agriculture

MDPI

Article

Artificial Intelligence-Based Fault Diagnosis and Prediction for Smart Farm Information and Communication Technology Equipment

Hyeon O. Choe [1] and Meong-Hun Lee [2,*]

[1] Department of Information and Communication Engineering, Sunchon National University, Suncheon-si 57922, Jellanam-do, Republic of Korea; wishind@scnu.ac.kr
[2] Department of Smart Agriculture Major, Sunchon National University, Suncheon-si 57922, Jellanam-do, Republic of Korea
* Correspondence: leemh777@scnu.ac.kr

Abstract: Despite the recent increase in smart farming practices, system uncertainty and difficulties associated with maintaining farming sites hinder their widespread adoption. Agricultural production systems are extremely sensitive to operational downtime caused by malfunctions because it can damage crops. To resolve this problem, the types of abnormal data, the present error determination techniques for each data type, and the accuracy of anomaly data determination based on spatial understanding of the sensed values are classified in this paper. We design and implement a system to detect and predict abnormal data using a recurrent neural network algorithm and diagnose malfunctions using an ontological technique. The proposed system comprises the cloud in charge of the IoT equipment installed in the farm testbed, communication and control, system management, and a common framework based on machine learning and deep learning for fault diagnosis. It exhibits excellent prediction performance, with a root mean square error of 0.073 for the long short-term memory model. Considering the increasing number of agricultural production facilities in recent years, the results of this study are expected to prevent damage to farms due to downtime caused by mistakes, faults, and aging.

Keywords: smart farming; sensors; RNN; LSTM; ontology; prediction

Citation: Choe, H.O.; Lee, M.-H. Artificial Intelligence-Based Fault Diagnosis and Prediction for Smart Farm Information and Communication Technology Equipment. *Agriculture* **2023**, *13*, 2124. https://doi.org/10.3390/agriculture13112124

Academic Editors: Maciej Zaborowicz and Jakub Frankowski

Received: 5 October 2023
Revised: 30 October 2023
Accepted: 2 November 2023
Published: 10 November 2023

1. Introduction

In recent years, rapid progress has been made in agricultural technology in terms of enhancements in productivity and convenience, which are together referred to as smart farming. This has been facilitated by the convergence of various information and communication technologies (ICTs) [1]. In countries around the world, including South Korea, agriculture is undergoing technological evolution via smart convergence based on data collection, analysis, and prediction. Current agricultural practice is focused on developing differentiated technology related to "software and hardware platforms", "data intelligence", and "convergence of various technologies, such as artificial intelligence (AI), the cloud, and the Internet of Things (IoT)" to develop an intelligent smart farming industry [2]. Smart farms centered around facility horticulture provide services related to crop growth, environmental information management, system control, disease control, and growth algorithms customized to suit the requirements of local farmers. However, the applicability of these technologies to the current agricultural production process remains limited. A first-generation smart farming system operated based on ICT convergence was developed with a focus on labor reduction and convenience. However, it suffers from several problems, including difficulties in checking system operation and remote-monitoring-based control, cost-intensive CCTV operation for visual monitoring, and the high complexity of criteria for assessing abnormalities in sensing values. Besides such technical and financial problems,

the application of advanced technologies to real-world local farms suffers from additional basic limitations [3–5].

The South Korean government's smart farm distribution project has considerably increased the number of smart farms across the country, but the aforementioned problems hinder the widespread adoption of smart farming. Unlike other sectors, an agricultural production system is extremely vulnerable to fault-induced downtime, which can lead to irrevocable damage to crops and farms, in addition to incurring maintenance and repair costs. Therefore, there is a compelling need for system prognostics, health management technology, and condition-based maintenance technology [6]. This will prevent damage to farms due to downtime induced by accidents, faults, and aging based on meticulous data monitoring of ICT equipment in agricultural production facilities. Several studies have investigated sensor-based fault detection by classifying various types of anomalous data in the field of sensor networks and proposed error detection methods for each data type [7]. Faults detected based on sensing values have been assessed corresponding to individual and multiple data within a given space. Particularly, a moving-average-based assessment technique was used to study time series, and anomalous data were assessed based on a geospatial understanding of the sensing values [3,4]. One study pointed out the problem of poor generalizability of the methods used for sensor outlier assessment based on the ambiguity of data forms and models [8]. Traditional outlier treatment uses Bayesian analysis of data sensed within a specific space; however, the utilization of only limit values within a specific dataset, such as temperature or humidity, is not suitable for facility horticulture smart farming [9,10].

To this end, this paper presents the design and implementation of an error detection system based on ontology and recurrent neural networks (RNNs). The system uses sensor and controller data generated during smart farming and implements an architecture to detect and diagnose the malfunction of sensors and controllers in the smart farming system.

The remainder of this paper is arranged as follows. In Section 2, an overview of previous studies on sensor-based fault detection is presented. In Section 3, the architecture and composition of the proposed system to detect faults in parts constituting the smart farming system are discussed. The system design of an AI model used for experimental verification is presented in Section 4. Subsequently, the model implementation and experimental analysis are presented in Section 5. Finally, Section 6 presents the conclusions of the study.

2. Materials and Methods

In this context, an AI algorithm and an ontological technique are developed to enhance model generalizability based on data type. The overall aim is to identify and predict malfunctions in smart farm ICT equipment by evaluating both internal and external control data, unlike the current prediction approach based solely on temperature and humidity data.

2.1. RNNs

RNNs are a type of artificial neural network (ANN) comprising a directed cycle composed of hidden nodes connected by directed edges. They have garnered significant attention recently, alongside convolutional neural networks (CNNs), owing to their suitability for processing sequential sensor data, such as voice and textual data [11].

As illustrated in Figure 1, RNNs can flexibly create various structures, which are capable of accepting inputs of any length, by manipulating their network architecture.

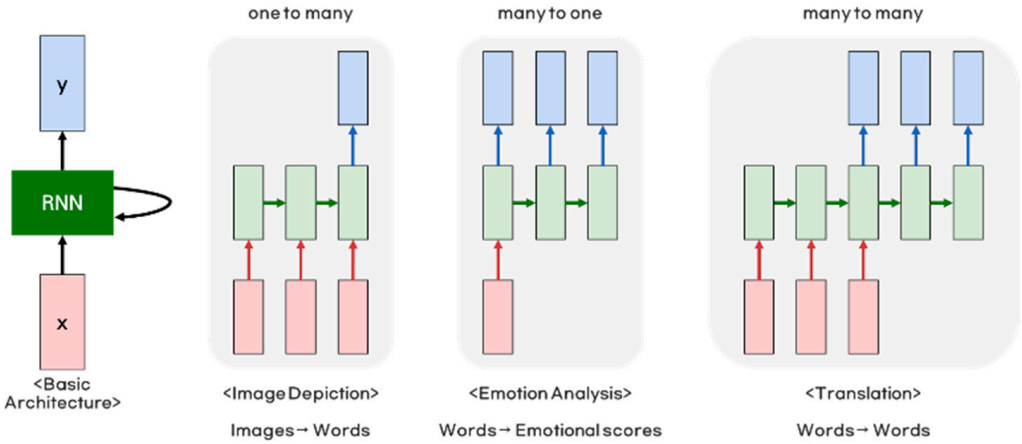


Figure 1. Architecture of the RNN algorithm ("one-to-many", "many-to-one", "many-to-many") [12].

Figure 2 depicts the basic architecture of RNNs. The green, red, and blue boxes represent the hidden states (h), input (x), and output (y), respectively. The current hidden state (h_t) is updated based on the previous hidden state (h_{t-1}). The current output (y_t) is updated by using h_t based on the equation given in the figure. The activation function of the hidden state is taken to be the nonlinear hyperbolic tangent function (tanh) [13–19].

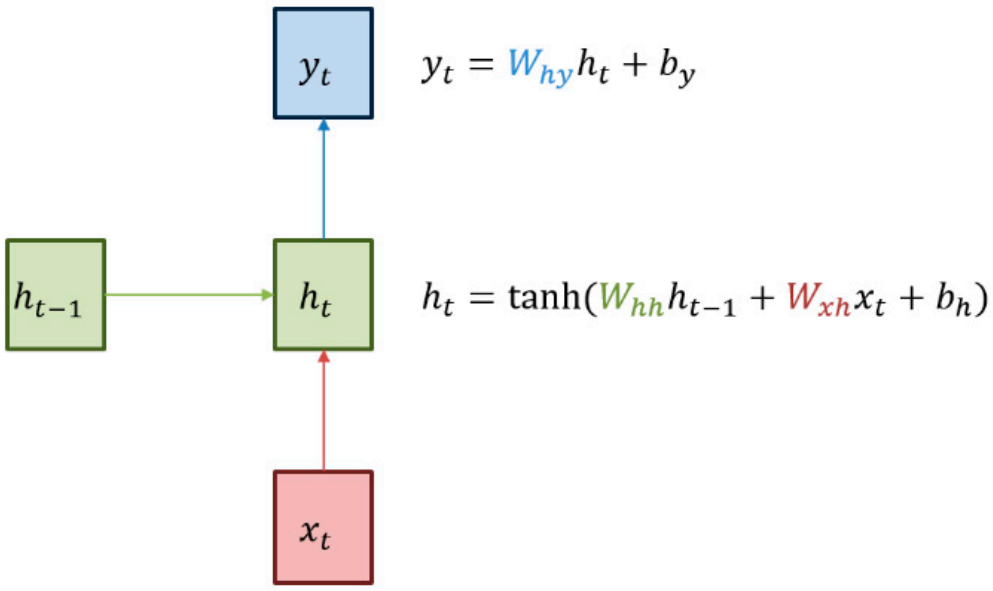


Figure 2. Basic architecture of RNNs [12].

2.2. Smart Farming Systems

Smart farming is defined as a convergence technology that incorporates ICT into existing agricultural, livestock, and fishery industries to improve their productivity. Smart farming enables the measurement and analysis of temperature, humidity, and sunlight using ICT and the remote control of the environment using mobile devices. In the smart farming system depicted in Figure 3, smart farm operations consist of defining the growth conditions such as temperature, humidity, and CO_2 level using growth environment maintenance/management software, and monitoring the growth environment by automatically collecting data related to temperature, humidity, solar radiation, and CO_2 levels. Moreover, the system enables convenient management of the environment, e.g., the automatic/remote operation of HVAC, window opening and closing, and the supply of CO_2 and nutrient feed. However, faults in ICT equipment can cause considerable damage to farms, making an effective fault diagnosis system essential [20–23].

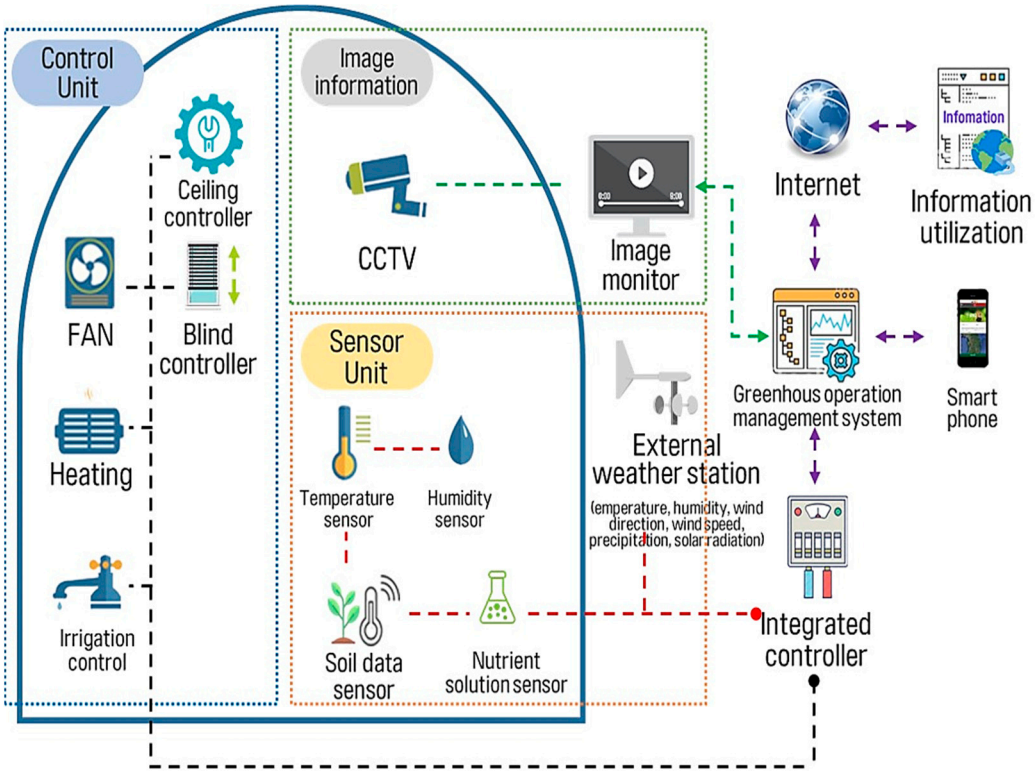


Figure 3. Smart farming architecture.

3. System Configuration

A testbed installed at a farm was used to implement the proposed fault detection system. The entire system constituting the smart farm comprises the cloud in charge of the IoT equipment installed at the smart farm, communication and control, system management, and the machine-learning- and deep-learning-based common framework constituting the fault diagnosis engine.

As depicted in Figure 4, the common framework for fault diagnosis consists of a framework to perform ontological and deep-learning-based fault diagnosis based on equipment thresholds, conditions, actuator conditions, and user-defined rules.

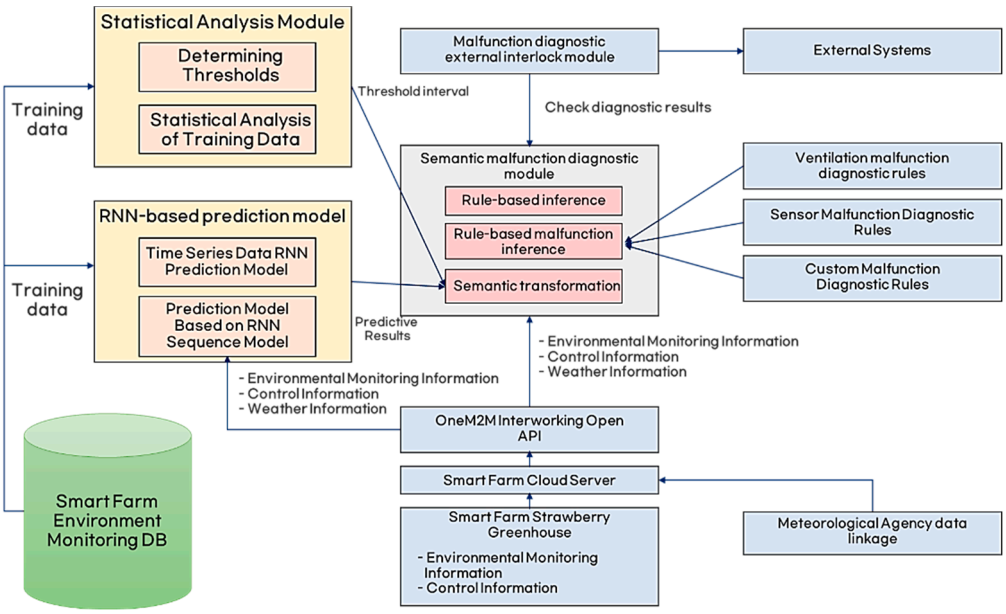


Figure 4. Block diagram of the fault diagnosis system.

Moreover, actual data are transmitted along the paths illustrated in Figure 5 within an integration module connecting the common interfaces to integrate the sensing data with the actuator control data received from the OneM2M Integration Open API cloud [24].

Figure 5. Data transfer paths of the actuator control interface integration module.

The statistical analysis module is a statistical value calculation module that diagnoses faults based on the difference between the measured value and the value predicted by the RNN-based prediction model. Time series, i.e., the RNN prediction model and the RNN sequence prediction model, are trained, and sensing values are predicted based on series of incoming sensing and control values.

The semantic fault diagnosis module consists of smart farm ontology, time ontology, geospatial ontology, event ontology, and fault diagnosis ontology. Fault diagnosis ontology is used to define the concepts of threshold and prediction data used to assess the faults detected by the statistical analysis module. Using the notification interface provided by the cloud, malfunction notifications are classified as faults and the results of fault diagnosis are notified to the user.

Figure 6 illustrates the software architecture, which consists of the implementation environment, a fault diagnosis engine layer, an interface layer, and a user interface layer.

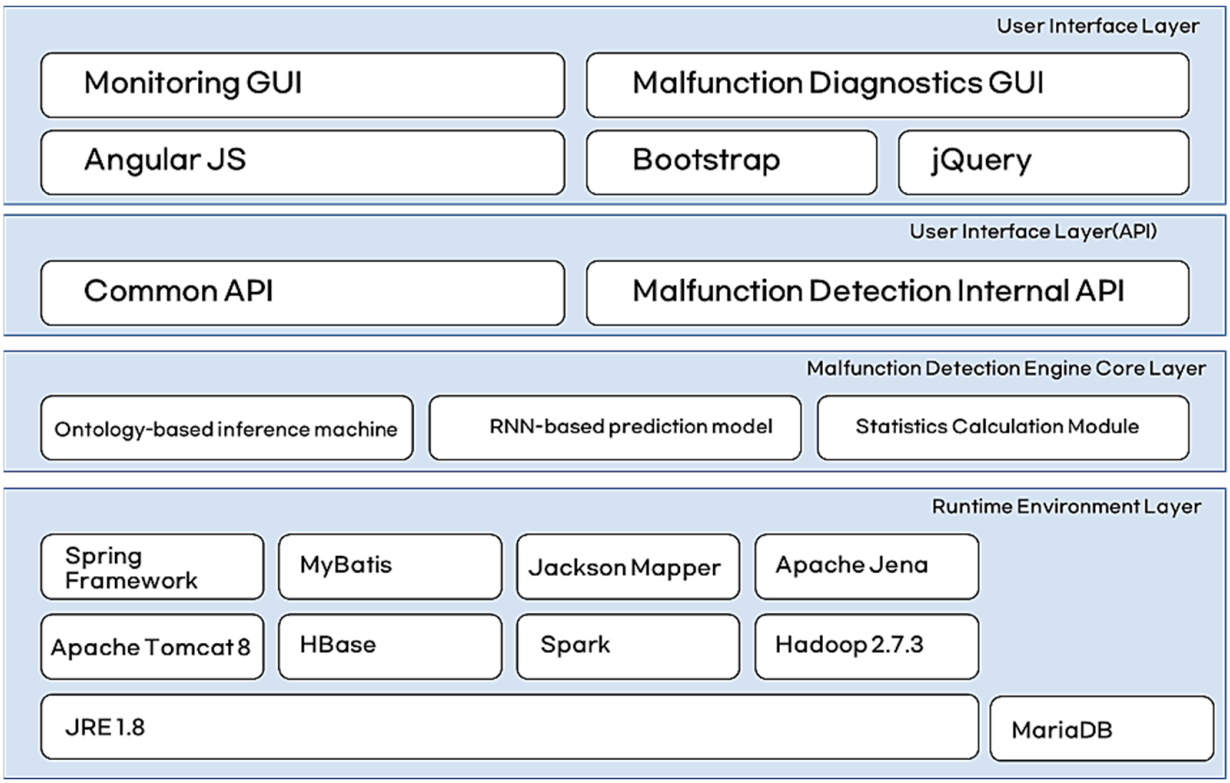


Figure 6. Software architecture.

Figure 7 illustrates the integration architecture within the fault diagnosis prediction system. Equipment control data and equipment-measured values transmitted through the common API and fault API are collected by the fault diagnosis module in real time via the HTTP RESTful API, stored in a queue for a certain length of time, and then diagnosed via the scheduler job.

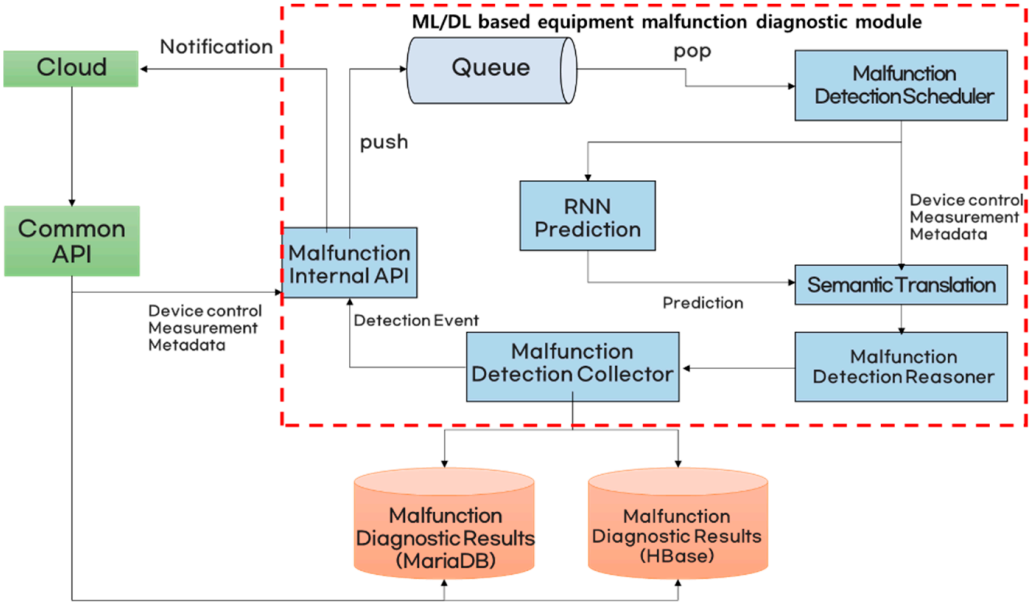


Figure 7. Integration architecture within the fault diagnosis prediction system.

To perform RNN prediction using the data collected by the scheduler job and add the RNN prediction results and the collected data source to the ontology, conversion-to-triples is performed during the semantic transformation process, and the converted triples are added to the semantic storage.

Inference rules registered in advance for ontology are applied to perform fault diagnosis as well as store and manage the inferred fault events in cloud notifications and RDBMS/HBase. Equipment fault diagnosis results are provided directly to RDBMS and HBase when requested in the fault API.

4. Model Design

The ontology for fault diagnosis is defined to enable fault diagnosis based on knowledge. The main environmental data for the smart farm are collected by the sensor network; thus, the ontology is defined based on the semantic sensor network ontology [25–29].

Figure 8 presents the ontology architecture for smart farm fault diagnosis. The semantic sensor network ontology is designed for general use, and includes the detection target of the sensor, the detection method, metadata, sensors, the sensor deployment system, and various attributes. However, its direct use for fault diagnosis is inefficient due to its complexity. Therefore, the smart farm sensor network is defined based on the stimulus–sensor–observation pattern and other patterns related to sensors and the sensor deployment system.

The W3C time ontology is used to describe the concept of time. For smart farm equipment deployment, a spatial concept suitable for smart farming is defined based on the open-source spatial data ontology.

Table 1 defines the concepts related to the fault diagnosis event and the conceptual relationships around the event.

Fault diagnosis methods are classified as (i) methods based on smart farm ontology and rules and (ii) fault event generation methods that use the ontology based on thresholds obtained by first analyzing the normal distribution of the difference between the sensor sequence values transmitted in real time using the RNN prediction model (abs (predicted value − measured value)) and the values generated during model training (abs (predicted value − measured value)) and, then, comparing these two values using the rules listed below.

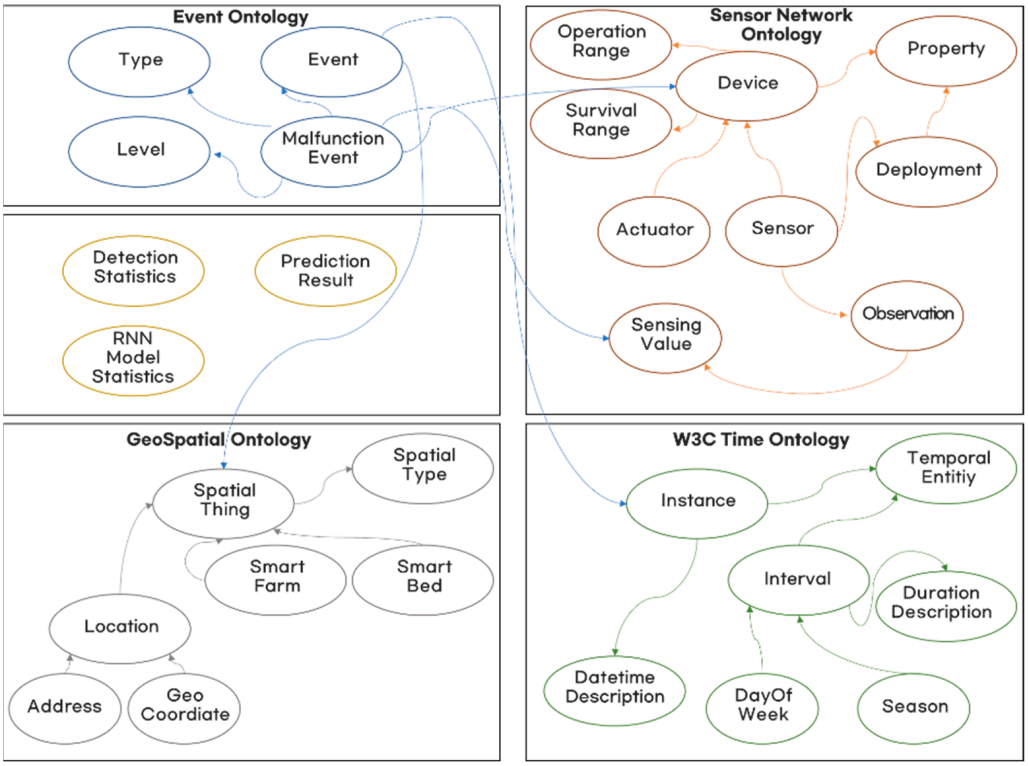


Figure 8. Ontology architecture for smart farm fault diagnosis.

Table 1. Concepts related to the fault diagnosis event.

Ontology	Description
Event Ontology	Event ontology defines the concept of an "event" representing the fault diagnosis result. Events are derived by ontological semantic transformation of the modeled equipment composition data and measurement values; storage is in the form of Resource Description Framework triples and user rule-based inference.
Detection Ontology	The sensing values exhibit specific patterns depending on various environmental factors, which enables the use of normal time series sensing values as training data to train the RNN model for the estimation of sensing values.
	To assess values exceeding a threshold as faults by comparing the predicted and measured values, it is necessary to define the concepts that the predicted values represent.
	Moreover, based on the predicted values, the concepts necessary for fault diagnosis are defined. For fault diagnosis, it is important to configure the threshold based on statistical analysis of normal data. Detection ontology is used to determine the threshold value for each equipment type.
Sensor Network Ontology	The sensor network ontology is composed of the concepts that represent the equipment used for smart farming (sensors, actuators, IoT nodes, network equipment, etc.) and their respective measurement values.
	The basic pattern comprises ontologies that are suitable for smart farming and based on the semantic sensor network ontology.
Geospatial Ontology	Geospatial ontology comprises spaces where smart farming equipment is installed and operated, e.g., farms, greenhouses, and zones. These spatial concepts share structural relationships.
Time Ontology	OWL-Time represents the OWL-2 DL ontology for the concept of time and is used to describe the temporal attributes of resources.
	Time ontology provides the words and phrases required to express topological (sequencing) relationships between moments and intervals related to temporal locations, including information about duration, date, and time.
	Temporal location and duration can be expressed using the traditional (Gregorian) calendar and clock, or other time reference systems, such as Unix time, geological time, or other types of calendars.

A fault event is added in the following cases:

Rule 1: The temperature sensing values obtained from the equipment differ from those obtained from other equipment of the same type installed in the same area.

Rule 2: A temperature sensor and a heater are adjacent among the equipment installed in the same area.

Rule 3: Temperature and humidity sensors are installed in the same area and the current temperature is 5 °C or less and the humidity is 10% or less.

Rule 4: The power measured by a power sensor is zero during the operation of some equipment.

Rule 5: When both the RNN-predicted value of an installed device and Many2OneModelStatistics for the device exist, a malfunction event is added if the VARIANCE of the RNN-predicted value is greater than the VARIANCE threshold of the Many2OneModelStatistics.

Figure 9 shows a hierarchical ontology configuration model, wherein the fault diagnosis processing flow involves a cyclic query of collected data via semantic transformation and their conversion into triples for the input (the optimal batch cycle is configured considering the quasi-real-time performance and the sensor collection cycle). The first request made after the data input is inferred automatically, and the inferred triples are extracted by transmitting the query to SparQL.

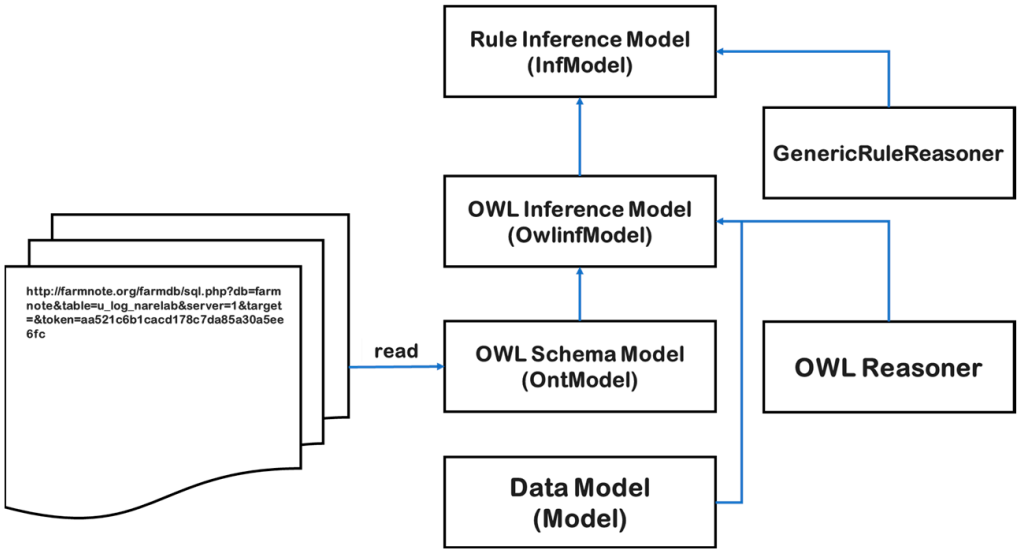


Figure 9. Hierarchical ontology configuration model. (http://farmnote.org/farmdb/sql.php?db=farmnote&table=u_log_narelab&server=1&target=&token=aa521c6b1cacd178c7da85a30a5ee6fc) (accessed on 11 June 2023).

The extracted inferred data are transferred to other integrated subsystems or stored in a database. Finally, all triples of the data model are deleted, as are all inferred triples, by rebinding the upper models (OWL inference model, rule inference model).

Table 2 provides an overview of the methods used to configure the fault detection inference rules. Inference rules for detecting an anomaly state of a sensor in a smart farm can be derived by applying these methods.

First, the sensing value is predicted using the RNN Many2One and RNN Seq2Seq models and compared with the measured value. If the difference between the predicted and measured values exceeds the pre-determined threshold, the event is considered a fault event. The threshold used at this time is determined by comparing the measured data used during model training with the values predicted by the RNN model.

Second, when the difference between the sensing values of sensors of identical type (e.g., temperature sensors) installed in the same zone/area exceeds the threshold (5 °C), the sensors in the zone/area are suspected to be faulty. As it is not possible to determine the exact faulty sensor, all sensors are deemed to be faulty.

Table 2. Methods for configuring the fault detection inference rules.

Category	Measure	Notes
Single-equipment (sensor) fault detection	Threshold exceeded	Single value; processing in real time.
	Improperly installed (location/process)	Metadata regarding sensor installation required (e.g., temperature sensors to be installed near ventilator and radiator at an appropriate distance).
	Sensor expired	Sensor expiration date metadata required.
	Disturbed communication and data collection	Storage and management of last collection date required.
Multi-equipment (sensor) fault detection	Detection by observing inter-sensor correlations	Collection of all correlated sensing values required (accumulation for a certain period of time before deployment) (e.g., high temperature and humidity).
	Detection by observing the difference from the nearest sensing value.	Occurrence of a difference exceeding the threshold of a given sensor.
Situational fault detection	Controller fault detection in an improper environment during the growth stage.	Data on growth stage required. Occurrence of an event in a suspected controller type, not a specific controller fault.
	Controller fault detection when the sensing value does not change for a long time during controller operation.	Controller activation data required.

Third, when high-temperature, high-humidity conditions beyond a certain threshold are different from the general conditions of high temperature and low humidity, which exerts a harmful effect on plant growth at smart farms, the sensor concerned is suspected to be faulty.

Finally, the minimum and maximum allowable ranges are checked as metadata of the measured values of a sensor to determine whether the sensor is in an environment where it can operate normally and reliably. The range may fall within the normal range, enabling normal operation (a range measurable by the sensor or ideal value to be specified in the normal plant factory environment). If the range checked falls outside the normal range, the corresponding sensors are deemed to be faulty. Table 3 provides an overview of the inference rules to be applied in different conditions.

Table 3. Inference rules to be applied in different conditions.

Rule ID	Description
rule_rnn_many2one	Anomaly state detection using the RNN Many2One model
rule_rnn_seq2seq	Anomaly state detection using the RNN Seq2Seq model
rule1	Fault inference based on a comparison with the threshold of a single sensor
rule2	Fault inference based on the correlations of two or more sensing values, which yields two events
rule3	Actuator fault inference based on actuator control data and power sensing values

5. Model Implementation and Experimentation

Diagnosis of malfunction using the above-defined ontology requires the consideration of the critical point of the smart farm sensor. As a result, if the sensor value is predicted to be abnormal and deviates significantly from the predicted value, it may be diagnosed as a malfunction.

Sensor value prediction was performed based on hourly data collected from a strawberry farm spanning one year obtained from the Korea Agency of Education, Promotion,

and Information Service in Food, Agriculture, Forestry and Fishery. Data analysis reveals that various factors affect the temperature sensing values, as outlined in Table 4.

Table 4. Baseline training datasets [30].

Category	Datasets
Farm	Strawberry farm.
Environment	Indoor environment, outdoor environment, hydroponic environment, soil environment.
Sensing value	Temperature, humidity, CO_2, surface temperature, humidity, solar radiation, wind speed, precipitation (dry/wet).

The measurement items include measurement season, measurement time, outdoor temperature, solar radiation, clouds, fog, precipitation (dry or wet days), and amount of precipitation. The measurable and replaceable data applicable to a real prediction model are listed in Table 5.

Table 5. Training data affecting the temperature values.

Measurement Items	Alternative Items
Day (ref. 365 days)	Measurement season
Measurement time	Measurement time
Outdoor temperature	Outdoor temperature
Solar radiation	Solar radiation, clouds, fog
Precipitation	Dry day/wet day, amount of precipitation

As the data collected by the Korea Agency of Education, Promotion, and Information Service in Food, Agriculture, Forestry and Fishery provide information about the aforementioned items and temperature measurement values corresponding to each time slot, the data were extracted by sequence and item, and the training data were pre-processed. These datasets were used to train the many-to-one model by dividing the time series input sensing values into sequences. The errors corresponding to the differences between the predicted and measured values calculated during training followed a normal distribution, and a threshold was added (semantic transformation) to the ontology by setting appropriate confidence intervals. This value was input (semantic transformation) into the ontology, and faults were assessed by comparing it with the threshold value calculated by the model following the rules.

A system-wide configuration of constants and variables was applied to the RNN learning model, which was normalized and denormalized via MinMaxScaler and RevMinMaxScaler functions, respectively. The training and test datasets were separated by loading the farm file with the loadData function.

RNN cell/multi-RNN cell were defined to configure the RNN network after loading data from the "temp" directory corresponding to each farm into the main code, and a fully connected layer was defined to test the learning result. Based on the definition of the cost function, the cost was set to be minimized using the Adam optimizer, and several training iterations were performed by the (training) node following the system-wide configuration. Finally, the learning results were stored, and the learning model was tested using the farm data stored for testing.

Algorithm 1 presents the code responsible for calculating the predicted values based on an actual sensor sequence. It specifically focuses on the section that displays the outcomes of training the recurrent neural network (RNN) cell in the program.

In the main code, the code first examines the command line arguments and the file path for input data. It loads the data from the specified file. Afterwards, the RNN Cell/Multi-RNN cell is defined, the RNN network is configured, the fully connected layer is defined, the node is run, and learning is conducted as many times as globally set. Finally, the results are output as a graph.

Algorithm 1 Prediction algorithm based on sensor sequence

```
# ------------------------------------------------------------
# Program Start
# ------------------------------------------------------------
# Load training data
loadData()

print('Size of training data: ' + str(len(trainX)))
print('Size of test data: ' + str(len(inputX))

# Input placeholders
X = tf.placeholder(tf.float32, [None, seq_length, data_dim])
Y = tf.placeholder(tf.float32, [None, 1])

# Construct the RNN network
# RNN Cell (Available cells: Basic LSTM, LSTM, GRU, ...)
# cell = tf.contrib.rnn.BasicLSTMCell(
#        num_units=hidden_dim, state_is_tuple=True, activation=tf.tanh)
cell = tf.contrib.rnn.GRUCell(
num_units=hidden_dim, activation=tf.tanh)

# Multi-RNN Cells
cells = tf.contrib.rnn.MultiRNNCell([cell] * NUMBER_OF_RNN_CELL_LAYERS)

# Dynamic RNN (outputs: output, _states: previous states in the RNN network)
# If RNN cells and input data are given as arguments, the RNN cells are connected to form a network.
outputs, _states = tf.nn.dynamic_rnn(cell, X, dtype=tf.float32)

# Add fully connected layers to obtain prediction values
Y_pred = tf.contrib.layers.fully_connected(
outputs[:, -1], output_dim, activation_fn=None)

# Define the cost function (sum of the squares)
loss = tf.reduce_sum(tf.square(Y_pred - Y))

# Define the cost Tensor for Tensorboard
tf.summary.scalar("cost", loss)

# Summary
summary = tf.summary.merge_all()

# Define the optimizer
optimizer = tf.train.AdamOptimizer(learning_rate)
train = optimizer.minimize(loss)

# RMSE (Root Mean Square Error, the square root of the mean squared differences between actual and predicted values)
targets = tf.placeholder(tf.float32, [None, 1])
predictions = tf.placeholder(tf.float32, [None, 1])
rmse = tf.sqrt(tf.reduce_mean(tf.square(targets - predictions))

with tf.Session() as sess:
init = tf.global_variables_initializer()
sess.run(init)

# Create a summary writer
writer = tf.summary.FileWriter(TB_SUMMARY_DIR)
```

Algorithm 1 *Cont.*

```
writer.add_graph(sess.graph)
global_step = 0

# ----------------------------------------------------------------
# Training Phase
# ----------------------------------------------------------------
for i in range(iterations):
s, _, step_loss = sess.run([summary, train, loss], feed_dict={
X: trainX, Y: trainY})
#print("[step: {}] loss: {}".format(i, step_loss))

writer.add_summary(s, global_step=global_step)
global_step += 1

# Save the training results
saver = tf.train.Saver()
saver.save(sess, './model/malfunction_predict.pd')

# ----------------------------------------------------------------
# Testing Phase
# ----------------------------------------------------------------
# Perform predictions on test data and display the results using plots
test_predict = sess.run(Y_pred, feed_dict={X: inputX})
rmse_val = sess.run(rmse, feed_dict={
targets: sensingValueY, predictions: test_predict})
print("inputX RMSE: {}".format(rmse_val))

correct_prediction = test_predict - sensingValueY
accuracy = tf.reduce_mean(correct_prediction)

plt.figure(figsize=(20, 4))
plt.plot(RevMinMaxScaler(sensingValueY), 'b-', label='Sensing')
plt.plot(RevMinMaxScaler(test_predict), 'r-', label='Prediction')
plt.xlabel("Time Period")
plt.ylabel("Temperature")
plt.legend(loc='best')

# Perform predictions on the first test data and display the results using plots
test_predict = sess.run(Y_pred, feed_dict={X: firstTestX})
rmse_val = sess.run(rmse, feed_dict={
targets: firstTestY, predictions: test_predict})
print("firstTestX RMSE: {}".format(rmse_val))

correct prediction = test_predict - firstTestY
accuracy = tf.reduce_mean(correct_prediction)

plt.figure(figsize=(20, 6))
plt.plot(RevMinMaxScaler(firstTestY), 'b-', label='Sensing')
plt plot(RevMinMaxScaler(test_predict), 'r-', label='Prediction')
plt.xlabel("Time Period")
plt.ylabel("Temperature")
plt.legend(loc='best')

plt.show()
```

Figure 10 graphically depicts the tensor board cost with respect to the sensor values. The test was conducted by varying the number of training iterations from 500 to 1000, 5000, 10,000, and 100,000. After 5000 training iterations, no significant difference is observed.

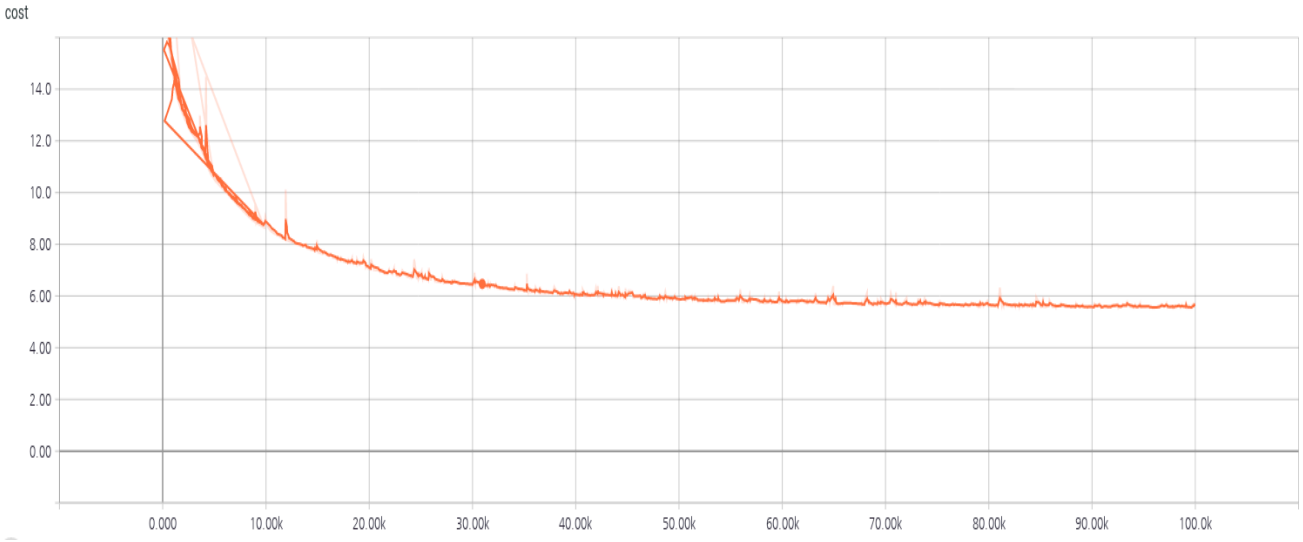


Figure 10. Tensor board cost graph.

The root mean squared error (RMSE), a widely used error metric, was used to compare the learning results. In the error measurement results listed in Table 6, the RNN cell exhibits an RSME value for the LSTM that is lower than that for the GRU by 0.003. The training data are time series data; thus, this can imply that the LSTM algorithm analyzes time series data more effectively than GRU.

Table 6. Errors corresponding to each learning model.

RNN Cell	RSME
LSTM	0.0732
GRU	0.0760

Table 7 presents the test results corresponding to different numbers of training iterations. The lowest mean error is achieved after 5000 iterations, and the mean error increases as the number of training iterations is increased to 10,000 and 100,000, presumably due to overfitting.

Table 7. Errors corresponding to different numbers of training iterations.

Number of Training Iterations	RSME
500	0.0806
1000	0.0785
5000	0.0784
10,000	0.0768
100,000	0.0928

Concrete accuracy is expressed numerically, as in Table 7; thus, it is difficult to determine the accuracy of these values. Figures 11–14 visually express the prediction accuracy by comparing actual data and predicted data, and it can be seen that the values predicted by the model follow the actual sensor values relatively well. A real-time smart farm equipment fault diagnosis experiment was conducted using the prediction model derived in this study.

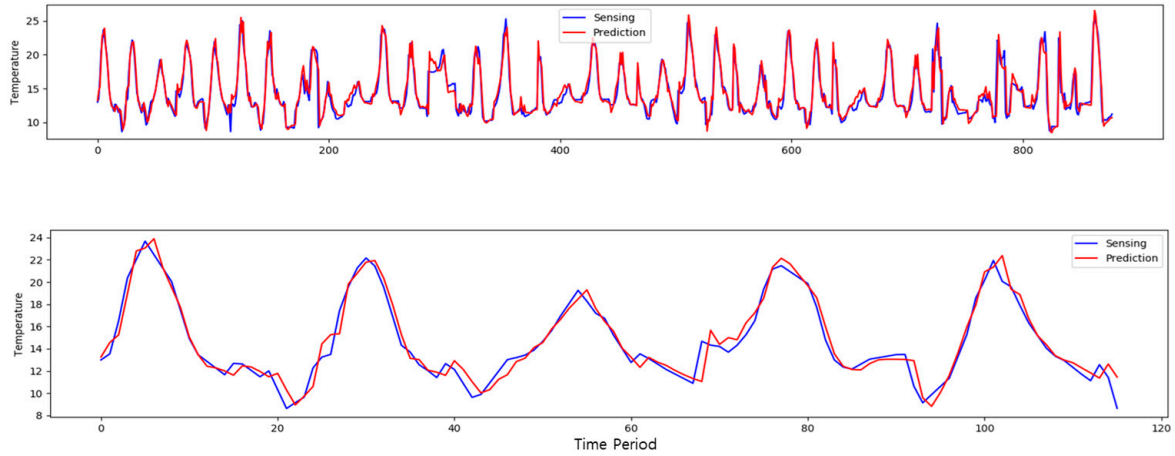
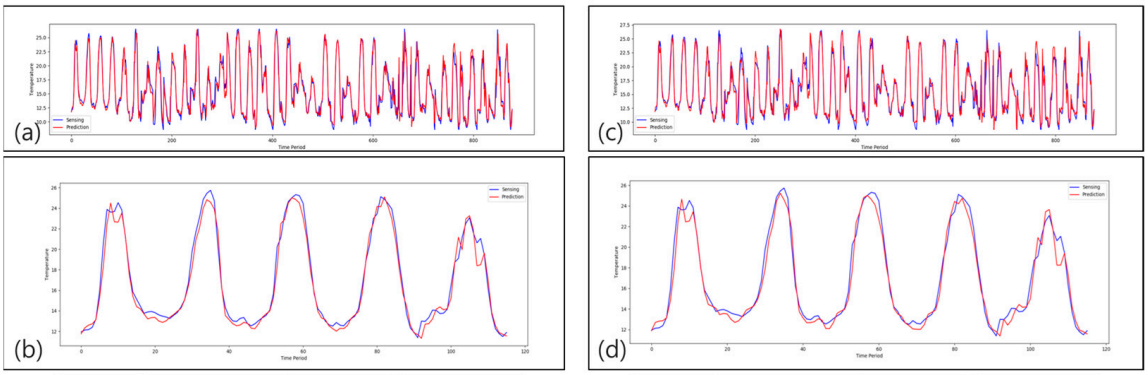


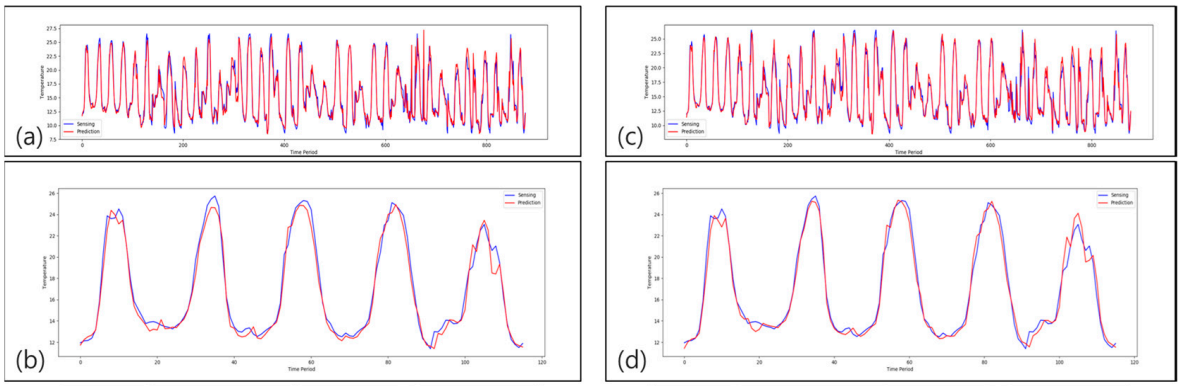
Figure 11. Test results obtained using the measured data (**top**: overall result value, **bottom**: magnified graph of some test results).



Parameter	Value
RNN Cell	LSTM
Sequence Length	5
Hidden Dimension	10
iterations	500

Parameter	Value
RNN Cell	GRU
Sequence Length	5
Hidden Dimension	10
iterations	500

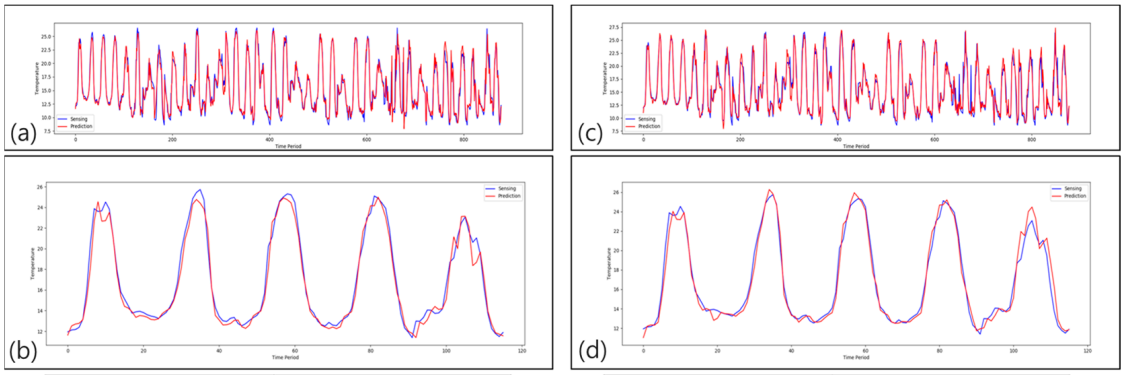
Figure 12. (**Left**) LSTM model, 500 iterations; (**Right**) GRU model, 500 iterations. (a) LSTM model overall results, (b) LSTM model Enlarged graph, (**c**) GRU model Overall results, (**d**) GRU model Enlarged graph.



Parameter	Value
RNN Cell	GRU
Sequence Length	5
Hidden Dimension	3
iterations	5000

Parameter	Value
RNN Cell	GRU
Sequence Length	5
Hidden Dimension	10
iterations	5000

Figure 13. (**Left**) LSTM model, 5000 iterations; (**Right**) GRU model, 5000 iterations. (a) LSTM model overall results, (b) LSTM model Enlarged graph, (**c**) GRU model Overall results, (**d**) GRU model Enlarged graph.



Parameter	Value
RNN Cell	GRU
Sequence Length	5
Hidden Dimension	5
iterations	1000

Parameter	Value
RNN Cell	GRU
Sequence Length	5
Hidden Dimension	5
iterations	10000

Figure 14. (**Left**) LSTM model, 1000 iterations; (**Right**) GRU model, 10,000 iterations. (a) LSTM model overall results, (b) LSTM model Enlarged graph, (**c**) GRU model Overall results, (**d**) GRU model Enlarged graph.

The observed RMSE value in the experiment is 0.062557. The graphs plotting the measured (blue) and predicted values (red) in Figure 11 are quasi-identical, indicating that the prediction model exhibits good performance.

Figure 15 depicts the smart farm equipment fault diagnosis test environment where the proposed model was evaluated experimentally. The equipment fault API integration function, equipment fault diagnosis unit function, power measurement equipment integration, and RNN model were evaluated. The sensor used in this experiment was a temperature sensor(Naretrends Co., Ltd., Buchon-si, Korea) used in smart farms.

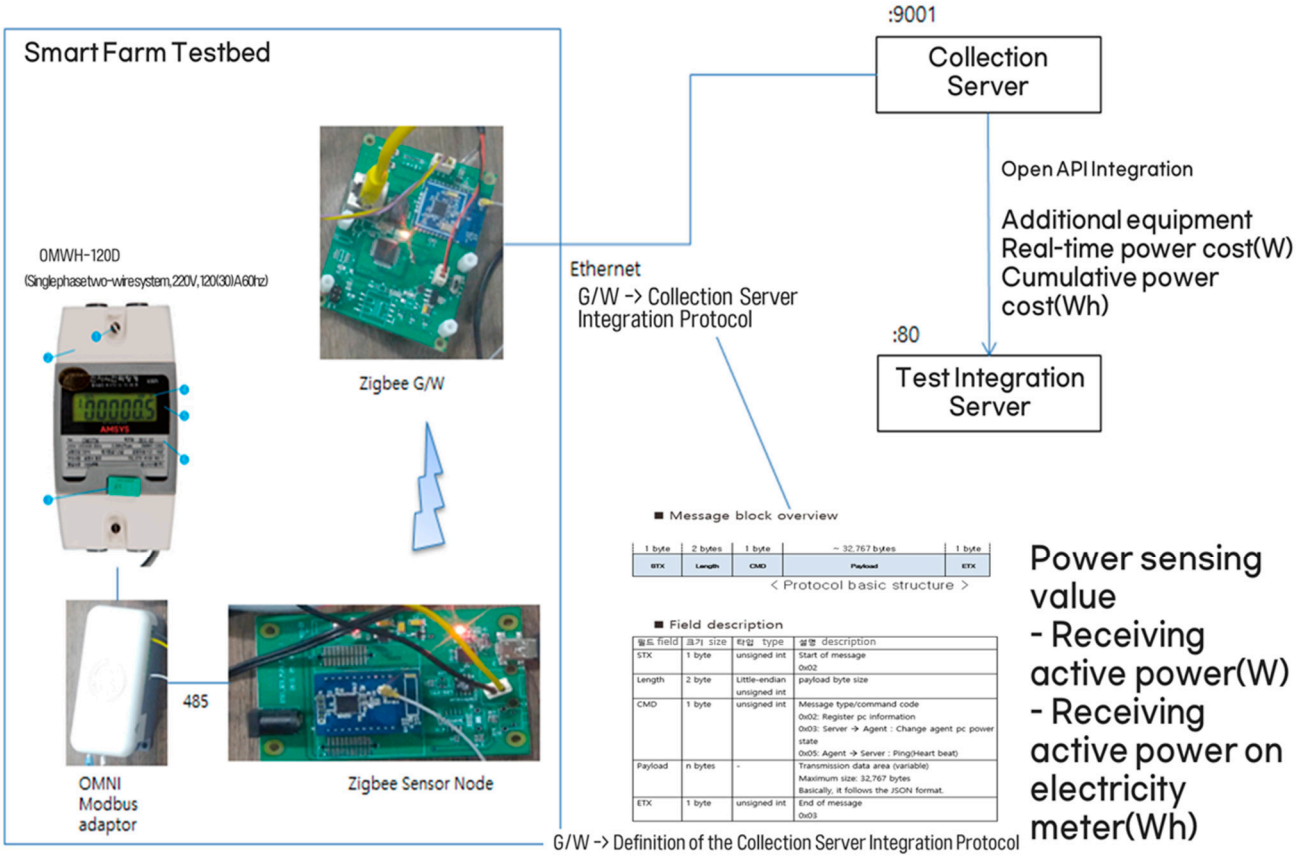


Figure 15. Configuration of the sensor node test environment.

Figure 16 shows a block diagram of the RNN-based fault diagnosis workflow. If fault diagnosis is conducted based on the threshold value after RNN-based prediction of sensor data, there is an integrated interface to submit external data. This is performed using the method "receiveDeviceMeasureNotification", and the path indicated is its URL path. The data received are not directly entered into the fault diagnosis ontology but are sent to the "Malfunction Detection Scheduler" queue. Fault detection inference can be conducted only after different types of sensor values have been received for a certain length of time. Subsequently, the semantic storage, which is loaded with ontology models, reasoners, and rules, is managed. Once instances are added to the semantic storage through the process of semantic transformation, an event query automatically leads to fault inference. In other words, a query triggers inference. Thus, adding triples does not automatically lead to inference, but inference starts as the need arises. A semantic transformation method such as "translatePredictionResult" is applied using a semantic transformer. The object "PredictionResult" is received as a factor and is transformed into a triple consisting of a subject, a predicate, and an object, which is then added to the ontology using the function "semanticStorageManager.addTriple". Then, the sensor values cyclically collected through the external interface stored in the "Malfunction Detection Scheduler" queue are subjected to semantic transformations, and fault event reporting is carried out through the process of query and fault inference. Here, "doCollect" issues an event query and triggers inference in the ontology, and the Cloud sends fault events to the website.

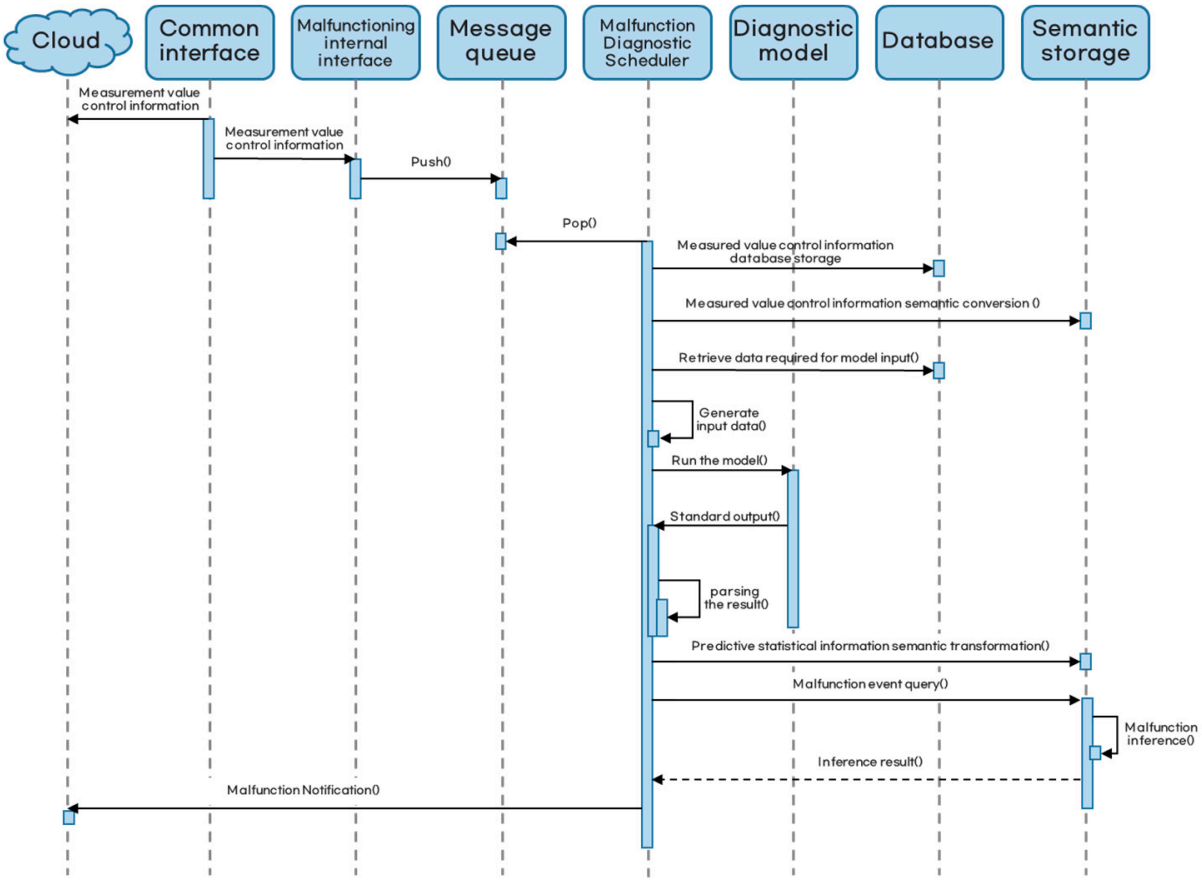


Figure 16. Block diagram of the RNN-based fault diagnosis workflow.

Figure 17 shows the RNN-based fault diagnosis program. The sensor values are predicted using the prediction model, the predicted data are subjected to ontology inference, and the diagnosis results are presented to the user. Once the greenhouse is selected as the area and test data are entered, the predicted and measured values are presented via real-time monitoring to notify the user of threshold crossings.

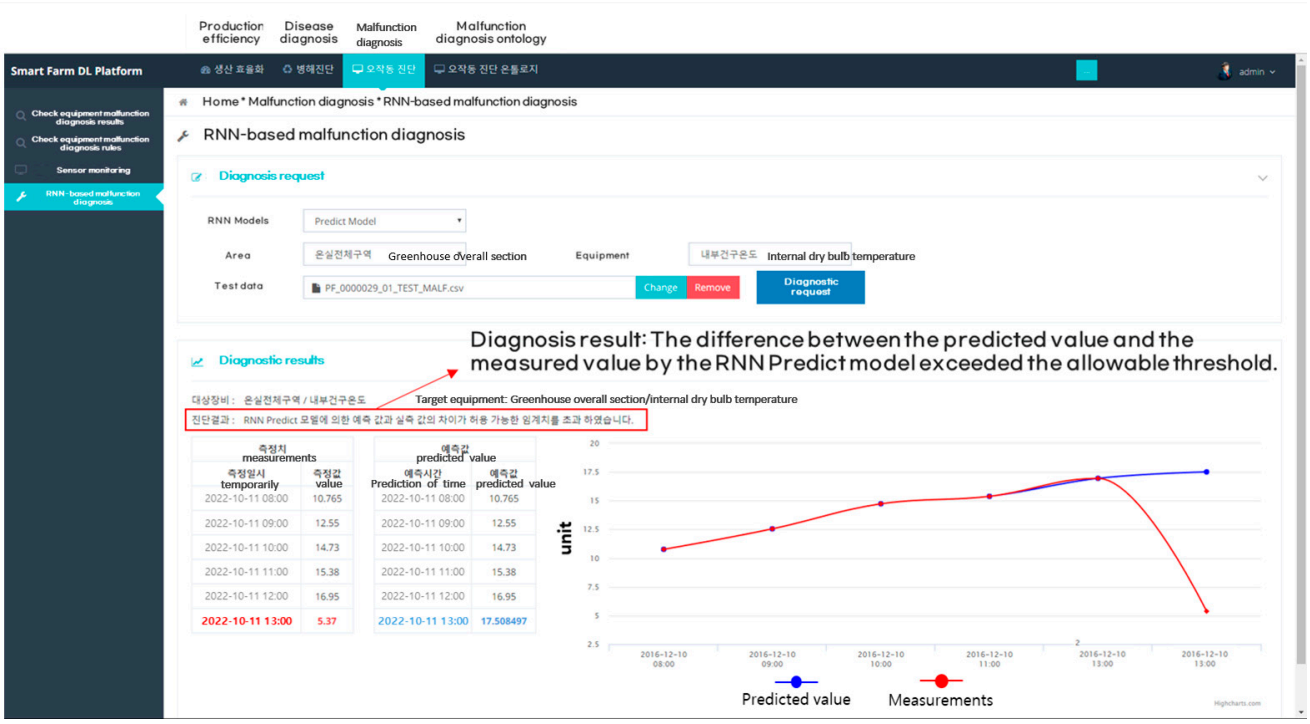


Figure 17. RNN-based fault diagnosis program interface.

The proposed system and previously provided fault detection systems have long been used in other fields, and accurate comparisons between them are difficult because they employ different methods. However, the continuous analysis of sensor outliers detected by the proposed system is expected to improve fault detection in smart farming systems.

6. Conclusions

In this study, a fault detection technique for smart farming equipment was designed and implemented with the aim of preventing damage to farms due to downtime caused by mistakes, faults, and aging of ICT devices. This is particularly significant owing to the widespread use of such devices in agricultural production facilities.

To this end, a model was designed based on RNN algorithms. The model was trained using hourly data obtained from the Korea Agency of Education, Promotion, and Information Service in Food, Agriculture, Forestry and Fishery which were collected at a strawberry farm over a one-year span. The data were extracted in sequence, and necessary items were preprocessed as training data. The learning results were tested by considering 500, 1000, 5000, 10,000, and 100,000 training iterations. The RSME of the optimized model was 0.07, confirming that it exhibits a high prediction power in an environment in which ICT equipment operation is difficult.

In farms operating large-scale, modernized, high-tech greenhouses, growth management optimization with respect to facilities and crop characteristics is essential, and efficient operation of sensors and controllers is fundamental. Further, to ensure effective smart farm operation, human interference should be minimized. Furthermore, damage to smart farm equipment and crops can be minimized by detecting malfunctions and arranging for prompt replacements. The technology discussed in this study is a key element in the construction of smart farms capable of self-reliant operation and fault detection.

Moreover, this study enables the prevention of disputes between farms and companies, empowers device PL insurance through linkage with insurance companies, and can be used as a source of base data suitable for agricultural research.

It is also expected to maximize the output of the smart farm system industry by predicting the remaining useful life and promoting the use of sensors and data to monitor

the status of smart farm equipment or mechanical systems, the use of secure diagnostic technology to detect signs of failure, and the use of condition-based maintenance technology to maintain normal operation. It is expected to enhance the reliability of the service and secure global competitiveness for smart farm companies.

In future work, we intend to predict malfunction faults by collecting vibration, current, and image values obtained using actuators used in greenhouses.

Author Contributions: Writing—original draft, H.O.C.; Project administration, M.-H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Korea Institute of Planning and Evaluation for Technology in Food, Agriculture and Forestry (IPET) and the Korea Smart Farm R&D Foundation (KosFarm) through the Smart Farm Innovation Technology Development Program funded by the Ministry of Agriculture, Food and Rural Affairs (MAFRA) and the Ministry of Science and ICT (MSIT), Rural Development Administration (RDA) (421021-03). This work was supported by Korea Institute of Planning and Evaluation for Technology in Food, Agriculture and Forestry (IPET) through Short-term Advancement of Smart Agricultural Technology in Open Fields Project, funded by Ministry of Agriculture, Food and Rural Affairs (MAFRA) (322031-03).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the de-cision to publish the results.

References

1. Kaaya, J. Role of Information Technology in Agriculture. *Proc. FOA Conf.* **1999**, *4*, 315–328.
2. Sharma, A.B.; Golubchik, L.; Govindan, R. Sensor Faults: Detection Methods and Prevalence in Real-World Datasets. *ACM Trans. Sens. Netw.* **2010**, *6*, 23. [CrossRef]
3. Mourad, M.; Bertrand-Krajewski, J.L. A Method for Automatic Validation of Long Time Series of Data in Urban Hydrology. *Water Sci. Technol.* **2002**, *45*, 263–270. [CrossRef] [PubMed]
4. Ahn, H.; An, S.-Y.; Kim, J.-Y.; Lee, C.W. Humidity Sensor Prediction Model Using Environmental Data Analysis in Greenhouse Smartfarm. In Proceedings of the Symposium of the Korean Institute of Communications and Information Sciences, Jeju, Republic of Korea, 18–21 June 2021; pp. 159–160.
5. Lee, S. Cloud-Based Smart Farm Technology. *J. Korean Inst. Commun. Sci.* **2016**, *34*, 51–57.
6. Jeffery, S.R.; Alonso, G.; Franklin, M.J.; Hong, W.; Widom, J. Declarative Support for Sensor Data Cleaning, Lecture Notes in Computer Science. In Proceedings of the of 4th International Conference on Pervasive Computing, Dublin, Ireland, 7–10 May 2006; pp. 83–100. [CrossRef]
7. Ni, K.; Ramanathan, N.; Chehade, M.N.H.; Balzano, L.; Nair, S.; Zahedi, S.; Kohler, E.; Pottie, G.; Hansen, M.; Srivastava, M. Sensor Network Data Fault Types. *ACM Trans. Sens. Netw.* **2009**, *5*, 25. [CrossRef]
8. Hong, S.-J.; Kim, J.H.; Kim, T.-H. An Integrated Verification System for an RNN Inference Processor. In Proceedings of the Korean Society of Electronics Engineers Conference, Jeju, Republic of Korea, 28 June–2 July 2022; pp. 1918–1921.
9. Kim, Y.-D.; Jung, G.-H. Bayesian Methodology for Machine Learning. *J. Korean Inst. Commun. Sci.* **2016**, *33*, 60–64.
10. Seo, D.-S.; Park, Y.-G.; Park, J.-Y.; Kim, Y.-J. *A Study on Smart Farm Operation Status and Development Direction*; Ministry of Agriculture, Food and Rural Affairs, National Library of Korea: Sejong, Republic of Korea, 2016; Volume 3.
11. Joonyong, K.; Rack, P.K. Analysis of Accuracy and Loss Performance According to Hyperparameter in RNN Model. *J. Converg. Inf.* **2021**, *11*, 31–38.
12. Introduction to the Architecture of Recurrent Neural Networks (RNNs). Available online: https://pub.towardsai.net/introduction-to-the-architecture-of-recurrent-neural-networks-rnns-a277007984b7 (accessed on 11 June 2023).
13. Cho, S.M.; Kim, W. Automatic Document Title Generation with RNN and Reinforcement Learning. *J. Inf. Technol. Appl. Manag.* **2020**, *27*, 49–58.
14. Shin, S.H.; Lee, M.K.; Song, S.K. A Prediction Model for Agricultural Products Price with LSTM Network. *J. Korean Contents Assoc.* **2018**, *18*, 416–429.
15. Kim, D.-H. Livestock Price Prediction Study Using LSTM Based on Big Data. Ph.D. Thesis, Chonbuk National University, Cheongju, Republic of Korea, 2021.
16. Oh, H.-W.; Huh, J.-D. *IoT-Based Smart Factory Failure Prediction Analysis Technology for Productivity and Quality Improvement*; IEEE: Piscataway, NJ, USA, 2020; pp. 33–43.

17. Han, J.H.; Choi, D.-J.; Park, S.-U.; Hong, S.-K. DT-CNN Based Motor Failure Prediction Considering Outlier Data. *J. Inst. Control Robot. Syst.* **2020**, *26*, 932–939. [CrossRef]
18. Ryu, M.; Cha, S.-H. Developing a Knowledge Graph Based on Ontology Learning. *J. Korean Assoc. Comput. Educ.* **2022**, *25*, 51–57.
19. Park, J.; Song, M.; Ahn, S. Developing the Fault Diagnostics and Prognostics Model of a Rotating Machinery. *JKORMS* **2020**, *45*, 25–38. [CrossRef]
20. Lee, S.; Lee, J. Monitoring Procedure of Autocorrelated Processes Using the Deep Learning-Based LSTM Model. *J. Korean Data Inf. Sci. Soc.* **2022**, *33*, 237–248.
21. Elnahrawy, E.; Nath, B. Cleaning and Querying Noisy Sensors. In Proceedings of the of 2nd ACM International Conference on Wireless Sensor Networks and Applications, San Diego, CA, USA, 19 September 2003; pp. 78–87.
22. Tolle, G.; Polastre, J.; Szewczyk, R.; Culler, D.; Turner, N.; Tu, K.; Burgess, S.; Dawson, T.; Buonadonna, P.; Gay, D.; et al. A Macroscope in the Redwoods. In Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, San Diego, CA, USA, 2–4 November 2005; ACM Press: New York, NY, USA, 2005; pp. 51–63.
23. Bae, N.J. *Implementation of Ontology-Based Context-Aware Control Service Model for Plant Factory Environment*; Suncheon University: Suncheon, Republic of Korea, 2014.
24. Park, M.Y.; Kim, M.J.; Park, Y.M.; Song, J.S. Development of Internet of Things Platform Based on oneM2M Standards to Provide Data Augmentation Features for Artificial Intelligence Services. In Proceedings of the Symposium of the Korean Institute of Communications and Information Sciences, Gyeongju, Republic of Korea, 16–18 November 2022; pp. 1683–1684.
25. Ji, K.; Kwon, Y. Hadoop MapReduce Performance Optimization Analysis by Calibrating Hadoop Parameters. *J. Korean Inst. Inf. Technol.* **2021**, *19*, 9–19. [CrossRef]
26. Choi, M.; Gyeong, N.-J.; Lim, J.S. Design of Smart Farming System with Kafka and Spark Streaming. In Proceedings of the Korea Contents Association Comprehensive Conference, Limassol, Cyprus, 7 September 2022; pp. 405–406.
27. Lembo, D.; Stantarelli, V.; Savo, D.F.; Giacomo, G.D. Graphol: A graphical language for ontology modeling equivalent to OWL 2. *Futur. Internet* **2022**, *14*, 78. [CrossRef]
28. Jeon, Y.J.; Lee, H. A Study on the Ontology Modeling by Analyzing RiC-CM v0.2. *J. Korean Rec. Manag. Assoc.* **2020**, *20*, 139–158.
29. Kim, K.Y.; Jongmo, K.; Park, G.-D.; Sohn, M.M. Value of Information Assessment Framework Using Fuzzy Inference Ontology. *J. Korean Soc. Manag. Eng.* **2020**, *25*, 117–135.
30. Smartfarm Datamart. Available online: https://data.smartfarmkorea.net/structuredData/selectContHortiCultureDataViewLists.do (accessed on 11 June 2023).