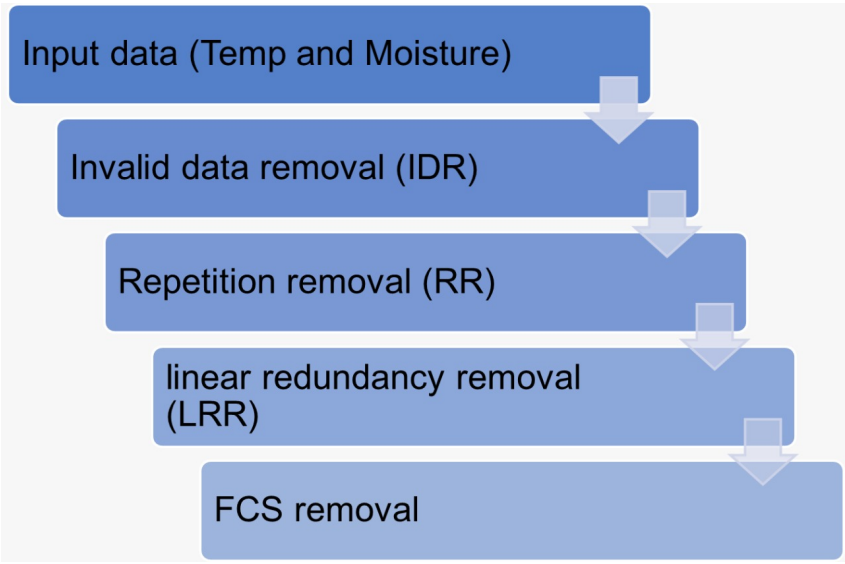

Figure 3: Data flow diagram of FCS solution

As can be seen in Fig 3, a step-by-step representation of data flowing through the system is explained below.

The farm temperature and humidity sensor dataset from Kaggle is used for this research project. (Mehendale (2022)). The input data is read as a table from the CSV dataset file. This table consists of separate timestamps, moisture, and temperature readings.

The invalid data is removed using the IDR algorithm. Here, every reading is tested for validity before it is sent to the FCS. During invalid data removal, all the readings are checked one by one. If the value is missing during the check, the corresponding time point and all the possible corrupt time points are removed from the data.

Once the filtered data is received from the IDR algorithm, it is processed for repetition removal. During repetition removal, every data point is read in relation to its previous data point. If the change is zero for consecutive data points, it is considered a repetition of the same data. All the repeating values and their corresponding time points are removed.

The repetition-free data is then fed to the pattern recognition algorithm. A linear relationship is investigated during pattern recognition, and slop and offset are found, which can save a lot of data. During linear redundancy removal, a set of 200 data points is considered together to determine if there is any linear redundancy. If linear redundancy is found, its corresponding r-squared value is checked, and if it is greater than 0.09, it is considered valid redundancy, and the corresponding slope and offset are sent.

During FCS removal, dual parameters are adjusted to optimize for minimal data transmission. The FCS algorithm consists of dual parameter tuning, i.e., the simultaneous determination of a data multiplying factor and a pattern recognition factor. Any instances following the said behavior with an acceptable error are compressed, and only n and k values are transmitted. The data can be easily reconstructed at the cloud end if the seed value and parameters such as n and k are known.

3.4 Algorithm

The algorithm section covers details about the proposed algorithm.(Saiz-Rubio and Rovira-Más (2020)) and its corresponding steps to be followed. A total of 9 steps are required to achieve the proposed logic. These steps are explained in detail in this section. Step 3 is specific to the proposed farming application which can be replaced for any other application on demand.

Algorithm 1 FCS Algorithm

Require: $TemperatureData$
Require: $MoistureData$
Require: $AcceptabletemperatureThresholdvalue$
  $Th \leftarrow AcceptableError$
Ensure: $Th < 0.01$
  $T \leftarrow Temperature$
  $M \leftarrow Moisture$
  $e \leftarrow Error$
  $K \leftarrow Patternrecognitionfactor(Kisinitiallysetto3)$
  $N \leftarrow Datamultiplierfactor(Nisinitiallysetto10)$
  while $DS > n$ **do**
    $T \leftarrow T$ ▷ Invalid data detection for T
    $M \leftarrow M$ ▷ Invalid data detection for M

$T \leftarrow T, T$ ▷ Data repeat removal for T
$M \leftarrow M, M$ ▷ Data repeat removal for M
$T \leftarrow T, 2T...$ ▷ Linear Data Redundancy removal for T
$M \leftarrow M, 2M...$ ▷ Linear Data Redundancy removal for M
while eTh **do** $K = K + 1/K - 1|e > 0/ < 0$
    if $K > 0$ **then**
        $N \leftarrow \frac{N}{2}$
        $if\ N < 1\ then\ Reset\ N$
    else if $K > 0$ **then**
        Save e
        Save K ,N
    end if
end while
$DS \leftarrow Databytessaved$
$SF \leftarrow Savingfactorinpercent$
$a \leftarrow T, M$
▷ Transmitt =0

3.5 Steps of FCS algorithm

1	Collect the temperature data and moisture data
2	Get the acceptable threshold from the user. Else, set to 90%
3	Assign variables T,M,e,K,N Where, T - Temperature reading M - Moisture reading e - error K - pattern Recognition factor N - data multiplier
4	If the given number is within an acceptable threshold. repeat K=K+1 if the error is positive
5	If k reaches the limit then n is reduced to half. If n reaches the limit, n is reset while keeping k in its original seed position.
6	If the error is within the limit n and k values are transmitted along with its seed value. The process is repeated till the last bit of the data is in the serial buffer. The data to be transmitted is saved, and latency with and without the algorithm is computed
7	The saving factor is computed.
8	The alpha is loaded with a new TM buffer and the process continues.
9	Repeat steps 1-8

3.6 Formulas Used

All the formulas used in the proposed algorithm with their corresponding variables and assumptions are mentioned in this subsection.

1. Y = mX+C
    - Y is the predicted value,
    - m is the slope of the line,
    - X is the current value,
    - C is the offset (if any)
2. Datarate = (473*1024)/8
    - 473 is chosen as it is the data rate in kbps in bytes per second EDGE 4G LTE or WiFI 802.1,
    - 1024 is used for conversion from kbps to bps,
    - Divide by 8 operation converts bits to bytes
3. Latency = 1000 * totaldatainbytes/datarate
    - Latency is the total time required to reach from the transmitter to the receiver.
    - totaldatainbytes is the variable that stores the total number of bytes in the data
    - datarate is the speed at which data is transmitted typically measured in bits per second
4. latency_updated = 1000 * saveddatainbytes/data rate
    - latency_updated is the new value of reduced latency post applying the proposed algorithm.
    - savedatainbytes represents the number of bytes saved after applying the proposed algorithm
5. savingfactor_stage_x = latency/latency updated
    - savingfactor_stage_x is the ratio of original latency to updated latency at stage x, where x can be from 1 to 7

4 Design Specification

This section covers details about the User interface as shown in Fig 5 of section 4.1, which is an application for the farmers or users to check the Saving Factor and Latency reduction achieved by the FCS algorithm for the selected Farm.

4.1 User Interface

Fig 4. below shows the Farm Cloud Solutions web application user interface. The First two fields are used for displaying the predicted temperature and moisture readings obtained at the cloud after data filtering. The values in temperature and moisture are real-time and change dynamically. When the user clicks on the Run button, latency reduction achieved with and without data filtering as part of experiment 3, is displayed on the screen. The values mentioned are in milliseconds and computed over a fixed

length of data in both cases. Then it also displays the Saving Factor which states how much percentage of data bytes were saved. Frontend application programming is done mainly using the GUIDE MATLAB tool. The real-time graphs shown in the back are representing 24 hours periodic signals observed from the data.

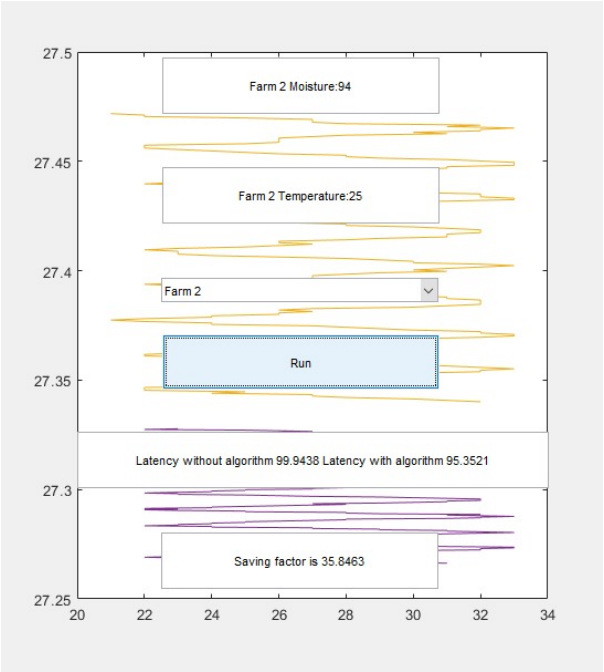

Figure 4: Front-end User Interface - Farm Cloud Solutions web application

5 Implementation

This section covers details information about the implementation of the FCS algorithm.

The dataset was collected by multiple readings of temperature and moisture acquired by the sensors (DHT11). The timepoint and its corresponding temperature and moisture readings were separated to form different arrays. If the data had been transmitted without compression, then the information would have been the total timepoints + total number of moisture readings.

TwoA = tT + tTemp + tMoist

Where,

TwoA = Total number of data bytes transmitted without algorithms.

tT = Total number of timepoints.

tTemp = Total number of temperature readings

tMoist = Total number of moisture readings

All the data is gathered from the Kasheli region in the Mumbai, Karjat suburb. For the past 100 years, the temperature range has remained between 3 degrees Celsius and 45 degrees Celsius. Hence, all the valid temperature readings were preserved, and readings that did not fall under this range were discarded. The typical moisture content measured in the region can vary from 40% to 99%. Hence, assuming some margin for error, the threshold of our margin was set to a minimum of 30 and a maximum of 100.

Once the valid data points are gathered and their corresponding time points are noted, the data reduction logic is applied. In the data reduction, for all the temperature readings, if the previous temperature is within 0.1% of the current temperature, then the current temperature is not transmitted, as this may save a lot of redundancy, and in every iteration, the current temperature is saved as a previous temperature and a new reading is loaded into the current temperature variable. If the temperature change is greater than 0.1%, then the previous value and corresponding information about how many times that previous value has appeared are transmitted.

This is like the concept of run length and coding, which is like the data compression technique. By doing this, slowly varying quantities such as temperature can get rid of repeated readings and hence help with redundancy removal. Like the temperature, all the moisture readings are also checked for repetition, and their redundancy is removed. Moisture is also a very slow-moving parameter and does not change very frequently.

Once the valid and non-repeated data is computed at the edge, linear relationships between the successive data points are tested. All the information on the time axis is considered the x-coordinate, and with a window size of 10 readings, all the values that appeared as the temperature readings were considered the y-axis points. The Polyfit linear function was applied, and the slope, as well as the error, were computed. If the error is greater than 1%, then the linear relationships are discarded. and all the points were transmitted as they are, but if the linear curve fittings are achieved with r squared = 0.99, then y = mx + c, where m is the slope of the line, c is the offset, x is the input (previous value), and y is the output (current value). A special character representing a linear relationship is inserted, followed by slop, offset, and the number of times the equation needs to be used. The window size for linear fitting is chosen after multiple experiments, and 10 was found to be the best-suited number for the dataset.

Once the original information is filtered for non-linear values, repetitions, and linearly proportional numbers, the FCS algorithm is implemented where the predicted value can be computed using the current value, data multiplied, and pattern recognition factor.

Predicted value = (N* current value)/k+) k

Error = actual value

If the error in the equation is less than 0, then the value of k is increased, and if the error is greater, the value of k is reduced. If the error exceeds 10% of the actual value, N is cut in half until it reaches 1, and n is reset if it falls below 1. This means that the data multiplication factor can go anywhere between 1 and infinity, whereas k can go anywhere between minus infinity and infinity. This logic has been derived from the proportionality constant for direct variation with an inverse variation relation.

The proposed novel algorithms not only work for temperature data but can be implemented for other data types that are steady in nature. To validate the final transmitted data towards the cloud, a principal component analysis is conducted, and the latent t square value. The coefficient scores were computed. To analyze the performance of the mobile 4G network, a data rate of 470 kbps is used, and convergence from bytes to bits and bits to kilobits is taken care of. The total number of bytes to be transmitted at the data rate gave us the latency of the proposed system.

There are four main operations carried out on temperature data. Repetition removal. Invalid data removal, linear redundancy removal, and FCS removal. The same procedure was followed for the moisture data. For the window size optimization, a different window size was chosen for moisture as it is slower compared to the temperature reading. The data multiplying factor and prediction factor require proper seed point selection.

6 Evaluation

The evaluation of the experiments performed is discussed in this section. Section 6.1 covers the Different stages at which performance analysis is carried out. And section 6.2. 6.3, 6.4, 6.5 covers the experiments performed as part of the analysis.

6.1 Different stages of performance evaluation

The performance of the system was verified using 2 main parameters i.e., Saving factor and Data bytes saved, formulas for which are mentioned in section 3.5 above.

6.1.1 Stage 1: Invalid data removal

Here, the entire dataset is scanned for invalid data points that do not possibly apply to the given demographic region. Since the data was from India, its temperature range was set between 3 degrees and 45 degrees. And any value above or below his range was considered a faulty value. The moisture range was also validated with values between 30% and 100%. Any values below 30 and above 100 were considered faulty and removed from the actual data. These ranges of temperature and moisture were set while considering the last 100 years' range variation for the demographic

6.1.2 Stage 2: Repetition removal for temperature and moisture data

Here, if any consecutive value gets repeated, i.e., only the time point is changing but the temperature is not, then the newer timepoint is discarded with its temperature value saving data. If both the timepoint and temperature change, then the entire data is forwarded as it is to stage 3.

6.1.3 Stage 3: Linear Redundancy removal

In stage 3, only 200 values were considered at a time to find out the linear relationship, and the entire data was scanned for the y = mx + c equation. And whenever the equation could fit, a special indication character was added to the data, followed by slope, and offset, respectively, along with the time point on the X-axis.

6.2 Experiment 1: Variation of parameter N in the FCS algorithm

For the FCS algorithm to work properly, its data multiplication factor N needs to be tuned. To ensure that N is fully optimized, a parametric switch was conducted with N varying between 5 and 20. The following values were selected for the variation of the N experiment: N=5, 7, 10,12,15, and 20. The distance between the selected value and the next was kept between 2 and 5, and the effect of variation size on the performance was explored. Step sizes of 2, 3, and 5 twice each as a measure of performance were tried. Data bytes saved were the parameter under consideration

6.3 Experiment 2 - Variation of K (pattern recognition factor)

After optimizing the value of N at 10, the value of K is changed from 1 to 8. The initial step size was kept at 1 and then increased to 3. K=1,2,3,4,5,8 is selected. The number of bytes saved was chosen as the parameter to measure the output of this experiment. This was done to determine the optimum value of K for a given N. Each time N changes K value might deviate from its best results.

6.4 Experiment 3 - Latency reduced with and without algorithm

For the computation of the percentage of latency reduced, it was necessary to calculate actual latency without our algorithm. The latency was computed as the number of data bytes transmitted per second * totaldatainbytes/data rate. Once the latency is calculated without the algorithm, the algorithm was run for four different cases, for each case, new latency is computed. This experiment was useful to find the efficiency of the proposed algorithm and how much latency in milliseconds can be saved by giving a side-by-side comparison.

6.5 Experiment 4 - The algorithm is not data dependent

For the analysis of the proposed algorithm for data variation, an experiment is conducted. In this experiment, the inputs given to the algorithm were checked for four farms. In all, four different farm datasets were utilized, each having a moisture reading, a temperature reading, and corresponding time points. To avoid any effect of biasing on the algorithm, it is assumed that all the parameters were defined in the form of variables and not a static value. For each of the four farms, corresponding bytes were saved, and the actual time for the data to reach the cloud was computed. To ensure that the data is compatible, the data format of the same database was used. Through this experiment, it can be proved that the proposed algorithm is not data-dependent. There was only a small variation in the output, which proved that the algorithm is robust.

6.6 Results

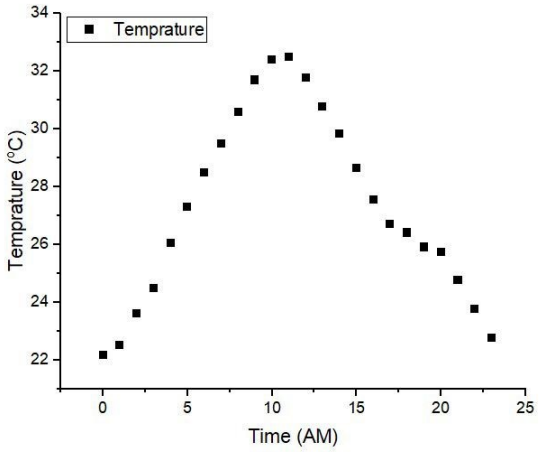

Figure 5: sample time vs temperature plot

Fig. 5 shows the sample time vs. temperature plot for farm 1. The temperature varies by roughly 10 degrees within a day, starting with a cold morning until 5 a.m. Then a hot day is observed until 6 p.m. Also, the temperature changes from 4 p.m. to 8 p.m. in the evening. The maximum temperature is around 1 p.m.

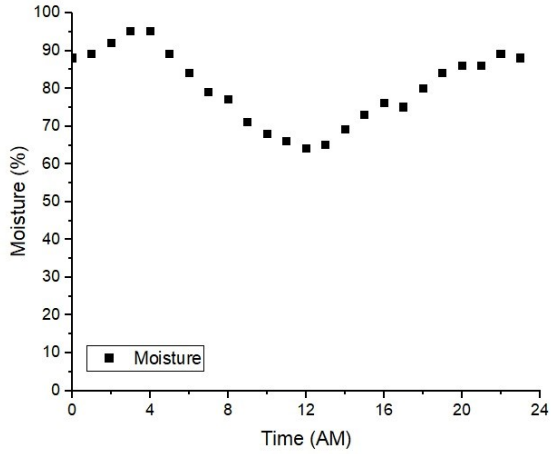

Figure 6: The plot of moisture vs time

Fig 6. shows that the moisture reading is cyclic in nature and maximum moisture was observed in the air somewhere during midnight. And the minimum moisture is around noon.

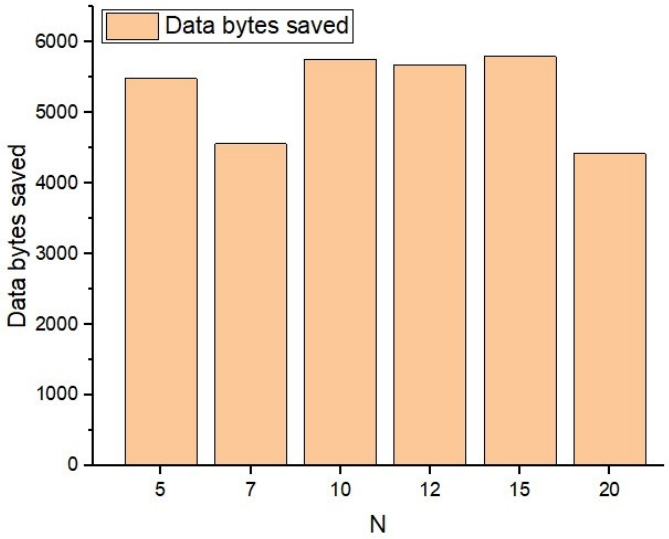

Figure 7: . Shows the variation of N and its corresponding effects on data bytes saved

Fig 7 is the result of experiment 1. The N was varied from 5 to 20 with the step size of 2 3, and 5. It was observed that maximum data bytes were saved for values 10, 12, and 15. Hence, N=10 is chosen. It is observed that with respect to N variation, data bytes saved could reach up to 3000 to 4000 which means 1/3 rd of the data is not sent to the

cloud and that itself is a big achievement. With this algorithm, we saved 25% latency reduction.

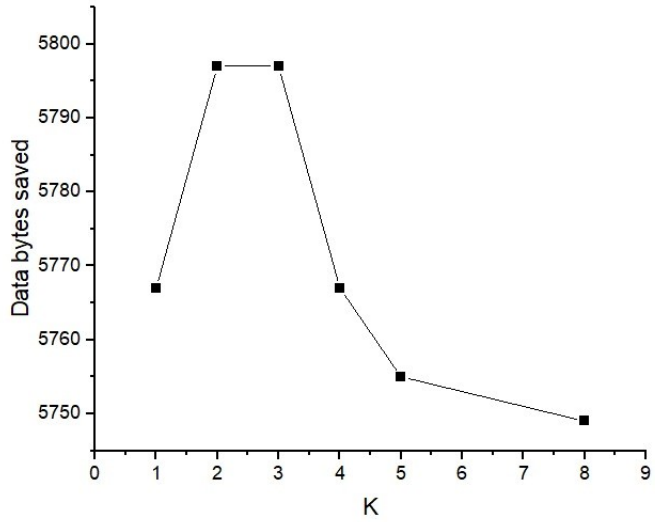

Figure 8: The plot of K variation vs time bytes saved.

Fig 8 is the result of experiment 2. The data bytes saved are minutely affected by the variation in the k value. Tried K from 1 up to 8 as a seed point and observed that around k =2 or 3, the value reaches the maximum saving point and hence selected k =3

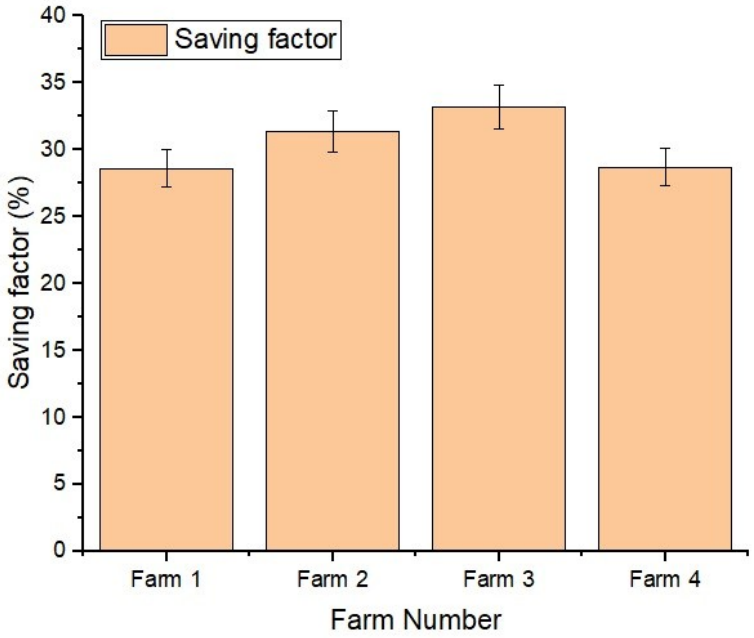

Figure 9: Variation of location vs % saving factor

Fig 9 is the result of experiment 3. Data from four different farms were gathered, and independent tests were carried out to check the overall efficiency of the proposed

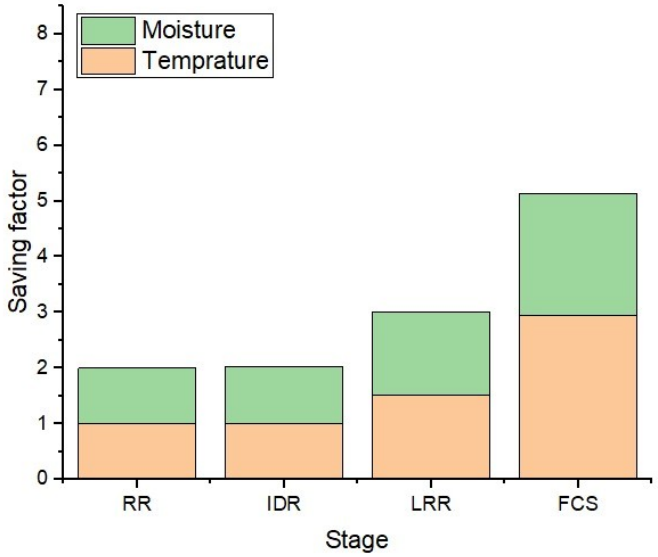

Figure 10: Different stages of the algorithm and corresponding latency improvement in terms of % saving factor.

algorithm. Regardless of farm location, at least a 25% improvement in latency can be observed. The maximum saving factor observed was 35%, as none of the readings were constant.

Fig 10 is the result of experiment 4. During repetition removal, roughly 1% of the latency was reduced for both moisture and temperature. During IDR (invalid data removal), another 1% of latency is removed. Overall Linearity improved by 3%, with improvement in both moisture and temperature being 1.5% each. Maximum improvement was achieved with FCS, which is more than 5%. Here, the temperature was better, with a 3% improvement, and the moisture content was roughly 2.5%. The overall combined logic could save around 25% of the time. and can go up to 35% as well.

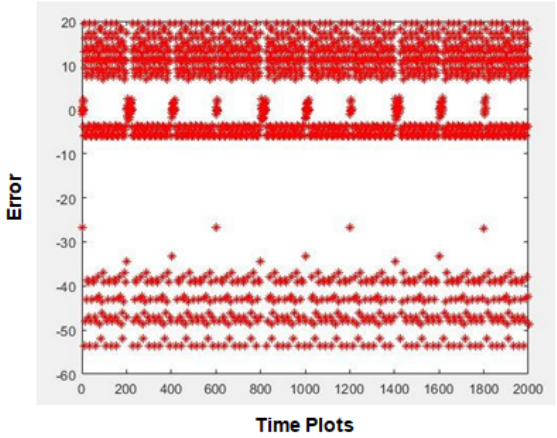

Figure 11: Convergence plots for the error

Fig. 11 shows the convergence plots for the error against different time points due to the proposed FCS algorithm. The error ranges between +20 to -60.and repeats after

approximately 200 iterations, it reaches 0 error. Due to the continuous fluctuation in the hourly humidity readings, the error remains zero for a very short duration. and varies between 0 to -3 for most of the readings. The reset operation of N was causing the error to reduce from -60 to +20. And a variation of K was bringing the plot closer to 0. Ideally, this plot should have been a straight line at e = 0, but practically, it seemed that humidity remains constant for very few readings, i.e., not more than 5 hours.

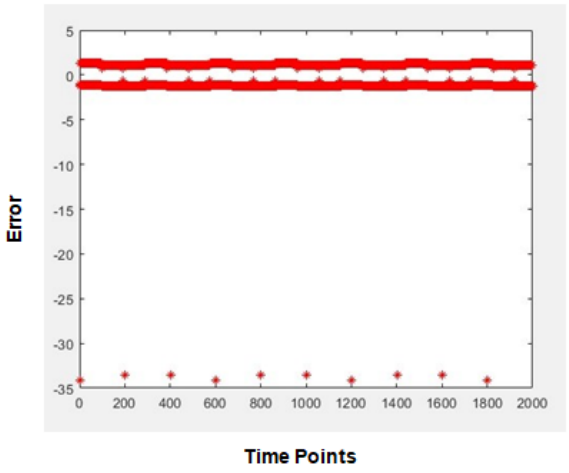

Figure 12: Error vs time point plot for the temperature against different time points readings with the FCS algorithm

Fig. 12 shows that most of the readings keep cycling between 03 and +3. and it goes to -35 whenever N is reset. The typical recent frequency was observed at N = 200. Also, the periodic nature indicates that the proposed algorithm can work efficiently as per the window size. Since none of the readings reach zero error, a 1% margin of error must be considered. and thus, allowing a saving of 196 bytes. just by sending the values of N and K along with the initial temperature.

6.7 Discussion

During the experimentation, it was found that the temperature and moisture data can be considered periodic within a period of 24 hours with a slight variation in amplitude. Although the sensors can gather the temperature and moisture data within a second, the actual reading does not fluctuate so fast. And even if there are some minor fluctuations, they are negligible. It was observed that temperature follows a bell-shaped curve whereas moisture follows an 's-shaped curve. The convergence during the FCS algorithm was found to be considerably slower than reported in the literature. The FCS algorithm convergence during the temperature plot was much better than the moisture plot. To ensure that the readings were not biased with the single farm, multiple farms were introduced in the study and the algorithm proved to be data independent. Overall FCS proved to be efficient in reducing the latency by filtering the unnecessary sensor data.

7 Conclusion and Future Work

With different experimentation, it can be concluded that the proposed FCS algorithm in combination with RR, ITR, and LRR can prove to be an efficient way of transmitting the data to the cloud. It is observed that the algorithm is independent of the place from which the reading is taken. In all, at least 25% of the latency is removed due to our proposed mechanism, and hence it is logical to use this method whenever sending data to the cloud. Although N and K values were varied and finalized N = 10 and k = 3, respectively, there is still scope for future researchers to explore whether the maxima found were local or global maxima.

The study was limited to 2 sensor data but with current advancements in technology, it is possible to have multiple sensor data. It also has a limitation of the demographic region with all four farms within a specific region, the study could have been extended to multiple geo-locations across the world. In addition to that, instead of textual sensor reading, new-generation camera-based sensors can also use this algorithm.

References

Di, H., Gao, D. and AlRegib, G. (2019). Developing a seismic texture analysis neural network for machine-aided seismic pattern recognition and classification, *Geophys. J. Int.* **218**(2): 1262–1275.

Ebrahimie, E., Ebrahimi, F., Ebrahimi, M., Tomlinson, S. and Petrovski, K. R. (2018). A large-scale study of indicators of sub-clinical mastitis in dairy cattle by attribute weighting analysis of milk composition features: highlighting the predictive power of lactose and electrical conductivity, *J. Dairy Res.* **85**(2): 193–200.

Ferehan, N., Haqiq, A. and Ahmad, M. W. (2022). Smart farming system based on intelligent internet of things and predictive analytics, *Journal of Food Quality* .

Izolan, P. L. R., Rossi, F. D., Hohemberger, R., Konzen, M. P., Da Cunha Rodrigues, G., Saquette, L. R., Temp, D. C., Lorenzon, A. F. and Luizelli, M. C. (2020). Low-cost fog computing platform for soil moisture management, *2020 International Conference on Information Networking (ICOIN)*, IEEE, pp. 499–504.

Khan, A., Aziz, S., Bashir, M. and Khan, M. U. (2020). Iot and wireless sensor network based autonomous farming robot, *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)*, IEEE, pp. 1–5.

Kour, K., Gupta, D., Gupta, K., Juneja, S., Kaur, M., Alharbi, A. H. and Lee, H.-N. (2022). Controlling agronomic variables of saffron crop using iot for sustainable agriculture, *Sustainability* **14**(9): 5607.

Kulbacki, M., Segen, J., Kniec, W., Klempous, R., Kluwak, K., Nikodem, J., Kulbacka, J. and Serester, A. (2018). Survey of drones for agriculture automation from planting to harvest, *2018 IEEE 22nd International Conference on Intelligent Engineering Systems (INES)*, IEEE.

Li, X., Zhu, L., Chu, X. and Fu, H. (2020). Edge computing-enabled wireless sensor networks for multiple data collection tasks in smart agriculture, *Journal of Sensors* .

Mehendale, N. (2022). Farm temp and humidity.

Muangprathub, J., Boonnam, N., Kajornkasirat, S., Lekbangpong, N., Wanichsombat, A. and Nillaor, P. (2019). Iot and agriculture data analysis for smart farm, *Computers and electronics in agriculture* **156**: 467–474.

Nagasubramanian, G., Sakthivel, R. K., Patan, R., Sankayya, M., Daneshmand, M. and Gandomi, A. H. (2021). Ensemble classification and iot-based pattern recognition for crop disease monitoring system, *IEEE Internet of Things Journal* **8**(16): 12847–12854.

Saiz-Rubio, V. and Rovira-Más, F. (2020). From smart farming towards agriculture 5.0: A review on crop data management, *Agronomy (Basel)* **10**(2): 207.

Yao, H., Huang, Y., Tang, L., Tian, L., Bhatnagar, D. and Cleveland, T. E. (2018). Using hyperspectral data in precision farming applications, *Advanced Applications in Remote Sensing of Agricultural Crops and Natural Vegetation*, CRC Press, pp. 3–35.