On Time Series Forecasting Analysis of Soil Moisture by Using Artificial Neural Networks Based - on Rainbow Antimagic Coloring for Autonomous Irrigation System on Horizontal Farming

Dini Mufidati[1], Zainur Rasyid Ridlo[2,3], Slamin[3,4], Ika Nur Maylisa[1], and Dafik[1,3(✉)]

[1] Department of Mathematics Education Postgraduate, University of Jember, Jember, Indonesia
d.dafik@unej.ac.id
[2] Department of Science Education, University of Jember, Jember, Indonesia
zainur.fkip@unej.ac.id
[3] PUI-PT Combinatorics and Graph, CGANT, University of Jember, Jember, Indonesia
s.slamin@unej.ac.id
[4] Department of Computer Science, University of Jember, Jember, Indonesia

Abstract. Precision agriculture is one of the fields that play an important role in improving the social economy, it is due to that many people depend on the agricultural products. One of the supporting factors in agriculture is advancement the precision agriculture under the development of autonomous irrigation. Irrigation is an effort made by humans to irrigate agricultural land. Prediction the soil moisture is one way to help farmers to do watering. We will use an Artificial Neural Networks (ANN) together with the concept of rainbow antimagic coloring to forecast soil moisture for autonomous irrigation system on horizontal farming. The result shows that the best architecture for is obtained by model ANN- cascade forwardnet 566 with an MSE of 4.5127×10^{-19}.

Keywords: Artificial neural network · rainbow antimagic coloring · time series forecasting analysis · soil moisture

1 Introduction

Precision agriculture is one of the fields that play an important role in improving the social economy, it is due to that many people depend on the agricultural products. Horizontal agriculture on the surface of the land is the traditional form of agriculture throughout history, but with the expansion of urban population density and increased demand for land in the cities for more profitable uses, and the lack of resources that are compatible with the nature and requirements of traditional farming practices [7]. For that, the needed for alternative model of agriculture has been emerged, which is the horizontal agriculture is the new model that helped to create green areas within cities to take advantage of those abandoned and inactive areas on the one hand and the lack of sufficient areas for agriculture on the ground on the other.

I. H. Agustin (Ed.): ICONNSMAL 2022, AISR 177, pp. 234–256, 2023.
https://doi.org/10.2991/978-94-6463-174-6_18

Machine learning is a branch of artificial intelligence that employs a variety of statistical, probabilistic and optimization techniques that allows computers to "learn" from past examples and to detect hard-to-discern patterns from large, noisy or complex data sets. This capability is particularly well-suited to time seris forcasting applications, [7]. The utilization of IoT technology equipped with Artificial Intelligence (AI) and Machine Learning (ML) are effective tools in increasing the effectiveness of control management system on horizontal farming.

The Artificial Neural Network (ANN) application for the prediction of microclimates in greenhouses is learned to recognize patterns. These systems are highly appropriate to reflect knowledge that cannot be programmed or justified [1]. An artificial neural network is an information-processing system that has certain performance characteristics in common with biological neural networks. Artificial neural networks have been developed as generalizations of mathematical models of human cognition or neural biology [13]. Machine learns relation about input data with no target data provided. One of the goals learning is to do clustering. Big data is needed to have a good output of the learning. The goal of the reinforcement learning is generating a pattern in simulation and to get an accurate pattern simulation, required data needed (Fig. 1).

All graphs considered in this paper are finite, undirected, connected with neither loops nor multiple edges [8]. For basic terminologies and notations of graphs, we follow [3]. *Let* $G = (V, E)$ be a graph with vertex set $V(G)$ and edge set $E(G)$. The order of G is $|V(G)| = p$ and the size of G is $|E(G)| = q$. A graph labeling is one of big concepts in graph theory that has attracted many mathematicians around the globe. A graph labeling is a mapping from the set of elements in a graph (vertices, edges, or both) to the set of numbers (usually positive integers), called labels. There are many types of graph labeling techniques that have been established see [6] for the most complete survey on labelings.

In [6] defined antimagic graphs. A graph G is called antimagic if there exists a bijection $f : E(G) \rightarrow \{1, 2, ..., q\}$ such that the weights of all vertices are distinct. The vertex weight of a vertex v under f, $wf(v)$, is the sum of labels of edges incident with v, that is, $wf(v) = \Sigma uv \in (G)f(uv)$. In this case, f is called an antimagic labeling. Futhermore, an (a,d) - edge-antimagic vertex labeling of graphs was defined in [18]. For a given graph G with p vertices and q edges, a bijection $f : V(G) \rightarrow \{1, 2, ..., p\}$ is called an (a,d) - edge-antimagic vertex labeling of G if the setof edge weights consists of

Fig. 1. The Ilustration of farming design on green house [9]

an arithmetic progression a, $a + d, ..., a + (q_1)d$, where a and d are two fixed positive integers. The edge weight of an edge e = uv under f, $wf\ (uv)$, is the sum of labelsof its end vertices, that is, $wf(uv) = f(u) + f(v)$. A graph that admits an (a, d)-edge antimagic vertex labeling is called an (a, d) - edge-antimagic vertex graph.

Another important topic in graph theory is graph colorings. In [4] introduced the term rainbow coloring of a graph. Let $c : E(G) \rightarrow \{1, 2, ..., k\}$ be an edge k − coloring of a graph G where adjacent edges may be colored the same. A path in G is rainbow if no two edges of it are colored the same. The edge-colored graph G is rainbow-connected if every two distinct verticesare connected by a rainbow path. The edge k - coloring in which G is rainbow-connected is calleda rainbow k - coloring. The minimum integer k - coloring in order to make G rainbow - connected is called the rainbow connection number of G and is denoted by $rc(G)$. This graph invariant has gained many attentions in [2, 11, 12, 14–16]. The most complete survey on rainbow colorings can be found the survey by [17].

Let $shack(c_4, y, n)$ be a shackle graph with $(Vshack(c_4, y, n)) = \{x_i, 1 \leq i \leq n\} \cup \{y_i, 1 \leq i \leq n + 1\} \cup \{z_i, 1 \leq i \leq n\}$ and $E(shack(c_4, y, n)) = \{x_i y_i, x_i y_{i+1}, 1 \leq i \leq n\} \cup \{z_i y_i, z_i y_{i+1}, 1 \leq i \leq n\}$. The cardinality of the vertices set of $|V(shack(c_4, y, n))| = 3n + 1$ and the cardinality of the edges of $|E(shack(c_4, y, n))| = 4n$. The rainbow antimagic coloring of $shack(c_4, y, n) = 2n$. The following is the label on vertex and edge from $shack(c_4, y, n)$.

$$
\begin{aligned}
l(x_i) &= \begin{cases} 3n + 1 \text{ for } i = 1 \\ n + 1 \quad \text{for } 2 \leq i \leq n \end{cases} \\
l(y_i) &= i \quad \text{for } 1 \leq i \leq n \\
l(z_i) &= 3n + 1 - i \quad \text{for } 1 \leq i \leq n \\
w(x_i y_i) &= \begin{cases} 3n + 1 - i \text{ for } 1 \leq i \leq 2 \\ n + 2_i \quad \text{for } 2 \leq i \leq n \end{cases} \\
w(x_i y_i + 1) &= n + 2_i + 1 \quad \text{for } 2 \leq i \leq n \\
w(z_i y_i) &= 3n + 1 \quad \text{for } 1 \leq i \leq n \\
w(z_i y_i + 1) &= 3n + 2 \quad \text{for } 1 \leq i \leq n
\end{aligned}
$$

Next, it is evaluated that the $shack(c_4, y, n)$ edge coloring is a rainbow antimagic connection. Suppose that, taken $x, y \in Vshack(c_4, y, n)$ is showed to be 6 cases as show in Table 1.

Furthermore, let $W = (w_{ij})$ be a matrix which is used to store the connection weights for a neural network. The network input to unit the target Y_j with product of the vectors $x = (x_1, x_2, .., x_n)$ and w_{ij} (the j column of the weight matrix), $Y' = x \cdot w.j = \sum_{i=1}^{n} x_i w_{ij}$. A bias can be included by adding a component $x_0 = 1$ to the vector$x$, namely $x = (1, x_1, x_2, .., x_n)$. For illustration of neural network can be seen in Fig. 2.

The network input to unit Y_j is given by.

$$Y' = \sum_{i=0}^{n} x_i w_{ij} = w_{0j} + \sum_{i=1}^{n} x_i w_{ij} = \beta_j + \sum_{i=0}^{n} x_i w_{ij}$$

Table 1. Rainbow path in $shack(c_4, y, n)$

Case	x	y	Rainbow path
1.	x_i	x_j	$x_i, y_{i+1}, x_{i+1}, \ldots x_j$
2.	x_i	y_j	$x_i, y_{i+1}, x_{i+1}, \ldots y_j, z_j$
3.	x_i	z_j	$x_i, y_{i+1}, x_{i+1}, \ldots y_j, z_j$
4.	y_i	y_j	$y_i, x_i, y_{i+1}, x_{i+1}, \ldots y_j$
5.	y_i	z_j	$y_i, x_i, y_{i+1}, x_{i+1}, \ldots y_j, z_j$
6.	z_i	z_j	$z_i, y_i, x_i, y_{i+1}, x_{i+1}, \ldots x_j, y_j, z_j$

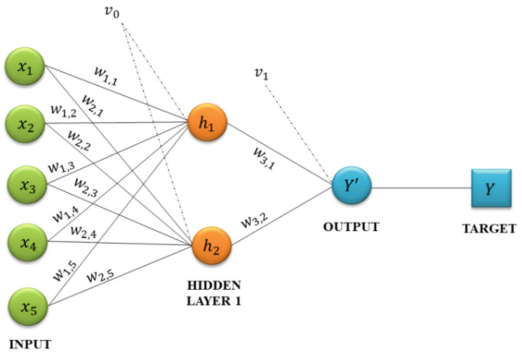

Fig. 2. The Ilustration of artificial neural network architecture.

There are some types of activation functions, The basic operation of neural networks is to sum its weight times input signal and apply the activation function [5].

(1) Linear activation function

$Y' = f(x) = ax + b$, when $a = 1, b = 0$, it is an identity.

(2) Binary step activation function with threshold θ

$$Y' = f(x) = \begin{cases} 1 \ if \ x \geq \theta \\ 0 \ if \ x < \theta \end{cases}$$

(3) Binary sigmoid activation function

$$Y' = F(x)\frac{1}{1+e^{-\sigma x}} \text{ and } f'(x) = \sigma f(x)\big[1 - f(x)\big].$$

(4) Bipolar sigmoid activation function

$$Y' = g(x) = 2f(x) - 1 = \frac{1}{1+e^{-\sigma x}} \text{ and } g'(x) = \frac{\sigma}{2}\big[1 + g(x)\big]\big[1 - g(x)\big].$$

(5) Hyperbolic tangent activation function (Tanh)

$$Y' = h(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \text{ and } h'(x) = [1 + h(x)][1 - h(x)].$$

The aim of this research is design an ANN based automatic watering system with the laying of plant using the rainbow antimagic coloring, vertex of the graph serves to determine the location of coordinates position plants, the edge on the graph serve to automatic watering system. The soil moisture used to determine the decrease in water content in the soil, so that it can carrying out minimal watering based on forecasting the watering period required. Our also studies architecture mathematically on artificial neural networks, analyze the error and analyze how are their bias and weight updated. Through this research, we also trained dan tested some independents input data related to the temperature, air humidity and soil moisture in lemon grass plant, ginger plant, turmeric plant and mint leaves plant.

2 Methods

The type of the research is an analytical and experimental study. In general, the analytical study uses a bibliography analysis to explore the findings. With combining between the analytical and experimental approach. We analyzed the effectiveness of four models with two artificial neural network architectures for soil moisture forecasting in the horizontal greenhouse systems by using matlab programming in term of iteration number, correlation, learning rate and mean square error.

At the experimental study stage, the research is directed at three stages, namely training stage, testing stage, and forecasting stage. The training stage aims to obtain an effective model of ANN. In testing stage, it aims to measure how accurate the determined learning model. The obtained model is then used for doing forecasting to have a decision making. In this research, the data set has been collected as many as 1.152 data set from four types of plants, were observed at 06.00 a.m. - 06.00 p.m. with range 15 min for six days. The last we analyze the use of rainbow antimagic antimagic coloring on planting topology to place the soil moisture sensor.

This agricultural activity involves soil components so that there is a need for research related to variables related to soil components. There are three fundamental variables that affect soil quality in agriculture including temperature, humidity, and soil moisture. With arduino box micro control board tools there is already a circuit that can be entered a code to carry out our instructions. NodemCu 8266 a device used for sending data over long distances, the data we send are analog data and digital data, digital data is data in the form of a value read by a sensor, for analog data we use conversion data from electrical signal that will be converted into soil moisture. DHT11 reads two variables, two components namely temperature and humidity, placed in the air, resistant to light rain weather, for heavy rain weather we modify the place to be used to place the sensor. Jumper wires connect the cables on the sensor and arduino. This is a technical program, so we don't use a solder connection for the sensor to the box.

Therefore, from the humidity data, air temperature and soil moisture will be processed using the matlab programs for machine learning and deep learning topics. The

Internet of Thing (IoT) involved is the existence of an electrical circuit that we use in constructing the sensor, on DHT11 which is connected to the nodemCu circuit there are two components that is reading data and sending, reading using sensors while sending it using an internet connection. The final result of our sensor readings can show how much soil moisture the soil can be watered, so that we have got the soil moisture as a threshold value for watering plants, we enter the data to arduino.

3 Findings

A. The Rainbow Antimagic Coloring

Let $shack(c_4, y, n)$ be a shackle graph with $(Vshack(c_4, y, n)) = \{x_i, 1 \leq i \leq n\} \cup \{y_i, 1 \leq i \leq n+1\} \cup \{z_i, 1 \leq i \leq n\}$ and$E(shack(c_4, y, n)) = \{x_i y_i, x_i y_{i+1}, 1 \leq i \leq n\} \cup \{z_i y_i, z_i y_{i+1}, 1 \leq i \leq n\}$. The cardinality of the vertices set of $|V(shack(c_4, y, n))| = 3n + 1$ and the cardinality of the edges of $|E(shack(c_4, y, n))| = 4n$. The rainbow antimagic coloring of $shack(c_4, y, n) = 2n$. From shack graph with $c = 4$ and $n = 4$ we have the edge weights obtained is $W = \{7, 8, 9, 10, 11, 12\}$. The weight of 7 obtained from the sum $2 + 5$, for weight of 10 obtained from the sum $1 + 9$, for weight of 11 obtained from the sum $1 + 10$, for weight of 12 obtained from the sum $2 + 10$. For more details can be seen in Table 2 (Fig. 3).

B. *Soil Moisture*

The forecasting of watering time on lemongrassplantation is two days later after the last watering. Threshold value for watering plants is 50%. On forecasting of watering

Table. 2. The weight of edge on shack graph with $Vshack(c_4, y, n)$

Weight 7, 9	Weight 10	Weight 11	Weight 12, 8
2,5	1,9	1,10	2,10
3,6	2,8	2,9	3,5
	3,7	3,8	
	4,6	4,7	

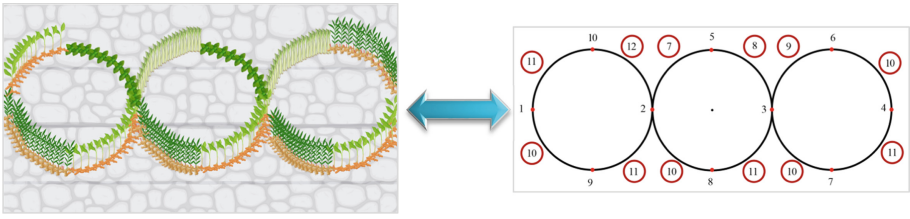

Fig. 3. The ilustration of use Rainbow Antimagic Coloring to Automatic Irrigation System on Horizontal Farming

time on ginger plantation is three days later after the last watering. Threshold value for watering plants is 61%, forecasting of watering time on turmeric plantation is three days later after the last watering. Threshold value for watering plants is 58%. and forecasting of watering time on mint leaves plantation is two day later after the last watering. Threshold value for watering plants is 58% (Figs. 4 and 5).

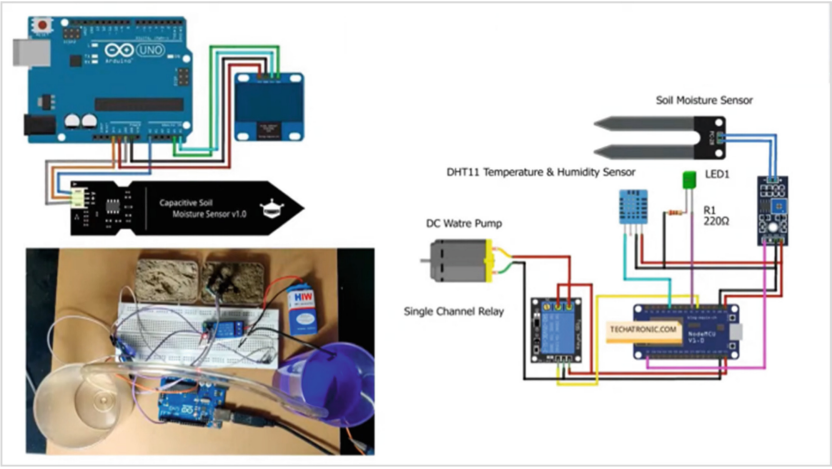

Fig. 4. The Ilustration on soil moisture. [10]

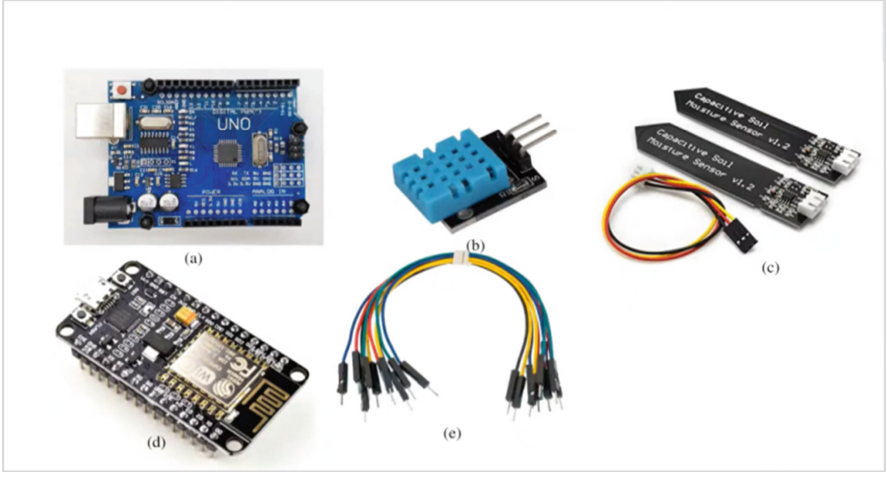

Fig. 5. The components from soil moisture. (a) Arduino (b) NodemCu 8266 wifi (c) Soilmoisture V.2 (sensor) (d) NodemCu 8266 wifi (e) Jumper wires [10].

C. *Mathematical Analysis*

In this section, manual steps for calculating ANN will be presented from five input layers, one hidden layer of two nodes and one target. The illustration can be seen in the Fig. 6.

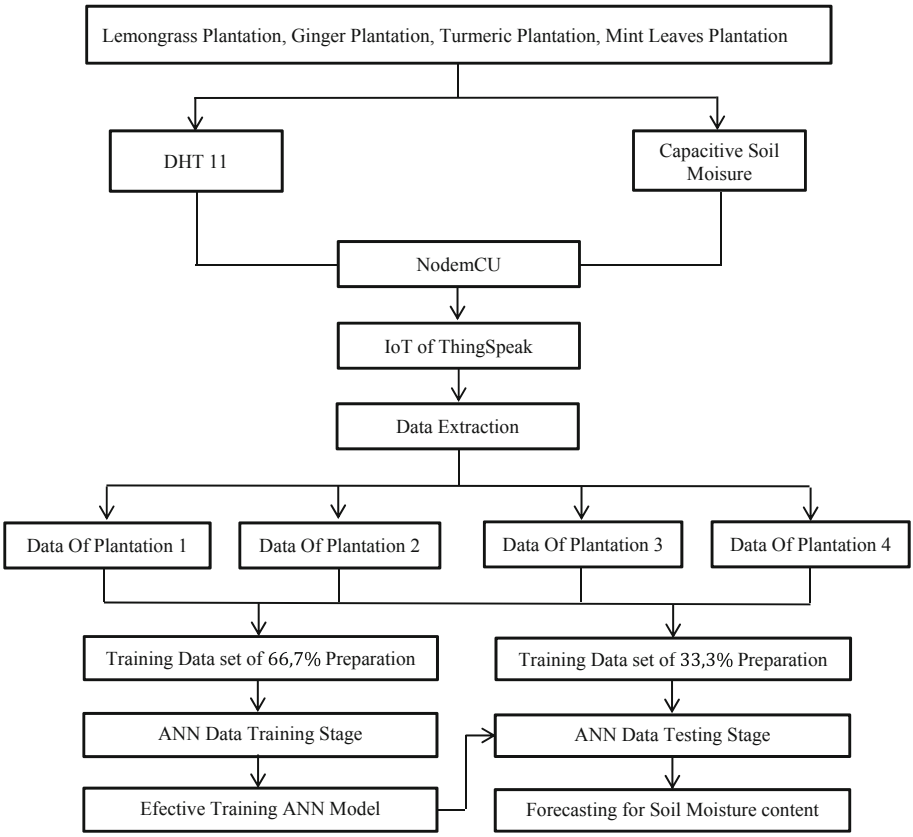

Fig. 6. The implementation framework of rainbow antimagic coloring

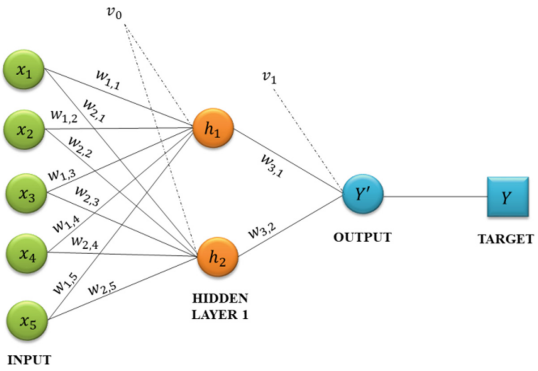

Fig. 7. The illustration of neural network architecture with one layer of two nodes.

Backpropagation Algorithm

Step 0. Initialize weights w, β, α, error_goal

Step 1. Stage Feedforward

$h_i = \beta + \sum_{i=0}^{n} x_i \cdot w_{i,j}$ where n is lots of input

Step 2. Counting Output

$$y_{in_i} = \beta + \sum_{i=0}^{n} w_{i,j} \cdot h_i$$

Step 3. Activations output using log sigmoid.

$$\hat{y}_m = \frac{1}{1 + e^{-y_{in_m}}}$$

Step 4. Counting output error

$$\delta k_i = (t - \hat{y}_i)^2$$

$$MSE = \frac{\sum_{i=0}^{n}(t - \hat{y}_i)^2}{n}$$

Step 5. Counting backpropagation error between output layer with hidden layer

$$\delta j_i^j = \beta \sum_{i=0}^{n} w_{i,j} * \delta k_i$$

Step 6. Counting backpropagation error between hidden layer with input layer

$$\delta j_i^j = \sum_{i=0}^{n} w_{i,j} * \delta k_i$$

Step 7. Update weight and bias

$$W_{new} = W_{old} + \alpha * \delta k_i * h_i$$

$$\beta_{new} = \beta_{old} + \alpha * \delta j_i + x_i$$

Step 8. Checking termination criteria (maximum epoch count or MSE≤error_goal)

Step 9. If not, go back to step 1

Let $W = (w_{ij})$ is a matrix that represents the weights, namely the relationship between one neuron and another neuron. The network input to the target unit Y_j (with no bias to unit j) is a simple dot product of the vectors $x = (x_1, x_2, \ldots, x_n)$ where n is the number of inputs and w_j where j is the number of columns in the weight matrix.

So that, $\hat{Y} = x \cdot w_j = \sum_{i=1}^{n} x_i w_{ij}$. Figure 6 illustrates a simple neural network. Bias (β) can be included in vector x by adding the component $x_0 = 1$, so that vector x becomes $x = (1, x_1, x_2, \ldots, x_n)$.

Observation 1. Given a neural network with one hidden layer there are two neurons. Figure 6 illustrates this artificial neural network. For example, $x = \{x_i | i = 1, 2, \ldots, n\}$, where x is the input data and n is the number of input data. Let h and β be the number of hidden layers and bias. The output ($\hat{Y}$) is written as $= \frac{1}{1+e^{-(\beta_1 + W^1(\beta_0 + W^0 x))}}$, where $[h_1; h_2] = \beta^0 + W^0 x$.

Proof. Input the data $x = \{x_1, x_2, \ldots, x_n\}$, cause we have one layer and two neurons, we define weights $W^0 = \begin{bmatrix} w^0_{1,1}\ w^0_{1,2}\ \cdots w^0_{1,n}\ ;\ w^0_{2,1}\ w^0_{2,2}\ \cdots w^0_{2,n} \end{bmatrix}$, $W^1 = \begin{bmatrix} w^1_1\ w^1_2 \end{bmatrix}$ and bias β^0, β^1.

$$\begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} \beta^0_1 \\ \beta^0_2 \end{bmatrix} + \begin{bmatrix} w^0_{1,1}\ w^0_{1,2} \cdots w^0_{1,n} \\ w^0_{2,1}\ w^0_{2,1} \cdots w^0_{2,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \beta^0 + W^0 x$$

Then we obtain y_{in} as follows

$$y_{in} = \beta^1 + \begin{bmatrix} w^1_1 \\ w^1_2 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \beta^1 + \begin{bmatrix} w^1_1 \\ w^1_2 \end{bmatrix} \cdot \left(\beta^0 + W^0 x\right)$$
$$y_{in} = \beta^1 + W^1\left(\beta^0 + W^0 x\right).$$

So that the resulting output ($\hat{y}$) is as follows

$$\hat{y} = \frac{1}{1 + e^{-y_{in}}} = \frac{1}{1 + e^{-(\beta^1 + W^1(\beta^0 + W^0 x))}}$$

■

Observation 2 Given a neural network architecture with two hidden layers. In each hidden layer there are two and three neurons. Figure 6 illustrates the architecture of this neural network. Let h and β be the number of hidden layers and bias, respectively. The output $(\hat{y})$ is written as $\hat{y} = \frac{1}{1+e^{-(\beta_1 + \sum_{i=1}^{2} w_i h_i)}}$, where $[h_1; h_2] = \beta^0 + W^0 x$.

Proof. Input the data $x = \{x_1, x_2, \ldots, x_n\}$, we have two hidden layers which have two neurons and three neurons, we define weights.

$W^0 = \begin{bmatrix} w^0_{1,1}\ w^0_{1,2} \ldots w^0_{1,n}; w^0_{2,1}\ w^0_{2,2} \ldots w^0_{2,n} \end{bmatrix}$, $W^1 = \begin{bmatrix} w^1_{1,1}\ w^1_{1,2}; w^1_{2,1}\ w^1_{2,2}; w^1_{3,1}\ w^1_{3,2} \end{bmatrix}$,

$W^2 = \begin{bmatrix} w^2_1; w^2_2; w^2_3 \end{bmatrix}$ and bias $\beta_0,\ \beta_1,\ \beta_2$.

The output from hidden layer 1 is as follows.

$$\begin{bmatrix} h^1_1 \\ h^1_2 \end{bmatrix} = \begin{bmatrix} \beta^0_1 \\ \beta^0_2 \end{bmatrix} + \begin{bmatrix} w^0_{1,1}\ w^0_{1,2} \cdots w^0_{1,n} \\ w^0_{2,1}\ w^0_{2,1} \cdots w^0_{2,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \beta^0 + W^0 x$$

The output from hidden layer 2 is as follows.

$$\begin{bmatrix} h_1^2 \\ h_2^2 \\ h_3^2 \end{bmatrix} = \begin{bmatrix} \beta_1^1 \\ \beta_2^1 \\ \beta_3^1 \end{bmatrix} + \begin{bmatrix} w_{1,1}^1 & w_{1,2}^1 & \cdots & w_{1,n}^1 \\ w_{2,1}^1 & w_{2,1}^1 & \cdots & w_{2,n}^1 \\ w_{3,1}^1 & w_{3,2}^1 & \cdots & w_{3,2}^1 \end{bmatrix} \begin{bmatrix} h_1^1 \\ h_2^1 \end{bmatrix}$$

$$\begin{bmatrix} h_1^2 \\ h_2^2 \\ h_3^2 \end{bmatrix} = \begin{bmatrix} \beta_1^1 \\ \beta_2^1 \\ \beta_3^1 \end{bmatrix} + \begin{bmatrix} w_{1,1}^1 & w_{1,2}^1 & \cdots & w_{1,n}^1 \\ w_{2,1}^1 & w_{2,1}^1 & \cdots & w_{2,n}^1 \\ w_{3,1}^1 & w_{3,2}^1 & \cdots & w_{3,2}^1 \end{bmatrix} \cdot \left(\beta^0 + W^0 x\right)$$

$$\left[h_1^2\right] = \beta^1 + W^1\left(\beta^0 + W^0 x\right)$$

Then we obtain y_{in} as follows

$$y_{in} = \beta^2 + \begin{bmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \end{bmatrix} \begin{bmatrix} h_1^2 \\ h_2^2 \\ h_3^2 \end{bmatrix} = \beta^2 + \begin{bmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \end{bmatrix} \cdot \left(\beta^1 + W^1\left(\beta^0 + W^0 x\right)\right)$$

$$y_{in} = \beta^2 + W^2\left(\beta^1 + W^1\left(\beta^0 + W^0 x\right)\right)$$

So that the resulting output $\left(\hat{y}\right)$ is as follows

$$\hat{y} = \frac{1}{1 + e^{-y_{in}}} = \frac{1}{1 + e^{-(\beta^2 + W^2(\beta^1 + W^1(\beta^0 + W^0 x)))}}$$

■

Illustration

$$x = \left[0.79;\ 0.78;\ 0.77;\ 0.76;\ 0.75;\ 0.74;\ 0.73;\ 0.72;\ 0.71\right]$$

These vectors then become input to Fully-Connected Neural Networks (FCNN). Some of the parameters we used in FCNN are weights $\left(W^0 \text{ and } W^1\right)$, bias $\left(\beta^0 \text{ and } \beta^1\right)$, learning rate $\alpha = 0.1$, and target (t).

$$W^0 = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.1 & 0.2 & 0.3 & 0.1 & 0.2 & 0.3 \\ 0.2 & 0.3 & 0.1 & 0.2 & 0.3 & 0.1 & 0.2 & 0.3 & 0.1 \\ 0.3 & 0.1 & 0.2 & 0.3 & 0.1 & 0.2 & 0.3 & 0.1 & 0.2 \end{bmatrix}$$

$$W^1 = [0.1;\ 0.2;\ 0.3]$$

$$\beta^0 = [0.1;\ 0.1;\ 0.1]$$

$$\beta^1 = [0.1;\ 0.1;\ 0.1]$$

$$t = 1$$

The first step in FCNN is feedforward neuron one as follows.

$$h_{1,1}^1 = \beta_1^0 + x_1 * w_{1,1}^0 = 0.1 + 0.1 * 0.1 = 0.1790$$

$$h_{1,2}^1 = \beta_1^0 + x_2 * w_{1,2}^0 = 0.1 + 0.1 * 0.2 = 0.2560$$

$$\vdots$$

$$h_{1,9}^1 = \beta_1^0 + x_9 * w_{1,9}^0 = 0.1 + 0.81 * 0.3 = 0.3130$$

$$h_1^1 = \left[0.1790\ 0.2560 \ldots 0.2440\ 0.3130\right]$$

Next is feedforward neuron two as follow.

$$\begin{aligned}
h_{2,1}^1 &= \beta_2^0 + x_1 * w_{2,1}^0 = 0.1 + 0.1 * 0.2 = 0.2580 \\
h_{2,2}^1 &= \beta_2^0 + x_2 * w_{2,2}^0 = 0.1 + 0.1 * 0.3 = 0.3340 \\
&\vdots \\
h_{2,9}^1 &= \beta_2^0 + x_9 * w_{2,9}^0 = 0.1 + 0.81 * 0.1 = 0.1710 \\
h_2^1 &= \left[0.2580\ 0.3340 \ldots 0.3160\ 0.1710\right]
\end{aligned}$$

The last feedforward is in neuron three, the calculate as follow.

$$\begin{aligned}
h_{3,1}^1 &= \beta_3^0 + x_1 * w_{3,1}^0 = 0.1 + 0.1 * 0.3 = 0.3370 \\
h_{3,2}^1 &= \beta_3^0 + x_2 * w_{3,2}^0 = 0.1 + 0.1 * 0.1 = 0.1780 \\
&\vdots \\
h_{3,9}^1 &= \beta_3^0 + x_9 * w_{3,9}^0 = 0.1 + 0.81 * 0.2 = 0.1750 \\
h_3^1 &= \left[0.3370\ 0.1780 \ldots 0.1750\ 0.1750\right]
\end{aligned}$$

Next is to calculate the output (y)

$$\begin{aligned}
yin_1 =& \beta_1^1 + w_1^1 * h_{1,1}^1 + \cdots + w_1^1 * h_{1,9}^1 + w_2^1 * h_{2,1}^1 + \ldots \\
&+ w_2^1 * h_{2,9}^1 + w_3^1 * h_{3,1}^1 + \ldots + w_3^1 * h_{3,9}^1 \\
yin_1 =& 0.1 + 0.1 * 0.2580 + \cdots + 0.1 * 0.1710 + \cdots + 0.1 \\
&* 0.1790 + \cdots + 0.1 * 0.3130 + \cdots + 0.1 * 0.3370 \\
&+ \cdots + 0.1 * 0.1750 \\
yin_1 =& 1.3666
\end{aligned}$$

Then activate each neuron output. The activation function used is the sigmoid log.

$$\begin{aligned}
y_m &= \frac{1}{1 + e^{-yin_m}} \\
y_1 &= 0.7968
\end{aligned}$$

After obtaining the next output, we can calculate the output error.

$$\begin{aligned}
\delta k_1 &= (t_1 - y_1)^2 \\
\Delta k_1 &= (1 - 0.8066)^2 \\
\Delta k_1 &= 0.0413 \\
MSE &= \delta k_1 = 0.0413
\end{aligned}$$

Then calculate the backpropagation error between output layer and hidden layer:

$$\delta j_1^0 = \beta_1^0 + w_1^1 * \delta k_1$$

$$\delta j_1^0 = 0.1 + 0.1 * 0.0374$$
$$\delta j_1^0 = 0.1041$$

Then calculate the backpropagation error between hidden layer and input layer:

$$\delta j_1^1 = w_{1,1}^0 * \delta k_1 + w_{1,2}^0 * \delta k_1 + w_{1,3}^0 * \delta k_1 + \ldots + w_{1,9}^0 * \delta k_1$$
$$\delta j_1^1 = 0.0743$$
$$\delta j_2^1 = w_{2,1}^0 * \delta k_1 + w_{2,2}^0 * \delta k_1 + w_{2,3}^0 * \delta k_1 + \ldots + w_{2,9}^0 * \delta k_1$$
$$\delta j_1^1 = 0.0743$$
$$\delta j_3^1 = w_{3,1}^0 * \delta k_1 + w_{3,2}^0 * \delta k_1 + w_{3,3}^0 * \delta k_1 + \ldots + w_{3,9}^0 * \delta k_1$$
$$\delta j_3^1 = 0.0743$$

The final step is to update the weights and biases as follows:

- The weights between input layer and hidden layer

$$w\_new_{1,1}^0 = w_{1,1}^0 + \alpha * \delta k_1 * h_1^1$$
$$w_{new1,1}^0 = \left[0.1019\ 0.2027\ \ldots\ 0.2025\ 0.3033\right]$$
$$w\_new_{2,1}^0 = w_{2,1}^0 + \alpha * \delta k_1 * h_2^1$$
$$w\_new_{2,1}^0 = \left[0.2027\ 0.3035\ \ldots\ 0.3033\ 0.1018\right]$$
$$w\_new_{3,1}^0 = w_{3,1}^0 + \alpha * \delta k_1 * h_3^1$$
$$w\_new_{3,1}^0 = \left[0.3035\ 0.1019\ \ldots\ 0.1018\ 0.1018\right]$$
$$W\_new^0 = \begin{bmatrix} w\_new_{1,1}^0 \\ w\_new_{2,1}^0 \\ w\_new_{3,1}^0 \end{bmatrix}$$
$$W\_new^0 = \begin{bmatrix} 0.1019;\ 0.2027;\ \ldots\ 0.2025;\ 0.3033; \\ 0.2027;\ 0.3035;\ \ldots\ 0.3033;\ 0.1018; \\ 0.3035;\ 0.1019;\ \ldots\ 0.1018;\ 0.1018; \end{bmatrix}$$

- The weights between hidden layer and output layer

$$w\_new_1^1 = w_1^1 + \alpha * \delta k_1$$
$$w\_new_1^1 = 0.1093$$
$$w\_new_2^1 = w_2^1 + \alpha * \delta k_1$$
$$w\_new_1^1 = 0.2101$$
$$w\_new_3^1 = w_3^1 + \alpha * \delta k_1$$
$$w\_new_1^1 = 0.3109$$
$$W\_new^1 = \begin{bmatrix} w\_new_1^1 \\ w\_new_2^1 \\ w\_new_3^1 \end{bmatrix} = \begin{bmatrix} 0.1093 \\ 0.2101 \\ 0.3109 \end{bmatrix}$$

- Bias between input layer and hidden layer

$$\beta\_new_1^0 = \beta_1^0 + \alpha \cdot \delta j_1^1 = 0.1102$$

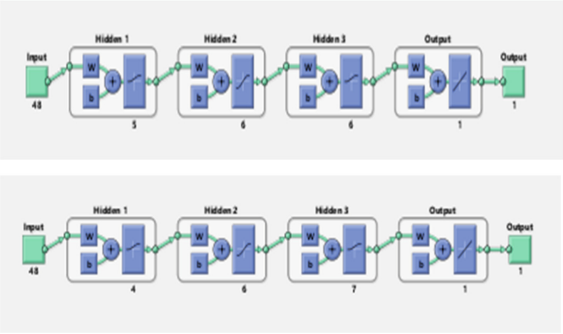

Fig 8. The Illustration of neural network architecture with hidden layers 566 and 467

$$\beta\_new_2^0 = \beta_2^0 + \alpha \cdot \delta j_2^1 = 0.1102$$
$$\beta\_new_3^0 = \beta_3^0 + \alpha \cdot \delta j_3^1 = 0.1102$$
$$\beta\_new^0 = \left[\beta\_new_1^0;\ \beta\_new_2^0;\ \beta\_new_3^0\right]$$
$$\beta\_new^0 = \left[0.1102;\ 0.1102;\ 0.1102\right]$$

- Bias between hidden layer and output layer

$$\beta\_new_1^1 = \beta_1^1 + \alpha \cdot \delta j_1^0 = 0.1102$$

D. *Data Training, Testing and Forecasting (Fig. 9)*

4 Results and Discussion

In this section we will do some simulation on two artificial neural networksarchitectures. The model used is Feedforwardnet, Fitnet, Paternet, and Cascadeforwardnet, for the architecture used in ANN-566 and ANN-657. Artificial Neural Networks architectures to train, test and forecast the data on horizontal farming issue namely temperature,air humidity and soil moisture. We then run the Matlab programming which has been developedin this study to test effectiveness of the two neural network architectures, see Fig. 8, and compare the effectiveness in term of ANN Models, Regression and Mean Square Error (MSE). In this simulation, we training 66,7% and testing the 33,3% of data set derived from four types on plant, see Table 2. We also did forcasting for a few days ahead on soil moisture.

The results of this study can be seen in Table 3. The selection of the best architecture is based on the smallest MSE value. Based on Table 3, the best architecture for lemongrass plantation is obtained by the fitnet ANN (b) 566 model, with MSE of 1.1101×10^{-15}. The best architecture for ginger plantation is obtained by model fitnet ANN (b) 566 with an MSE of 3.1663×10^{-18}. The best architecture for turmeric plantation is obtained by the model ANN- cascadeforwardnet (d) 467 with an MSE of 4.5127×10^{-19}. The best architecture for mint leaves plantation is obtained by model ANN feedforwardnet

Table 3. The performance indicator of ANN architectures and models on training and testing agriculture data set

LEMONGRASS					
ANN MODEL	ANN ARCHITEVTURE	MSE TRAIN	REGRESSION TRAIN	TIME TRAIN	MSE TEST
FEEDFORWARDNET	ANN- (A) 566	9.9444×10^{-11}	1	9.2584 s	$1.4917 \times {}^{-10}$
	ANN- (A) 467	7.5787×10^{-11}	1	1.6272 s	1.0309×10^{-10}
FITNET	ANN- (A) 566	1.3683×10^{-15}	1	1.0757 s	1.1101×10^{-15}
	ANN- (A) 467	9.9289×10^{-11}	1	7.6838 s	1.4892×10^{-10}
PATTERNNET	ANN- (A) 566	2.3004×10^{-9}	1	9.9695 s	2.5280×10^{-9}
	ANN- (A) 467	2.5804×10^{-10}	1	7.0055 s	2.9837×10^{-10}
CASCADEFORWARDNET	ANN- (A) 566	1.0300×10^{-13}	1	2.3743 s	1.1900×10^{-13}
	ANN- (A) 467	4.3817×10^{-11}	1	4.8501 s	6.3171×10^{-11}
GINGER					
ANN MODEL	ANN ARCHITEVTURE	MSE TRAIN	REGRESSION TRAIN	TIME TRAIN	MSE TEST
FEEDFORWARDNET	ANN- (A) 566	6.0124×10^{-12}	1	1.1536 s	3.0932×10^{-12}
	ANN- (A) 467	0.1081	1	8.5296 s	0.1570
FITNET	ANN- (A) 566	2.4393×10^{-18}	1	1.0325 s	3.1663×10^{-18}
	ANN- (A) 467	3.6121×10^{-11}	1	0.65263 s	2.2514×10^{-12}
PATTERNNET	ANN- (A) 566	1.8178×10^{-10}	1	9.3622 s	1.8418×10^{-10}
	ANN- (A) 467	0.0255	1	7.0165 s	0.0278
CASCADEFORWARDNET	ANN- (A) 566	1.3126×10^{-11}	1	1.1393 s	7.6266×10^{-13}
	ANN- (A) 467	2.9146×10^{-17}	1	2.1169 s	2.9536×10^{-17}
TURMERIC					
ANN MODEL	ANN ARCHITEVTURE	MSE TRAIN	REGRESSION TRAIN	TIME TRAIN	MSE TEST
FEEDFORWARDNET	ANN- (A) 566	3.8963×10^{-13}	1	0.70661 s	1.0875×10^{-14}
	ANN- (A) 467	4.9330×10^{-16}	1	0.63469 s	2.9699×10^{-16}

(*continued*)

Table 3. (*continued*)

LEMONGRASS					
ANN MODEL	ANN ARCHITEVTURE	MSE TRAIN	REGRESSION TRAIN	TIME TRAIN	MSE TEST
FITNET	ANN- (A) 566	9.3763×10^{-11}	1	23.7444 s	6.5148×10^{-12}
	ANN- (A) 467	3.5290×10^{-13}	1	6.6338 s	5.5575×10^{-15}
PATTERNNET	ANN- (A) 566	3.5055×10^{-9}	1	11.6458 s	3.6880×10^{-9}
	ANN- (A) 467	1.0160×10^{-9}	1	7.0519 s	1.2321×10^{-9}
CASCADEFORWARDNET	ANN- (A) 566	1.2322×10^{-18}	1	1.714 s	7.9625×10^{-19}
	ANN- (A) 467	4.1802×10^{-19}	1	1.9797 s	4.5127×10^{-19}
MINT LEAVES					
ANN MODEL	ANN ARCHITEVTURE	MSE TRAIN	REGRESSION TRAIN	TIME TRAIN	MSE TEST
FEEDFORWARDNET	ANN- (A) 566	9.1119×10^{-11}	1	1.2947 s	1.3668×10^{-10}
	ANN- (A) 467	2.5846×10^{-14}	1	0.69361 s	3.8121×10^{-14}
FITNET	ANN- (A) 566	1.2510×10^{-9}	1	7.5595 s	1.8764×10^{-9}
	ANN- (A) 467	4.7801×10^{-8}	1	23.9304 s	7.1692×10^{-8}
PATTERNNET	ANN- (A) 566	1.7619	1	7.3156 s	1.7054
	ANN- (A) 467	6.1402×10^{-8}	1	7.3731 s	7.1407×10^{-8}
CASCADEFORWARDNET	ANN- (A) 566	9.5774×10^{-13}	1	1.3386 s	1.0348×10^{-12}
	ANN- (A) 467	1.0854×10^{-12}	1	3.0063 s	1.5157×10^{-12}

(a) 467 with an MSE of 3.8121×10^{-14}. Finally, the best architecture for is obtained by model ANN-Cascadeforwardnet (d) 467 with an MSE of 4.5127×10^{-19}. Some of the parameters used in this research are training functions Lavenberg Marquardt, sigmoid and hyperbolic log transfer function sigmoid tangent. The target epochs, error and learning rate are 750, 10^{-10} respectively, and 0.1. Furthermore, we show the graphic on training, testing and forecasting for one day ahead to know the watering time on the lemongrass plantation. Based on Fig. 9, we know that for the next watering time of lemongrass plantation in two day later after the last watering. The watering time must be carried out on 05.30 p.m.–05.45 p.m. On the ginger plantation, based on Fig. 11 we know that for the next watering time of ginger plantation in three day later after the last watering. The watering time must be carried out on 04.30 p.m.–04.45 p.m. The ginger plantation, based on Fig. 13, we know that for the next watering time of turmeric

plantation in three day later after the last watering. The watering time must be carried out on 04.30 p.m.–04.45 p.m. The mint leaves plantation based on Fig. 15, we know that for the next watering time of mint leaves plantation in two days later after the last watering. The watering time must be carried out on 05.15 p.m.–05.30 p.m.

From shack graph with $c = 4$ and $n = 3$ we have the edge weights obtained is $W = \{7, 8, 9, 10, 11, 12\}$. We have there are 2 edge that have a weight of 7,9. There are 4 edges that have a weight of 10, There are 4 edges that have a weight of 11 and There are 2 edges that have a weight of 12,8. Therefore we representation on each edge to a specific type of plant. So, in this graph there are 12 edges with 4 types of edge weights.

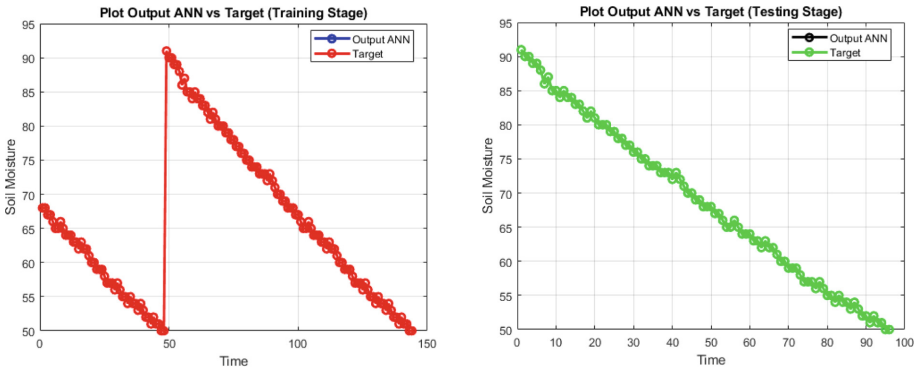

Fig. 9. The comparison of ANN output and target data of the soil moisture on training and testing stage. The illustration shows that ANN output interpolates uniformly on the data target either in training and testing stage from the lemongrass plantation.

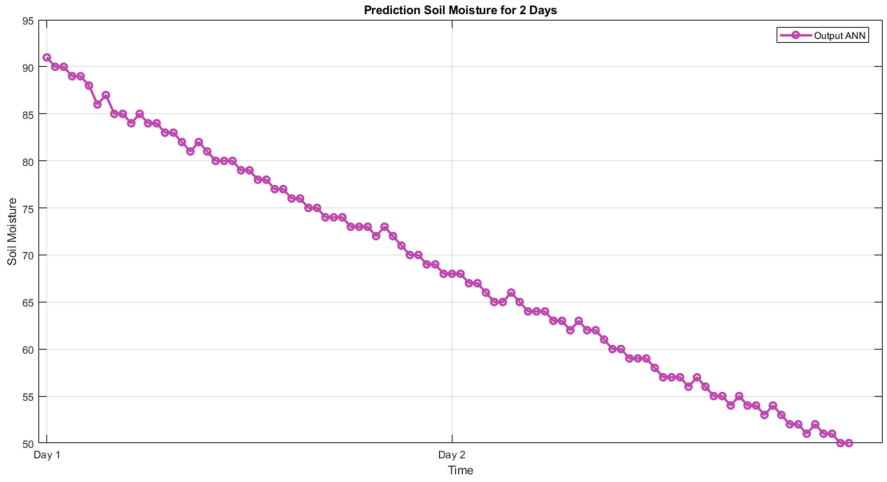

Fig. 10. The forecasting of soil moisture for determining the watering time on lemongrass plantation. The illustration shows that the next watering time of lemongrass plantation is in two day later after the last watering.

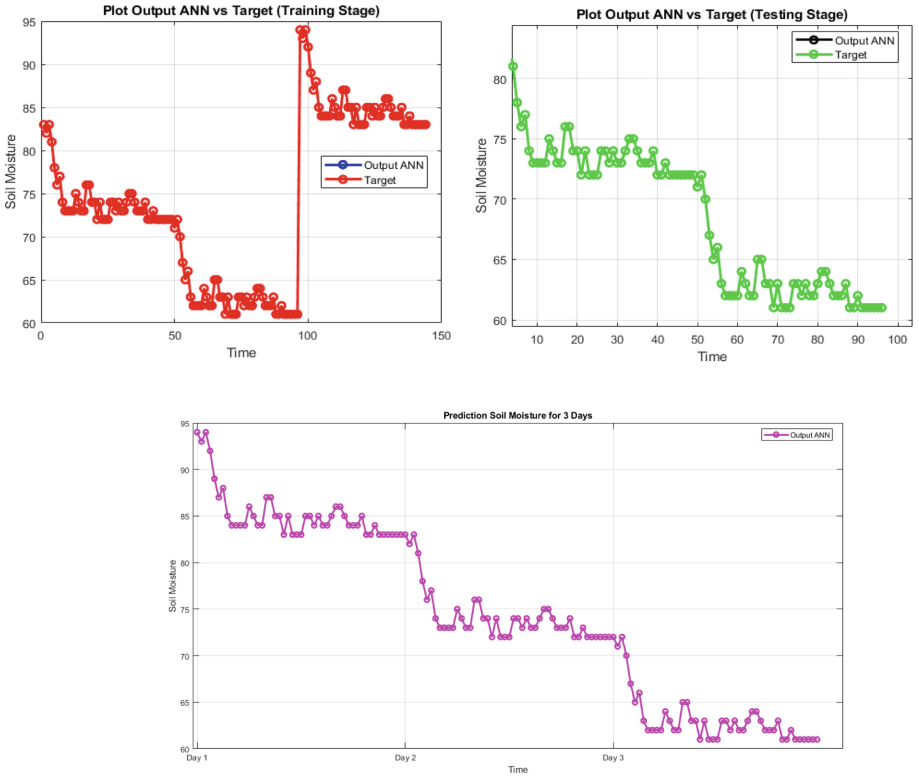

Fig. 11. The forecasting of soil moisture for determining the watering time ginger plantation. The illustration shows that the next watering time of ginger plantation is in three day later after the last watering.

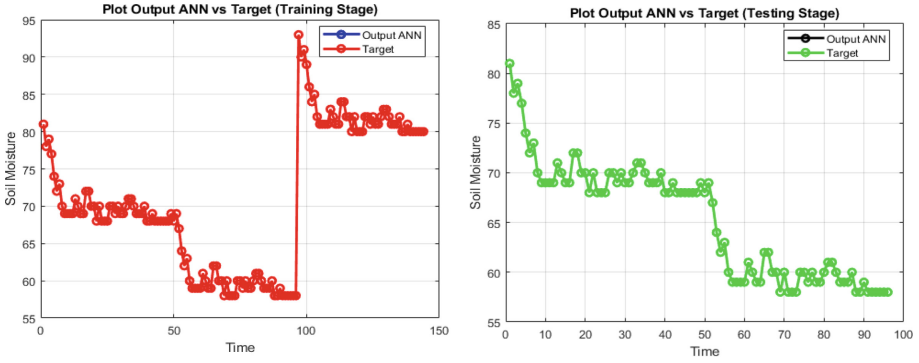

Fig. 12. The comparison of ANN output and target data of the soil moisture on training and testing stage. The illustration shows that ANN output interpolates uniformly on the data target either in training and testing stage from the turmeric plantation.

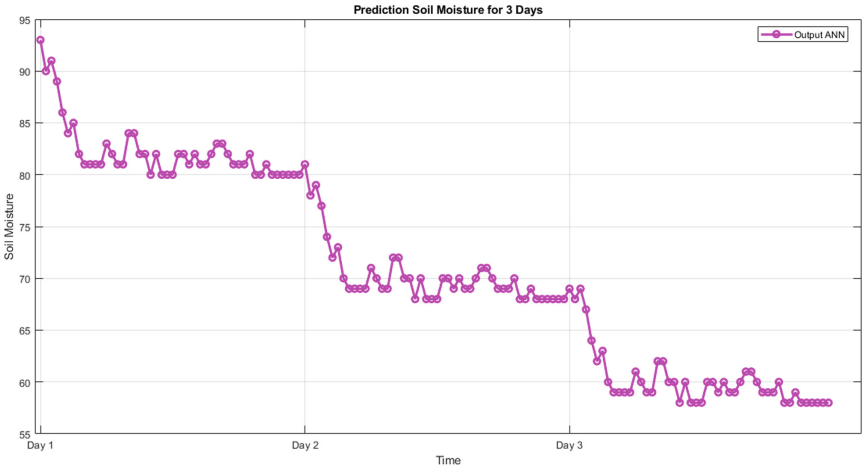

Fig. 13. The forecasting of soil moisture for determining the watering time turmeric plantation. The illustration shows that the next watering time of turmeric plantation is in three day later after the last watering

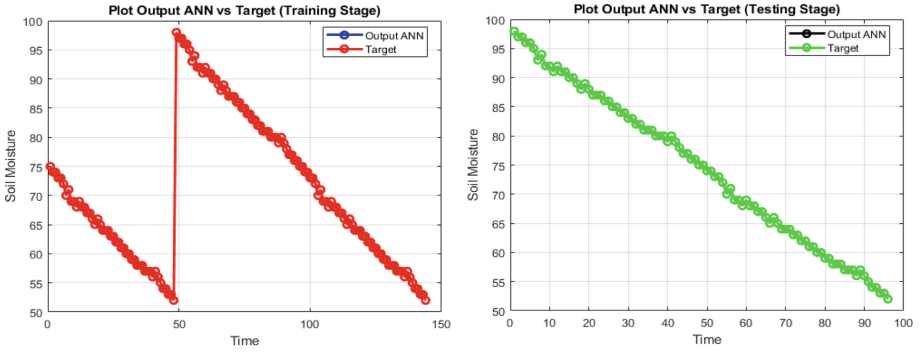

Fig. 14. The comparison of ANN output and target data of the soil moisture on training and testing stage. The illustration shows that ANN output interpolates uniformly on the data target either in training and testing stage from the mint leaves plantation.

The researcher determines the weight of 7,9 as a lemongrass plant, for a weight of 10 is a ginger plant, for a weight of 11 is a turmeric plant and for a weight of 12,8 is a mint leaves plant. The following is an overview of horizontal farming and its representation in shack graph with $c = 4$ and $n = 3$. In this paper only one forecasting result is included, as a representative of edges that have different weights $\{7, 8, 9, 10, 11, 12\}$, on each side weight has a different watering even though it has the same type of plant due to several internal factors from the plant. Figure 16 is a graph of the forecasting of soil moisture for determining the watering time on each plantation in the greenhouse.

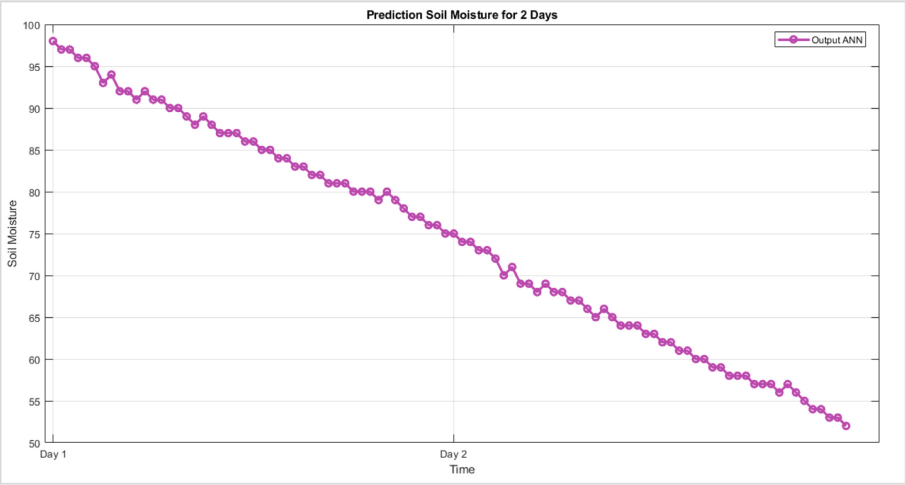

Fig. 15. The forecasting of soil moisture for determining the watering time on mint leaves plantation. The illustration shows that the next watering time of mint leaves plantation is in two day later after the last watering

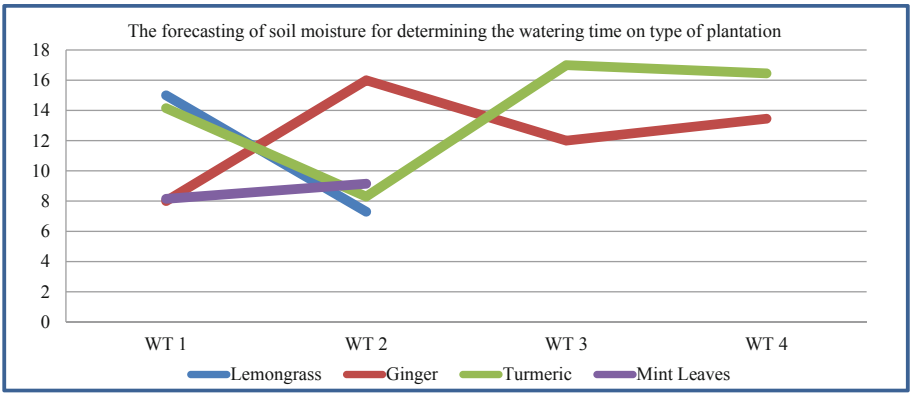

Fig. 16. The forecasting of soil moisture for determining the watering time on type of plantation. The illustration shows that the next watering time of plantation later after the last watering.

Based on Fig. 16, we obtain information that the blue line on the graph is the watering time of the lemongrass plants, the red line on the graph is ginger plants, the green line is green turmeric plants, and the purple color is mint leaves plants. There are different types of watering each crop area. Watering time (WT) is watering time information data obtained from the soil moisture sensor. Watering time (WT) is information data obtained from the soil moisture sensor. At 06.00 a.m. – 08.00 a.m. there are two areas of plants that must be watered, namely lemongrass (WT 2), and ginger (WT 1). At 08.01 a.m. – 10.00 a.m. there are three areas of plants that must be watered, namely Mint leaves (WT 1), turmeric (WT 2), and mint leaves (WT 2). At 10.01 a.m. – 12.00 a.m. there are one

Table 4. The classification of plantation watering based on watering time.

06.00 a.m. - 08.00 a.m.	08.01 a.m. - 10.00 a.m.	10.01 a.m. - 12.00 a.m.	12.01 a.m. -02.00 p.m.	02.01 p.m. - 04.00 p.m.	04.01 p.m - 06.00 p.m.
Lemongrass (WT 2)	Mint Leaves (WT 1)	Ginger (WT 3)	Ginger (WT 4)	Turmeric (WT 1)	Turmeric (WT 4)
Ginger (WT 1)	Turmeric (WT 2)			Lemongrass (WT 1)	Turmeric (WT 3)
	Mint Leaves (WT 2)			Ginger (WT 2)	

Table 5. The watering time specifications for each companion of plants

Plant	WT 1	WT 2	WT 3	WT 4
Lemongrass	03.00 p.m.	07.30 a.m		
Ginger	08.00 a.m.	04.00 p.m.	12.00 a.m	01.45 a.m
Turmeric	02.15 p.m.	08.30 a.m.	05.00 a.m.	04.45 p.m
Mint Leaves	08.15 a.m.	09.15 a.m.		

areas of plants that must be watered namely ginger (WT 3). At 12.01 a.m. – 02.00 p.m. there are one areas of plants that must be watered, namely ginger (WT 4). At 02.01 p.m. – 04.00 p.m. there are three areas of plants that must be watered, namely turmeric (WT 1), lemongrass (WT 1), and ginger (WT 2). At 04.00 p.m. – 06.00 p.m. there are two areas of plants that must be watered, namely turmeric (WT 4), and turmeric (WT 3). For more details can be seen in Table 4 and Table 5.

5 Conclusion

The horizontal farming using Artificial Intelligence and Machine Learning technology, including the concept of rainbow antimagic coloring will yield more harvests result than traditional agriculture. Utilization Internet of Things technology sourced from machine learning is a strategic step to cultivate larger plant varieties and companion plantation. In addition, plants are relatively more resistant to the weather disturbances due to its placement indoorly. It means fewer crops will be damaged by extreme weather or unexpected events, so that the sustainable conservation related to flora and fauna will be better preserved.

Acknowledgment. We would like to thank PUI-PT Combinatorics and Graph, CGANT, University of Jember, and especially the lecturer and members of the PUI-PT Combinatorics and Graph, CGANT, University of Jember for their outstanding guidance and support in the year 2023. We wish also to acknowledge the help provided by the technicians and staff in the hydrology laboratory, Universitas Jember. We would also like to show our deep appreciation to LP2M Universitas Jember, Indonesia.

References

1. A. E. García, G. M. S. Zarazúa, M. T. Ayala, E. R. Araiza and A. G. Barrios, "Applications of artificial neural networks in greenhouse technology and overview for smart agriculture development," Appl. Sci. 2020, 10, 3835; doi:https://doi.org/10.3390/app10113835.
2. Caro, Y., Lev, A., Roditty, Y., Tuza, Z. Yuster, R. (2008). On rainbow connection. Electron.J. Combin., 15, R57.
3. Chartrand, G. & Zhang, P. (2016). Graphs and digraphs, Sixth ed.., Taylor & Francis Group, 2016.
4. Chartrand, G., Johns, G.L., McKeon, K.A. Zhang, P. (2008). Rainbow connection in graphs. Math. Bohem., 133, 85-98.
5. Dafik, Z.R. Ridlo, I.H. Agustin, R. I. Baihaki, F.G. Febrinanto, R Nisviasari, Suhardi and A. Riski, "The implementation of artificial neural networks and resolving efficient dominating set for time series forecasting on soil moisture to advance the automatic irrigation system on vertical farming," in press.
6. Gallian, J.A. (2020). A dynamic survey of graph labeling. Electron. J. Combin., 23, DS6.
7. H. I. Khalil and K. A. Wahhab, "Advantage of vertical farming over horizontal farming in achieving sustainable city, Baghdad citycommercial street case study," IOP Conf. Series: Materials science and engineering 745 (2020) 012173 IOP Publishing doi:https://doi.org/10.1088/1757-899X/745/1/012173.
8. Hartsfield, N. Ringel, G. (1990). Pearls in Graph Theory. New York: Academic Press.
9. https://gpnmag.com/article/7-key-questions-to-ask-before-building-a-greenhouse/ (accesed: 15/12/2022).
10. https://www.youtube.com/watch?v=Ze3_NvQ0t7E (accesed: 15/12/2022).
11. Kemnitz, A. Schiermeyer, I. (2011). Graphs with rainbow connection number two. Discuss. Math. Graph Theory, 31, 313-320.
12. Krivelevich, M. Yuster, R. (2010). The rainbow connection of a graph is (at most) reciprocal to its minimum degree. J. Graph Theory, 63, 185-191.
13. L. Fausett, " Fundamental of neural networks architectures, algorihms and applications," pp.6.
14. Li, H., Li, X. Liu, S. (2012). Rainbow connection of graphs with diameter 2. Discrete Math.,312, 1453-1457.
15. Li, H., Li, X. Sun, Y. (2017). Rainbow connection number of graphs with diameter 3. Discuss. Math. Graph Theory, 37, 141-154.
16. Li, X. Shi, Y. (2013). Rainbow connection in 3-connected graphs. Graphs Combin, 29, 1471-1475.
17. Li, X. Sun, Y. (2017). An updated survey on rainbow connections of graphs - a dynamic survey. Theory Appl. Graphs, 0 (1) Article 3.
18. Simanjuntak, R., Bertault, F. Miller, M. (2000). Two new (a, d)-antimagic graph labelings. Proc. of Eleventh Australasian Workshop on Combinatorial Algorithms, 11, 179–189.