STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **15**, June 2026, pp 5148–5169.
Published online in International Academic Press (www.IAPress.org)

The Implementation of Artificial Neural Networks and Resolving Efficient Dominating Set for Time Series Forecasting on Vertical Farming Soil Moisture

D.E.W. Meganingtyas[1], Dafik [2,3,*], I.H. Agustin [2,3], R.I. Baihaki [2], Z.R. Ridlo [4], A.B. Angrenani [5], R. Nisviasari [2]

[1]*Department of Mathematics, State University of Jakarta, Indonesia*
[2]*PUI-PT Combinatorics and Graph, CGANT-University of Jember, Indonesia*
[3]*Department of Mathematics, University of Jember, Indonesia*
[4]*Department of Natural Science Education, University of Jember, Indonesia*
[5]*Department of Mathematics Education, University of Jember, Indonesia*

Abstract Vertical farming is a method of growing crops in which cultivated plants are arranged vertically using soil media within controlled indoor environments known as Controlled Environment Agriculture (CEA). Managing vertical farming systems requires precise regulation of air temperature, humidity, light intensity, and soil moisture levels. In recent years, Artificial Neural Networks (ANN) have emerged as powerful models for forecasting and decision support in precision agriculture. This study integrates ANN with a Resolving Efficient Dominating Set (REDS) graph optimization framework to enhance both data acquisition and forecasting efficiency in soil-moisture prediction. A total of 864 observations of temperature, air humidity, and soil moisture were recorded in four daily phases in an indoor *Aloe vera* vertical farm using DHT11 and capacitive soil moisture sensors. Four ANN models: Feedforwardnet, Patternnet, Fitnet, and Cascadeforwardnet, and two ANN architectures: ANN-466 and ANN-567, were trained and tested under a 70%-30% data split. The results show that the best architecture for Phase 1 was the Feedforwardnet ANN-567 model, with an MSE of 0.3810. The best architecture for Phase 2 was the Patternnet ANN-567 model, with an MSE of 1.13×10^{-9}. The best architecture for Phase 3 was the Cascadeforwardnet ANN-567 model, with an MSE of 1.12×10^{-10}. Finally, the best architecture for Phase 4 was the Cascadeforwardnet ANN-567 model, with an MSE of 1.07×10^{-17}. The integration of REDS and ANN establishes a cohesive and reproducible framework for precision irrigation management in CEA systems, offering spatial efficiency in sensor deployment and temporal accuracy in soil-moisture forecasting.

Keywords Artificial Neural Networks, Resolving Efficient Dominating Set, Soil Moisture, Time Series Forecasting, Vertical Farming

DOI: 10.19139/soic-2310-5070-3004

1. Introduction

Vertical farming is the practice of growing crops in which the cultivated plants are grown vertically rather than horizontally using soil media [1, 2], see Figure 1. It can be integrated into various structures (such as skyscrapers, disused warehouses, or containers), or in recent technology, agriculture is vertically developed specifically in a particular room by using Controlled Environment Agriculture (CEA) technology [3, 4]. Since various plant combinations can be cultivated in vertical farming, the use of CEA technology is inevitable because the control mechanisms of vertical farming are more complex, especially those related to temperature, relative humidity, light intensity, and soil moisture.

*Correspondence to: Dafik (Email: d.dafik@unej.ac.id). Department of Mathematics, University of Jember. 37 Kalimantan Road, Jember, East Java Province, Indonesia (68131).

ISSN 2310-5070 (online) ISSN 2311-004X (print)

The utilization of IoT technology equipped with Artificial Intelligence (AI) and Machine Learning (ML) is an effective tool for increasing the effectiveness of the control management system in vertical farming. An essential system in machine learning itself is the Artificial Neural Network (ANN). ANN is a computing system that adopts the thought process of the human brain, which is capable of providing stimulation, performing processing, and finally producing output [5, 6], see Figure 2 for illustration. ANN was first introduced in 1943 by Warren McCulloch and Walter Pitts and has grown very rapidly in the current era of disruptive technology [7].

Figure 1. The Illustration of vertical farming design

There are three types of machine learning, namely supervised learning, unsupervised learning, and reinforcement learning [8, 9]. Supervised learning is a type of ML where machines learn the relationship between input data and target data [10, 11]. The main purpose of supervised learning is to perform regression, which produces real number outputs, or classification, which produces categorical outputs such as true or false or RGB spectrum[12]. It is not necessary to have large input data sources to obtain good results in supervised learning. Unsupervised learning is a type of ML in which the machine learns the relationship among input data without target data provided. One of the goals of unsupervised learning is to perform clustering[13, 14]. Big data is required to produce good outputs from unsupervised learning. Reinforcement learning is a machine learning approach based on rewards and punishments[15, 16]. The goal of reinforcement learning is to generate patterns in game simulations. Big data is required to obtain accurate pattern simulations.

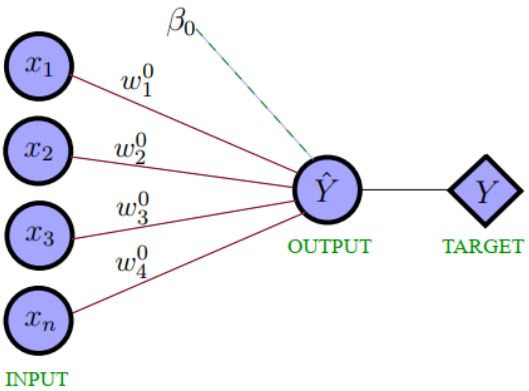

Figure 2. The Illustration of simple neural networks architecture

Let $W = (w_{ij})$ be a matrix used to store the connection weights for a neural network. The network input to unit Y_j (with no bias on unit j) is simply the dot product of the vectors $\mathbf{x} = (x_1, x_2, x_3, \ldots, x_n)$ and $\mathbf{w_{.j}}$ (the jth column of the weight matrix), namely $\hat{Y} = \mathbf{x} \cdot \mathbf{w_{.j}} = \sum_{i=1}^{n} x_i w_{ij}$, see Figure 2 for an illustration of a simple neural network. A bias can be included by adding a component $x_0 = 1$ to the vector $\mathbf{x}$, namely $\mathbf{x} = (1, x_1, x_2, x_3, \ldots, x_n)$.

The bias is treated exactly like any other weight, i.e., $w_{0j} = b_j$. The network input to unit Y_j is given by

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

The basic operation of neural networks is to sum the product of weights and input signals and then apply the activation function. Several types of activation functions are: (*i*) Linear activation function $\hat{Y} = f(x) = ax + b$, when $a = 1, b = 0$, it becomes an identity function; (*ii*) Binary step activation function with threshold θ, $\hat{Y} = f(x) = \begin{cases} 1 & \text{if } x \geq \theta, \\ 0 & \text{if } x < \theta, \end{cases}$ (*iii*) Binary sigmoid activation function $\hat{Y} = f(x) = \frac{1}{1+e^{-\sigma x}}$, and $f'(x) = \sigma f(x)[1 - f(x)]$, (*iv*) Bipolar sigmoid activation function $\hat{Y} = g(x) = 2f(x) - 1 = \frac{1-e^{-\sigma x}}{1+e^{-\sigma x}}$, and $g'(x) = \frac{\sigma}{2}[1 + g(x)][1 - g(x)]$, (*v*) Hyperbolic tangent activation function (Tanh) $\hat{Y} = h(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, and $h'(x) = [1 + h(x)][1 - h(x)]$.

Furthermore, let $V(G)$ and $E(G)$ be, respectively, the vertex and edge sets of a connected graph G. A subset D of $V(G)$ is an efficient dominating set of graph G if each vertex in G is either in D or adjacent to a vertex in D. A subset W of $V(G)$ is a resolving set of G if any vertex in G is uniquely distinguished by its representation with respect to every vertex in an ordered set W. Let $W = \{w_1, w_2, w_3, \ldots, w_k\}$ be a subset of $V(G)$. The representation of vertex $v \in V(G)$ with respect to an ordered set W is $r(v|W) = \{d(v, w_1), d(v, w_2), \ldots, d(v, w_k)\}$. The set W is called a resolving set of G if $r(u|W) \neq r(v|W)$ for $u, v \in V(G)$. A subset Z of $V(G)$ is called the resolving efficient dominating set of graph G if it is an efficient dominating set and $r(u|Z) \neq r(v|Z)$ for $u, v \in V(G)$. The minimum cardinality of the resolving efficient dominating set is denoted by $\gamma_{re}(G)$. Some results on the resolving efficient dominating set of graph G can be found in [17, 18].

Artificial Neural Networks (ANN) have proven effective in modeling non-linear environmental processes such as soil-moisture fluctuations[19, 20]. However, their predictive accuracy depends strongly on the spatial configuration and data quality of sensor networks. Graph-theoretical approaches, particularly the Resolving Efficient Dominating Set (REDS), provide a mathematical basis for optimizing sensor node placement to ensure maximum coverage with minimal redundancy[61]. The REDS can be applied to identify the most efficient and strategic locations for placing sensors within a vertical farming structure. By using the REDS algorithm, the system can determine an optimal dominating set that minimizes the number of sensor nodes required while still ensuring complete monitoring coverage of the cultivation area. This paper unifies these concepts by employing REDS to design an optimal sensor topology for a vertical farming prototype and training ANN models on data collected from those optimized nodes. The integration of topology optimization and machine learning establishes a closed-loop framework for reliable, resource-efficient precision farming.

The aim of this research is to study simple artificial neural network architectures mathematically, analyze the error, and analyze how their bias and weight are updated. Through this research, we also trained and tested several independent input datasets related to the temperature, air humidity, and soil moisture of an *Aloe vera* plantation obtained from DHT11 and capacitive soil-moisture sensors. *Aloe vera* is a species of plant with thick, fleshy leaves belonging to the genus *Aloe*. *Aloe vera* grows in dry subtropical climates such as those of South Africa, India, and several other regions of the world with similar conditions. This plant has essential roles in traditional and modern culture as a source of medicine, food, and cosmetics [21, 22]. *Aloe vera* as a traditional medicine and cosmetic ingredient has a strong laxative effect derived from its leaf gel [23, 24]. Further research shows that *Aloe vera* can be used as a treatment for ulcers, coughs, gastritis, cancer, headaches, and immune system deficiencies [25, 26, 27]. Thus, cultivating *Aloe vera* provides potential business opportunities in agriculture.

2. Methods

The type of this research is analytical and experimental. In general, the analytical study uses mathematical analysis to explore the findings. By combining analytical and experimental approaches, the research obtains comprehensive results. We analyzed the effectiveness of two ANN architectures (ANN-466 and ANN-567) across four ANN models (Feedforwardnet, Fitnet, Patternnet, Cascadeforwardnet) for soil moisture forecasting in vertical farming

by using programming analysis in term of iteration number, mean square error (MSE), correlation, and elapsed time.

The REDS algorithm is used to determine the optimal placement of sensor nodes in a comb structure representing a vertical farming environment by calculating the minimal set of sensor nodes capable of covering all observation points while maintaining complete spatial observability [28]. Each node is selected according to the specific sensor location that maximises data coverage and minimises redundancy [29]. The environmental data collected from these optimised REDS nodes is then used as training and testing input for the ANN model. This integration establishes an integrated REDS ANN workflow, where graph theory optimisation informs data collection, and ANN plays a role in machine learning analysis. The illustration of sensor placement using REDS is shown in Figure 3.

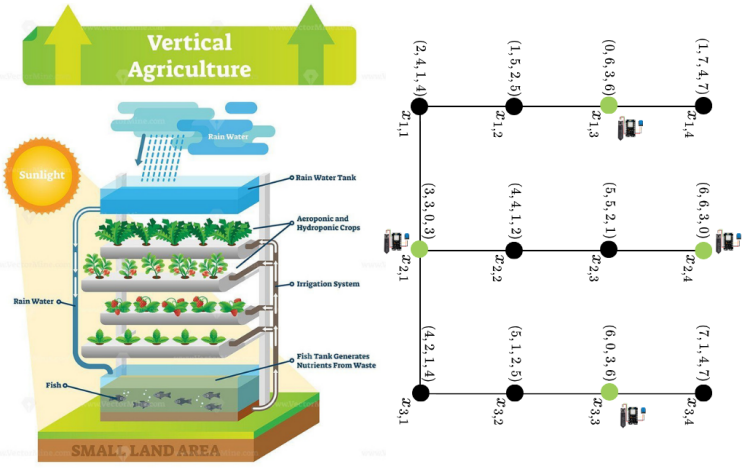

Figure 3. Sensor placement using REDS algorithm on Vertical Farming.

In the collecting data process, we combine two types of sensors, namely the DHT 11 and soil moisture sensor. The DHT11 sensor measures temperature and relative humidity[30, 31], the second sensor is a soil moisture sensor represent normalized capacitive sensor readings scaled to the range 0–100% where higher values indicate higher moisture content.[32, 33]. Both sensors are connected to a NodeMCU microcontroller, which transmits data to a cloud system for real time collection of temperature, relative humidity, and soil moisture levels in *Aloe vera* plants[34, 35].

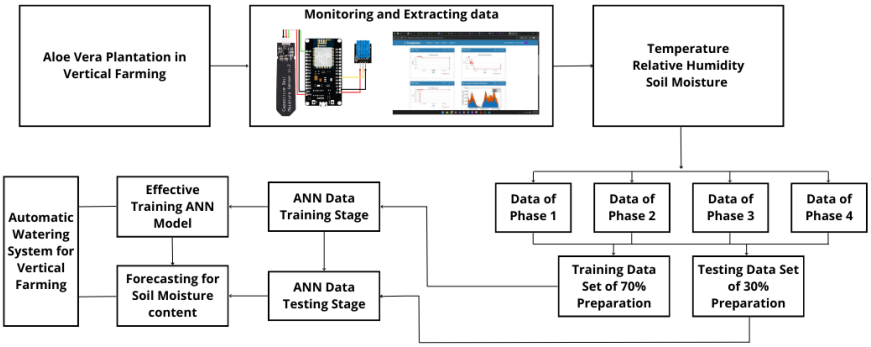

Figure 4. The forecasting process for determining the watering time on vertical farming.

The soil moisture level from the soil moisture sensor is used to activate the actuator to perform automatic watering based on a certain threshold value. However, air temperature and humidity are also analysed because these two parameters affect soil moisture levels[69, 70]. The placement of sensors using the REDS algorithm

is not merely theoretical, we incorporated physical limitation parameters into the vertical farming system we created. These parameters include the distance between plants (15–20 cm) and the range of humidity sensors (20–30 cm) [58, 59, 60]. The Euclidean distance between each selected sensor node and its dominated node was calculated, and all distances were within the 15 cm range, confirming that each dominated node was indeed within the actual sensing range. Thus, each selected node met the REDS dominance and efficiency criteria under real spatial constraints. Each sensor records data and sends the results every minute continuously for 21 days, resulting in a total of 864 observations during four daily phases.

Data acquired from these optimally located sensors were then fed into the ANN models to forecast soil-moisture dynamics across the defined daily phases. This hybrid structure ensures that the predictive modelling process directly benefits from mathematically optimized sensor deployment, thereby enhancing both data efficiency and forecasting accuracy. All raw sensor readings were normalized to a range of [0.1, 0.9] before training to stabilize the learning process and prevent saturation of sigmoid based activation functions [36, 37, 38]. Missing or inconsistent values were removed through linear interpolation. The normalized dataset was divided into 70% for training and 30% for testing in accordance with the forecasting design used in previous studies on environmental time-series modelling, the illustration of ANN forecasting process show on Figure 4. To develop a robust ANN model, we used the following multilayer Perceptron algorithm.

Perceptron Algorithm 1

Step 0. Initialize weights $\mathbf{w}$ and bias β. (For simplicity, set $\mathbf{w} = \beta = 0$.)
Set learning rate α $(0 < \alpha < 1)$. (For simplicity, set $\alpha = 1$.)
Step 1. While the stopping condition is false, do Steps 2{6.
Step 2. For each training pair S, Y, do Steps 3{5.
Step 3. Set activations of input units: $x_i = s_i \in S$.
Step 4. Compute response of output unit: $\hat{Y} = \beta_0 + \sum_{i=1}^{n} x_i w_i$.
Use the above activation functions with threshold θ.
Step 5. Update weights and bias. If $\hat{Y} \neq Y$, do the following:
$W_{\text{new}} = W_{\text{old}} + \alpha Y \mathbf{x}$, $\beta_{\text{new}} = \beta_{\text{old}} + \alpha Y$
Else, $W_{\text{new}} = W_{\text{old}}$, $\beta_{\text{new}} = \beta_{\text{old}}$.
Step 6. Test the stopping condition.
Step 7. If no weights changed in Step 5, stop; else, continue.

For each neural-network experiment, four ANN models were evaluated: Feedforwardnet, Patternnet, Fitnet, and Cascadeforwardnet. The four ANN models were selected because each offers a distinct structural approach to nonlinear modeling[62, 63, 64]. Feedforward networks provide the simplest baseline with strictly layered connections [65]. Fitnet is optimized for function-fitting, making it suitable for continuous soil-moisture regression[66]. Patternnet incorporates pattern-recognition heuristics that help capture temporal structures in the data[67]. Cascadeforward networks extend the standard feedforward design by adding direct connections from inputs to all subsequent layers, enabling faster learning of complex nonlinear relationships[68]. All models were trained under identical hyperparameter settings, including learning rate, optimizer, activation functions, epoch count, validation split, and batch size. This controlled setup ensured that performance differences could be attributed to architectural differences rather than inconsistent training conditions. Each network used the `Adam` training function and `linear` activation in the hidden layer. The same dataset partition was applied to all models for comparability[39, 40]. The architecture code "ANN-466" denotes 4 input neurons, 6 hidden neurons, and 6 output neurons predicting soil-moisture values for the next six time steps, and the architecture code "ANN-567" denotes 5 input neurons, 6 hidden neurons, and 7 output neurons predicting soil-moisture values for the next seven time steps. Performance was measured using mean square error (MSE), mean absolute error (MAE), root mean square error (RMSE), and coefficient of determination (R^2)[40].

The selection of the number of neurons in each hidden layer with patterns [4–6–6] and [5–6–7] in each ANN architecture and model is based on the theory that models with small, medium, and high layer configurations

provide optimal performance for multivariate time series forecasting [41, 42]. Neuron composition patterns with simple but moderate-capacity architectures tend to be more stable in the computational process and have better generalisation compared to structures that use overly large neuron compositions[43, 44]. Furthermore, Rachmatullah et al. (2021) recommend a progressive expansion structure, where the number of neurons in each hidden layer follows the increasing complexity of the main PCA components, enabling the sequential learning of basic feature representations, interactions between variables, and advanced non-linear patterns[45].

The optimal architecture for multivariate time series generally follows a small → medium → medium-large configuration, which has a significant impact on achieving an ideal balance between model capacity and training stability based on Neural Evolutionary Architecture Search (2023)[46, 47, 48]. Furthermore, research on MLP optimisation indicates that using three hidden layers with a moderate number of neurons (5–10 units) yields the best performance[49, 50]. In this study, temperature, air humidity, and soil moisture data are non-linear data. Therefore, configurations [4–6–6] and [5–6–7] were selected as ideal and consistent compositions to provide adequate learning capacity without causing overfitting or training instability.

In the ANN research stage, this study was divided into three phases: training, testing, and forecasting. The training phase aimed to obtain an effective ANN model. The testing phase aimed to measure the accuracy of the trained learning model. The model obtained was then used for forecasting to support decision making. In this study, the dataset consisted of 864 records divided into four time phases: (1) 06:00–09:00, (2) 09:00–12:00, (3) 12:00–15:00, and (4) 15:00–18:00. Each phase consists of 12 time segments.

3. Findings

3.1. Mathematical Analysis

Recall Equation 1, and let $\hat{Y}$ be an activation function of very basic neural network with no hidden layer, the network input to unit Y is given by $\hat{Y} = \beta_0 + \sum_{i=1}^{n} x_i w_i$, where β_0 and $w_i, i = 1, 2, \ldots, n$ are the bias and the elements of weight W, respectively, see Figure 2. The five popular activation functions are *(i)* Linear activation function $\hat{Y} = \beta_0 + \sum_{i=1}^{n} w_i x_i$, *(ii)* Binary step activation function with threshold θ is $\hat{Y} = \begin{cases} 1, & \text{if } f(x) \geq \theta \\ 0, & \text{if } f(x) < \theta \end{cases}$ *(iii)* Binary sigmoid activation function $\hat{Y} = \frac{1}{1+e^{-(\beta_0+\sum_{i=1}^{n} w_i x_i)}}$, *(iv)* Bipolar sigmoid activation function $\hat{Y} = \frac{1-e^{-(\beta_0+\sum_{i=1}^{n} w_i x_i)}}{1+e^{-(\beta_0+\sum_{i=1}^{n} w_i x_i)}}$, *(v)* Hyperbolic tangent activation function (Tanh) $\hat{Y} = \frac{e^{(\beta_0+\sum_{i=1}^{n} w_i x_i)} - e^{-(\beta_0+\sum_{i=1}^{n} w_i x_i)}}{e^{(\beta_0+\sum_{i=1}^{n} w_i x_i)} + e^{-(\beta_0+\sum_{i=1}^{n} w_i x_i)}}$.

Given that a simple neural network architecture with no hidden layer in Figure 2. How do we derive the mathematical activation function? Straightforwardly, suppose the input and the weight are $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ and $w_i^0 = \{w_1^0, w_2^0, \ldots, w_n^0\}$. Thus the activation function are in the form of scalar as follows:

$$
\begin{aligned}
\hat{Y}_{linear} &= \beta_0 + w_1^0 x_1 + w_2^0 x_2 + \cdots + w_n^0 x_n \\
\hat{Y}_{Binary} &= \begin{cases} 1, & \text{if } \beta_0 + w_1^0 x_1 + w_2^0 x_2 + \cdots + w_n^0 x_n \geq \theta \\ 0, & \text{if } \beta_0 + w_1^0 x_1 + w_2^0 x_2 + \cdots + w_n^0 x_n < \theta \end{cases} \\
\hat{Y}_{Binary\ sigmoid} &= \frac{1}{1 + e^{-(\beta_0 + w_1^0 x_1 + w_2^0 x_2 + \cdots + w_n^0 x_n)}} \\
\hat{Y}_{Bipolar\ sigmoid} &= \frac{1 - e^{-(\beta_0 + w_1^0 x_1 + w_2^0 x_2 + \cdots + w_n^0 x_n)}}{1 + e^{-(\beta_0 + w_1^0 x_1 + w_2^0 x_2 + \cdots + w_n^0 x_n)}} \\
\hat{Y}_{Tanh} &= \frac{e^{(\beta_0 + w_1^0 x_1 + w_2^0 x_2 + \cdots + w_n^0 x_n)} - e^{-(\beta_0 + w_1^0 x_1 + w_2^0 x_2 + \cdots + w_n^0 x_n)}}{e^{(\beta_0 + w_1^0 x_1 + w_2^0 x_2 + \cdots + w_n^0 x_n)} + e^{-(\beta_0 + w_1^0 x_1 + w_2^0 x_2 + \cdots + w_n^0 x_n)}}
\end{aligned}
$$

When we evaluate $\hat{Y} = \beta_0 + \sum_{i=1}^{n} x_i w_i$, the assignment of value to the weight commonly brings the major impact towards the learning behaviour of the network. If the algorithm successfully computes the correct value of the weight, it can converge faster to the solution, otherwise, the convergence might be slower or it might

cause divergence. To prevent this problem occurring, the step of gradient descent is controlled by a parameter called the learning rate, denoted by α. We use the gradient descent method to make an informed step change in W. This is an iterative method that involves multiple steps. Each time, the W value is updated according to $W_{new} = W_{old} - \alpha \cdot \bar{\nabla} error$. Here W_{new} denotes the new W position, W_{old} denotes the current W, $\bar{\nabla} error$ is the gradient of the error at W_{old}. The learning rate α will determine how quickly the solution converges on the minimum error. In the simple form we can write the updated weight and bias in the following: Perceptron and Hebb algorithms with learning rate α, we have $W_{new} = W_{old} + \alpha(Y - \hat{Y})\mathbf{x} \approx W_{old} + \alpha Y\mathbf{x}$, and $\beta_{new} = \beta_{old} + \alpha(Y - \hat{Y}) \approx \beta_{old} + \alpha Y$. In this results, we extend the architecture by adding the hidden layers: one, two or more with specific number of nodes per layer, the Illustration of ANN with one layer of two nodes show on Figure 5. This extension will be given in an observation.

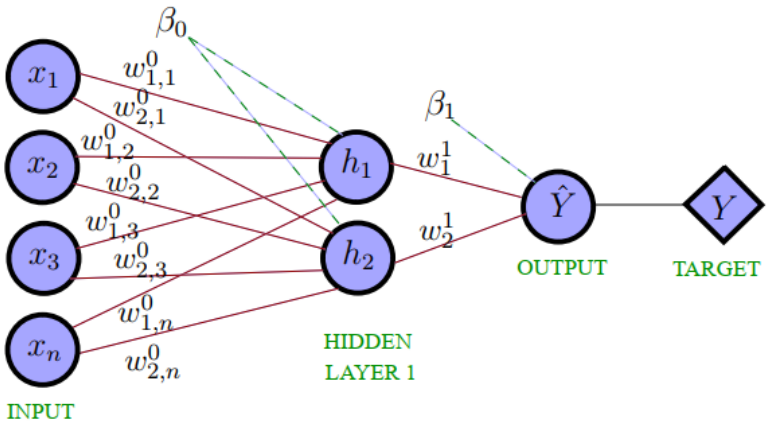

Figure 5. The Illustration of ANN with one layer of two nodes

Observation 1
Given that a neural network architecture with one hidden layer of two nodes. Let $\mathbf{x} = \{x_i | i = 1, 2, \ldots, n\}$ as the input data. Let h and β be the hidden layer and the bias respectively. The linear activation function can be written as $\hat{Y}_{linear} = \beta_1 + \sum_{i=1}^{2} w_i^1 h_i$, where $[h_1; h_2] = \beta_0 + W^0\mathbf{x}$.

Proof. The input data is $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$. Since we have one layer of two nodes, we define the two sets of weight as $W^0 = [w^0_{1,1}\ w^0_{1,2}\ \ldots\ w^0_{1,n}; w^0_{2,1}\ w^0_{2,2}\ \ldots\ w^0_{2,n}]$, $W^1 = [w^1_1; w^1_2]$ and the bias β_0, β_1. The neural network architecture can be depicted in Figure 5. By this architecture, we can derive the linear activation functions as $\hat{Y}_{linear} = \beta_1 + w^1_1 h_1 + w^1_2 h_2 = \beta_1 + \sum_{i=1}^{2} w^1_i h_i$, where

$$\begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = \begin{bmatrix} \beta_{0,1} \\ \beta_{0,2} \end{bmatrix} + \begin{bmatrix} w^0_{1,1} & w^0_{1,2} & \ldots & w^0_{1,n} \\ w^0_{2,1} & w^0_{2,2} & \ldots & w^0_{2,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \beta_0 + W^0 \times \mathbf{x} \tag{2}$$

It concludes the proof. □

By using Equation 2, it is easy to derive that the other activation functions are as follows

$$\begin{aligned}
\hat{Y}_{Binary} &= \begin{cases} 1, & \text{if } \beta_1 + \sum_{i=1}^{2} w_i^1 h_i \geq \theta \\ 0, & \text{if } \beta_1 + \sum_{i=1}^{2} w_i^1 h_i < \theta \end{cases} \\
\hat{Y}_{Binary\ sigmoid} &= \frac{1}{1 + e^{-(\beta_1 + \sum_{i=1}^{2} w_i^1 h_i)}} \\
\hat{Y}_{Bipolar\ sigmoid} &= \frac{1 - e^{-(\beta_1 + \sum_{i=1}^{2} w_i^1 h_i)}}{1 + e^{-(\beta_1 + \sum_{i=1}^{2} w_i^1 h_i)}} \\
\hat{Y}_{Tanh} &= \frac{e^{(\beta_1 + \sum_{i=1}^{2} w_i^1 h_i)} - e^{-(\beta_1 + \sum_{i=1}^{2} w_i^1 h_i)}}{e^{(\beta_1 + \sum_{i=1}^{2} w_i^1 h_i)} + e^{-(\beta_1 + \sum_{i=1}^{2} w_i^1 h_i)}}
\end{aligned}$$

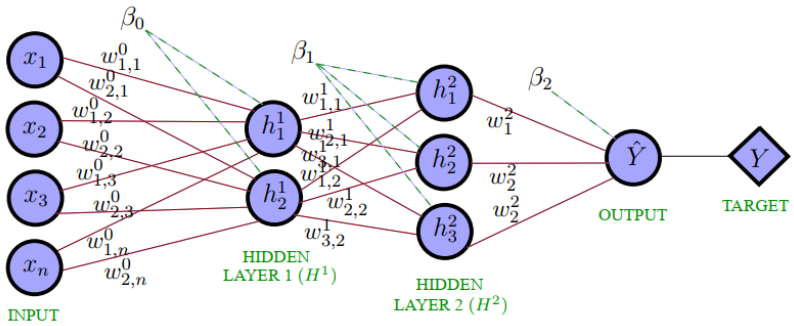

Figure 6. The Illustration of ANN architecture with two hidden layers

Observation 2
Given that a neural network architecture with two hidden layer of respectively two and three nodes. Let h and β be the hidden layer and the bias respectively. The linear activation function can be written as $\hat{Y}_{linear} = \beta_2 + \sum_{i=1}^{3} w_i^2 h_i^2$, where $[h_1^2; h_2^2; h_3^2] = \beta_1 + W^1 \times \mathbf{h}^1$, and $[h_1; h_2] = \beta_\mathbf{0} + W^0 \mathbf{x}$.

The Illustration of ANN architecture with two hidden layers show on Figure 6.

Proof. The input data is $\mathbf{x} = \{x_1, x_2, \ldots, x_4\}$. Since we have two hidden layer of respectively two and three nodes, we define the two sets of weight as $W^0 = [w_{1,1}^0\ w_{1,2}^0\ \ldots\ w_{1,n}^0; w_{2,1}^0\ w_{2,2}^0\ \ldots\ w_{2,n}^0]$, $W^1 = [w_{1,1}^1\ w_{1,2}^1; w_{2,1}^1\ w_{2,2}^1; w_{3,1}^1\ w_{3,2}^1]$, $W^2 = [w_1^2; w_2^2; w_3^2]$ and the bias $\beta_0, \beta_1, \beta_2$. The neural network architecture can be depicted in Figure 6. By this architecture, we can derive the linear activation function as $\hat{Y}_{linear} = \beta_2 + w_1^2 h_1^2 + w_2^2 h_2^2 + w_3^2 h_3^2 = \beta_2 + \sum_{i=1}^{3} w_i^2 h_i^2$, where

$$\begin{bmatrix} h_1^2 \\ h_2^2 \\ h_3^2 \end{bmatrix} = \begin{bmatrix} \beta_{1,1} \\ \beta_{1,2} \\ \beta_{1,3} \end{bmatrix} + \begin{bmatrix} w_{1,1}^1 & w_{1,2}^1 \\ w_{2,1}^1 & w_{2,2}^1 \\ w_{3,1}^1 & w_{3,2}^1 \end{bmatrix} \begin{bmatrix} h_1^1 \\ h_2^1 \end{bmatrix} = \beta_1 + W^1 \times \mathbf{h}^1$$

and

$$\begin{bmatrix} h_1^1 \\ h_2^1 \end{bmatrix} = \begin{bmatrix} \beta_{0,1} \\ \beta_{0,2} \end{bmatrix} + \begin{bmatrix} w_{1,1}^0 & w_{1,2}^0 & \ldots & w_{1,n}^0 \\ w_{2,1}^0 & w_{2,2}^0 & \ldots & w_{2,n}^0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \beta_0 + W^0 \times \mathbf{x} \tag{3}$$

It concludes the proof. □

Again, refer to Equation 3, it is easy to derive that the other activation functions are as follows

$$\begin{aligned}
\hat{Y}_{Binary} &= \begin{cases} 1, & \text{if } \beta_2 + \sum_{i=1}^{3} w_i^2 h_i^2 \geq \theta \\ 0, & \text{if } \beta_2 + \sum_{i=1}^{3} w_i^2 h_i^2 < \theta \end{cases} \\
\hat{Y}_{Binary\ sigmoid} &= \frac{1}{1 + e^{-(\beta_2 + \sum_{i=1}^{3} w_i^2 h_i^2)}} \\
\hat{Y}_{Bipolar\ sigmoid} &= \frac{1 - e^{-(\beta_2 + \sum_{i=1}^{3} w_i^2 h_i^2)}}{1 + e^{-(\beta_2 + \sum_{i=1}^{3} w_i^2 h_i^2)}} \\
\hat{Y}_{Tanh} &= \frac{e^{(\beta_2 + \sum_{i=1}^{3} w_i^2 h_i^2)} - e^{-(\beta_2 + \sum_{i=1}^{3} w_i^2 h_i^2)}}{e^{(\beta_2 + \sum_{i=1}^{3} w_i^2 h_i^2)} + e^{-(\beta_2 + \sum_{i=1}^{3} w_i^2 h_i^2)}}
\end{aligned}$$

To have more insight on the neural network architecture above, we will illustrate specific illustrations of the architectural calculation for specific input of data and put it on the Appendix. The illustration will cover the three of the architectures, namely the neural network with no hidden layer, neural network with one hidden layer of two nodes, neural network with two hidden layers of two and three nodes each. If the scale of the features input data are not the same, then we need to normalize the input data by using the formula $\mathbf{x}' = a + \frac{(x_i - x_{min})(b-a)}{x_{max} - x_{min}}$, where the feature scaling will bring all elements of $\mathbf{x}$ into the range $[a, b]$.

By using the linear activation function, we have $\hat{Y}^2_{linear} = \hat{Y}^2$. Since $\hat{Y}^2_{linear} \neq Y$, again we need to update W and β before continuing to calculate the linear activation function for $\mathbf{x} = [0.557 \ \ 0.260]$ and $\mathbf{x} = [0.900 \ \ 0.900]$. Once we have calculated for all input data $\mathbf{x}$, we can continue to the second iteration by the same manner. The Illustration of ANN architecture with three hidden layers show on Figure 7. □

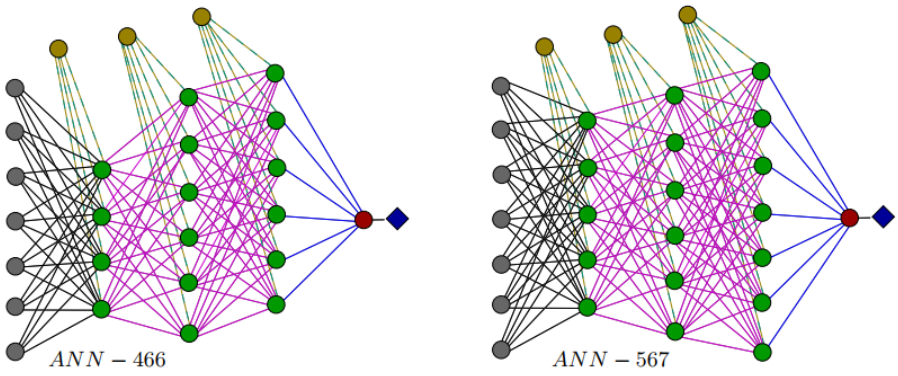

Figure 7. The Illustration of ANN architecture with three hidden layers

3.2. Data Training, Testing and Forecasting

From now on, we will do some computer simulation on two ANN architectures across four ANN models to train, test and forecast the data on vertical farming issue namely temperature, air humidity and soil moisture. We then run the Matlab programming which has been developed in this study to test effectiveness of two ANN architectures across four ANN models, see Figure 8, and compare the effectiveness in term of ANN Models, Regression and Mean Square Error (MSE). In this simulation, we trained 70% and tested the 30% of data set derived from four phases in which each phase reflects of 12 time segments, see Table 1. We finally also did forecasting for three days ahead on temperature, air humidity and soil moisture. The illustration shows that ANN output interpolates uniformly on the data target either in training and testing stage, see Figure 8. The illustration shows that the next watering time of aloe vera plantation is 3 days after the last watering, see Figure 9.

		Phase 1			
ANN Model	ANN Architecture	MSE Train	Regression Train	Time Train	MSE Test
Feedforwardnet	ANN-(a)466	3.0653×10^{-11}	1	0.7668 s	8.7061
	ANN-(a)567	9.9429×10^{-11}	1	0.64381 s	**0.3810**
Fitnet	ANN-(b)466	6.3786×10^{-10}	1	3.1013 s	40.6900
	ANN-(b)567	2.1076×10^{-15}	1	0.49911 s	3.1096
Patternnet	ANN-(c)466	2.3981×10^{-9}	1	3.4498 s	5.6989
	ANN-(c)567	4.4919×10^{-10}	1	3.1707 s	0.7745
Cascadeforwardnet	ANN-(d)466	5.3794×10^{-15}	1	0.62146 s	0.5250
	ANN-(d)567	5.2313×10^{-23}	1	0.56244 s	2.7204
		Phase 2			
ANN Model	ANN Architecture	MSE Train	Regression Train	Time Train	MSE Test
Feedforwardnet	ANN-(a)466	1.3344×10^{-12}	1	0.97366 s	1.8924×10^{-17}
	ANN-(a)567	1.0639×10^{-13}	1	0.4682 s	8.6779×10^{-15}
Fitnet	ANN-(b)466	1.0453×10^{-11}	1	0.6675 s	1.4498×10^{-11}
	ANN-(b)567	6.2623×10^{-17}	1	5.4094 s	8.8313×10^{-17}
Patternnet	ANN-(c)466	5.1985×10^{-10}	1	2.1392 s	4.7759×10^{-10}
	ANN-(c)567	1.7470×10^{-9}	1	1.6956 s	**1.1374** $\times 10^{-9}$
Cascadeforwardnet	ANN-(d)466	1.1027×10^{-18}	1	0.36338 s	8.6681×10^{-20}
	ANN-(d)567	2.0088×10^{-15}	1	0.36098 s	2.1493×10^{-15}
		Phase 3			
ANN Model	ANN Architecture	MSE Train	Regression Train	Time Train	MSE Test
Feedforwardnet	ANN-(a)466	1.8038×10^{-12}	1	0.24812 s	1.7669×10^{-12}
	ANN-(a)567	1.9969×10^{-11}	1	0.28055 s	2.9162×10^{-11}
Fitnet	ANN-(b)466	3.0244×10^{-16}	1	0.27564 s	3.6732×10^{-16}
	ANN-(b)567	6.2623×10^{-17}	1	5.4094 s	8.8313×10^{-17}
Patternnet	ANN-(c)466	9.6169×10^{-10}	1	9.0231 s	4.5504×10^{-10}
	ANN-(c)567	9.9638×10^{-11}	1	0.68983 s	1.1803×10^{-10}
Cascadeforwardnet	ANN-(d)466	7.6943×10^{-16}	1	0.56864 s	9.9713×10^{-16}
	ANN-(d)567	8.6575×10^{-11}	1	0.34536 s	**1.1232** $\times 10^{-10}$
		Phase 4			
ANN Model	ANN Architecture	MSE Train	Regression Train	Time Train	MSE Test
Feedforwardnet	ANN-(a)466	1.1585×10^{-9}	1	2.2923 s	8.4517×10^{-15}
	ANN-(a)567	4.9126×10^{-11}	1	0.56431 s	2.7740×10^{-11}
Fitnet	ANN-(b)466	4.08684×10^{-13}	1	0.39173 s	4.2850×10^{-16}
	ANN-(b)567	4.9785×10^{-19}	1	0.33753 s	2.7021×10^{-19}
Patternnet	ANN-(c)466	8.4219×10^{-10}	-	0.3615 s	1.2577×10^{3}
	ANN-(c)567	7.6331×10^{-10}	1	2.2668 s	3.2403×10^{-10}
Cascadeforwardnet	ANN-(d)466	4.9783×10^{-16}	1	0.3542 s	5.9286×10^{-16}
	ANN-(d)567	7.7662×10^{-17}	1	0.34536 s	$\mathbf{1.0780 \times 10^{-17}}$

Table 1. The performance indicator of ANN architectures and models on training and testing agriculture data set.

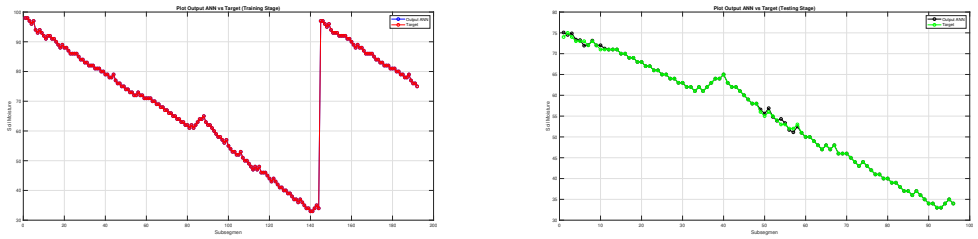

Figure 8. The comparison of ANN output and target data of the soil moisture on training and testing stages.

Some of the parameters used in this research are training functions Lavenberg Marquardt, sigmoid and hyperbolic tangent sigmoid function sigmoid tangent. In this study, the Artificial Neural Network was configured using a set of hyperparameters selected to ensure stable learning and minimize forecasting error. The ReLU activation function was used in the hidden layers to enhance nonlinear representation capability, while a linear activation was applied in the output layer to support continuous soil-moisture forecasting. Model training utilized the Adam optimizer, chosen for its adaptive learning-rate mechanism, with a fixed learning rate of 1×10^{-5} to prevent divergence and ensure gradual convergence[71].

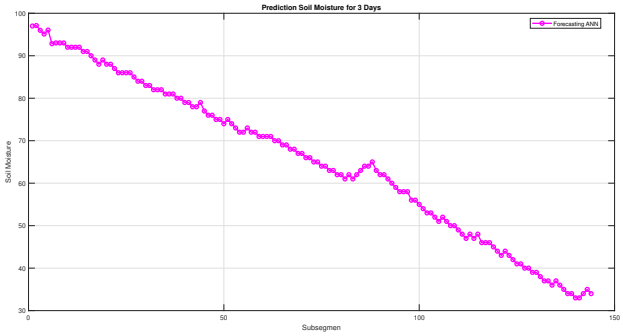

Figure 9. The forecasting of soil moisture for determining the watering time on aloe vera plantation.

The network was trained for 150 epochs, a number determined to be sufficient for achieving stable error reduction without inducing overfitting, while a batch size of 64 provided an effective balance between computational efficiency and gradient stability. To improve generalization, 30% of the data known as testing data were allocated for validation, and all input variables were normalized using a MinMax scaler, ensuring consistent feature scaling across the dataset. A random seed of 42 was set to maintain reproducibility of the results, and a lag parameter of 3 was applied to generate the temporal input windows required for multistep-ahead forecasting. The results of this study can be seen in Table 1. The selection of the best architecture is based on the smallest MSE value. Based on Table 1, the best architecture for phase 1 is obtained by Feedforwardnet ANN-567 model, with MSE of 0.3810. The best architecture for phase 2 is obtained by Patternnet ANN-567 model with an MSE of 1.1374×10^{-9}. The best architecture for phase 3 is obtained by Cascadeforwardnet ANN-567 model with an MSE of 1.1232×10^{-10}. Finally, the best architecture for phase 4 is obtained by model Cascadeforwardnet ANN-567 with an MSE of 1.0780×10^{-17}.

Apart from tabulating the indicator performance, we also show the graphics on training, testing and forecasting for three days ahead to know the watering time of aloe vera plantation. The results are shown in Figure 8 - Figure 13 regarded to the soil moisture, temperature and relative humidity. Based on Figure Figure 11, we know that for the next watering time of aloe vera plantation is 3 days after the last watering. The watering time should be carried

out in phase 4, namely on 16.45 - 17.15. The illustration from Figure 10 shows that the next watering time of aloe vera plantation is 3 days after the last watering with best temperature of 30^oC.

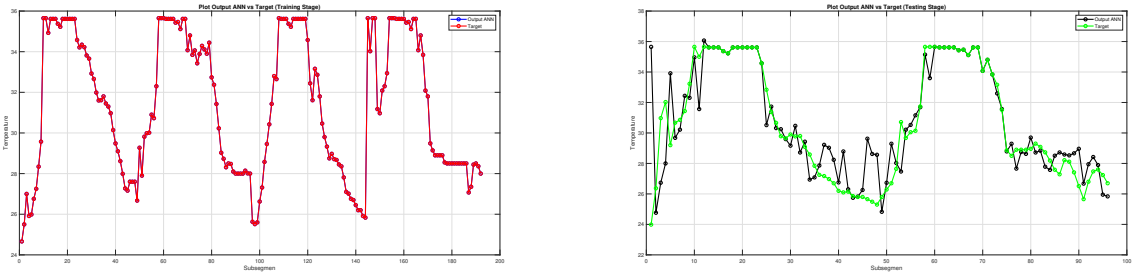

Figure 10. The comparison of ANN output and target data of the temperature on training and testing stage. The illustration shows that ANN output interpolates uniformly on the data target either in training and testing stage.

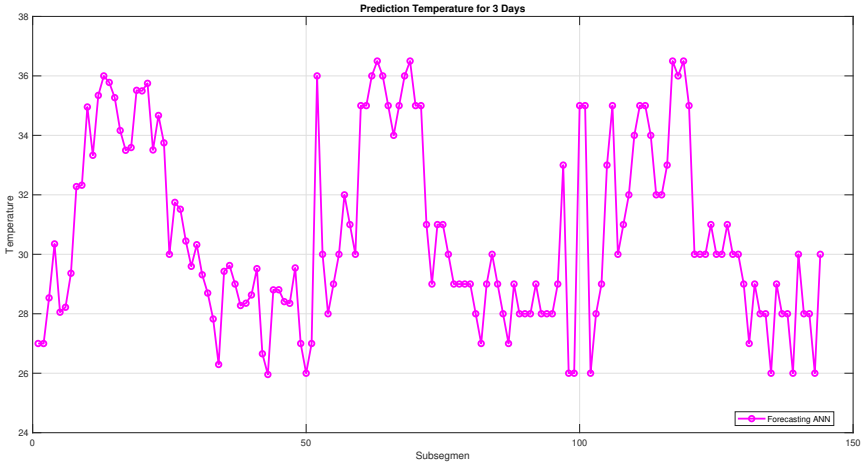

Figure 11. The forecasting of temperature for determining the watering time on aloe vera plantation.

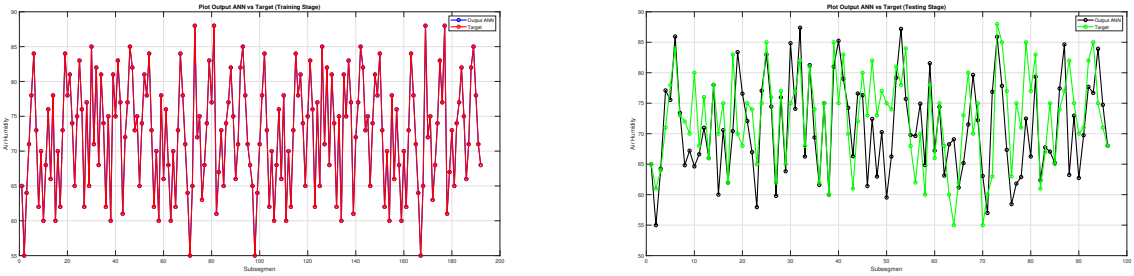

Figure 12. The comparison of ANN output and target data of the air humidity on training and testing stage. The illustration shows that ANN output interpolates uniformly on the data target either in training and testing stage.

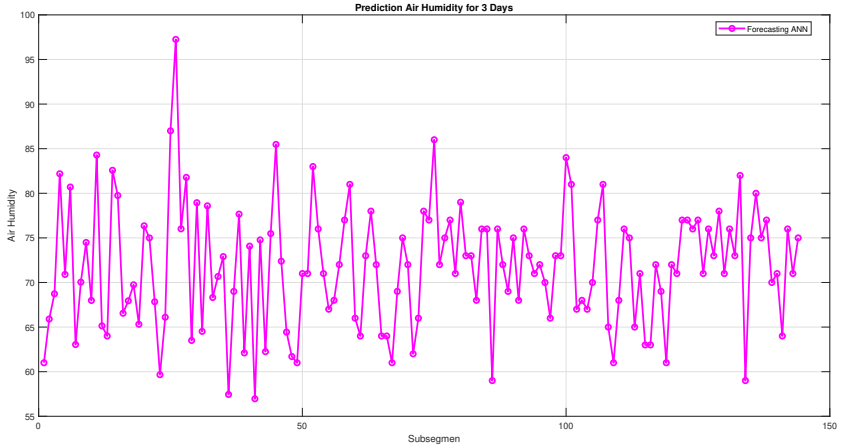

Figure 13. The forecasting of air humidity for determining the watering time on aloe vera plantation. The illustration shows that the next watering time of aloe vera plantation is 3 days after the last watering with best air humidity of 75%.

3.3. Resolving Efficient Dominating Set of Graphs

In this section, we will describe our new result of resolving efficient dominating set (REDS). This concept is very useful due to the planting topology can be presented in graph representative and the sensor placement to monitor the soil moisture should consider the planting topology otherwise it will be inefficient. The most effective concept in the sensor placement is using REDS[51, 52, 53]. There two REDS presented in this paper, namely REDS of distance one and two[54]. For the distance one (d=1), Kusumawardani *et. al.* has obtained a new results written in Theorem 1, and for the distance two (d=2), we show a new results written in Theorem 2 [55, 56, 57]. Note that the resolving efficient domination number of graph G of d=2 is denoted by γ_{re_2}.

Theorem 1
For every positive interger $m, n \geq 2$,[54]

$$\gamma_{re}(P_m \triangleright P_n) = \begin{cases} m\lceil \frac{n}{3} \rceil & \text{if } n \equiv 0 \ (mod\ 3) \text{ and } n \equiv 2 \ (mod\ 3) \\ m\lfloor \frac{n}{3} \rfloor + \lceil \frac{m}{3} \rceil & \text{if } n \equiv 1 \ (mod\ 3), m \equiv 1, 2 \ (mod\ 3) \\ \lceil \frac{m}{2} \rceil \lfloor \frac{n}{3} \rfloor + \lfloor \frac{m}{2} \rfloor \lceil \frac{n}{3} \rceil & \text{if } n \equiv 1 \ (mod\ 3), m \equiv 0 \ (mod\ 3). \end{cases}$$

Theorem 2
For every positive interger $m, n \geq 2$,[54]

$$\gamma_{re_2}(P_m \triangleright P_n) = \begin{cases} \lceil \frac{m}{2} \rceil & \text{if } n = 2 \\ m\lceil \frac{n}{5} \rceil & \text{if } n \equiv 3 \ (mod\ 5) \text{ and } m = 2 \\ m\lceil \frac{n}{5} \rceil & \text{if } n \equiv 0, 3, 4 \ (mod\ 5) \text{ and } m > 2 \\ m\lceil \frac{n}{7} \rceil + 1 & \text{if } n \equiv 0, 1, 2, 4 \ (mod\ 5) \text{ and } m = 2 \\ m\lfloor \frac{n}{5} \rfloor + \lceil \frac{m}{5} \rceil & \text{if } n \equiv 1 \ (mod\ 5) \text{ and } m > 2 \\ m\lfloor \frac{n}{5} \rfloor + \lfloor \frac{m}{2} \rfloor & \text{if } n \equiv 2 \ (mod\ 5), n \neq 2. \end{cases}$$

Proof. Graph $P_m \triangleright P_n$ is a connected graph with vertex set $V(P_m \triangleright P_n) = \{x_{i,j} : 1 \leq i \leq m, 1 \leq j \leq n\}$, and edge set $E(P_m \triangleright P_n) = \{x_{i,1}x_{(i+1),1} : 1 \leq i \leq m-1\} \cup \{x_{i,j}x_{i,j+1} : 1 \leq i \leq m, 1 \leq j \leq n-1\}$. The order and size are $|V(P_m \triangleright P_n)| = nm$ and $|E(P_m \triangleright P_n)| = nm - 1$, respectively.

We define the 2-resolving efficient dominating set $S \subset V(G)$ as follows: $S = \big\{\{x_{i,1} : i \equiv 1(mod\ 2), \text{ for } n = 2\} \cup \{x_{i,j} : 1 \leq i \leq m : j \equiv 3(mod\ 5), \text{ for } n \equiv 0, 3, 4(mod\ 5)\} \cup \{x_{i,j} : 1 \leq i \leq m, j \equiv 4(mod\ 5)\} \cup \{x_{i,1} : i \equiv 0(mod\ 3), \text{ for } n \equiv 1(mod\ 5) \cup \{x_{i,j} : 1 \leq i \leq m, j \equiv 0(mod\ 5)\} \cup \{x_{i,1} :$

$i \equiv 0(mod \quad 2)$, for $n \equiv 2(mod\, 5), n \neq 2\} \cup \{x_{1,j} : j \equiv 1(mod \quad 5), j = n\} \cup \{x_{2,j},\, j \equiv 0(mod \quad 5), j = n\}$, for $n \equiv 0, 4(mod\, 5), m = 2 \cup \{x_{1,j} : j \equiv 1(mod\ 5)\} \cup \{x_{2,j} : j \equiv 0(mod\ 5)\}$, for $n \equiv 1, 2(mod\, 5), m = 2 \cup \{x_{1,j} : j \equiv 1(mod\, 5)\} \cup \{x_{2,j} : j \equiv 0(mod\, 5) : j = n\}$, for $n \equiv 3(mod\, 5), m = 2\}$

We will show that S is 2-resolving efficient dominating set with minimum cardinality by the following steps:

First, we will show that S satisfies the properties of 2-efficient dominating set. For every $x_{i,j} \in S,\ d(x_{i,j}, x_{i,j}) \geq 2$, thus $|N(x_{i,j} \in V(P_m \triangleright P_n) - D) \cap D| \leq 2$ and every vertex in $x_{i,j} \in V(P_m \triangleright P_n) - D$ is dominated by one vertex in S. Therefore, we conclude that S is 2-efficient dominating set. Secondly, we will show that S also satisfies the properties of resolving set. Each element of S is in the subgraph $P_t(1 \leq t \leq m) \subset (P_m \triangleright P_n)$. Suppose P_r is a subgraph of P_t where $V(P_r) \cap S \neq \emptyset$. Since P_r is a path graph, each $u \in V(P_r)$ has a different representation respect to the elements $V(P_r) \cap S$. Since every vertex on the graph $P_m \triangleright P_n$ lies on $P_t(1 \leq t \leq m)$, it can be concluded that the representation of each vertex on the graph $P_m \triangleright P_n$ respect to the elements of S are all different. It implies that S satisfies the properties of resolving set.

Lastly, we will show that S is 2-resolving efficient dominating set with the minimum cardinality. Suppose we can set up S_1 such that $|S_1| \leq |S|$. Here are the conditions that might occur.

1. For $n = 2$, there exist vertices of $\{x_{i,1}\} \notin S_1 \rightarrow \exists\ x_{i,2}$ which are not dominated by S_1. Thus, S_1 does not show the efficient dominating set.
2. For $n \equiv 0, 3, 4(mod\, 5)$, there exist vertices of $\{x_{i,j}\} \notin S_1 \rightarrow \exists\ x_{i,j}$ which are not dominated by S_1. Thus, S_1 does not show the efficient dominating set.
3. For $n \equiv 1(mod\, 5)$,
   (a) There exist vertices of $x_{i,j} \notin S_1 \rightarrow \exists\ x_{i,j}$ which are not dominated by S_1. Thus, S_1 does not show the efficient dominating set.
   (b) There exist vertices of $x_{i,1} \notin S_1 \rightarrow \exists\ x_{i,1}$ which are not dominated by S_1. Thus, S_1 does not show the efficient dominating set.
4. For $n \equiv 2(mod\, 5), n \neq 2$,
   (a) There exist vertices of $x_{i,j} \notin S_1 \rightarrow \exists\ x_{i,j}$ which are not dominated by S_1. Thus, S_1 does not show the efficient dominating set.
   (b) There exist vertices of $x_{i,1} \notin S_1 \rightarrow \exists\ x_{i,2}$ which are not dominated by S_1. TThus, S_1 does not show the efficient dominating set.
5. For $m = 2$,
   (a) There exist vertices of $x_{1,j} \notin S_1 \rightarrow \exists\ x_{i,j}$ which are not dominated by S_1. Thus, S_1 does not show the efficient dominating set.
   (b) There exist vertices of $x_{2,j} \notin S_1 \rightarrow \exists\ x_{2,j}$ which are not dominated by S_1. Thus, S_1 does not show the efficient dominating set.

It implies that S_1 does not satisfy the properties of efficient dominating set. It concludes that $|S|$ must be the minimum cardinality of resolving efficient dominating set of $P_m \triangleright P_n$. □

4. Discussion

In this research we have analyzed the effectiveness of two ANN architectures across four ANN models in time series forecasting on soil moisture in the vertical farming and analyze it in term of iteration number, mean square error, regression, and their learning rate. We integrated REDS with ANN provides a bidirectional relationship between sensor-network design and data-driven forecasting. The compact architecture of vertical agricultural plants, taking into account the distance between plants in the study by Lim et al. (2025), further reinforces the feasibility of REDS, as the smaller size between sensor nodes reduces the spatial distance between growth points [59]. The results of our implementation of the REDS algorithm show that only 4 sensors are needed to

monitor 12 nodes without losing spatial resolution. This theory proves that REDS produces sensor placement that is mathematically optimal and agronomically feasible. While ANN captures temporal patterns, REDS ensures spatial sufficiency of the input data. This synergy reduces uncertainty arising from uneven sensor distribution. Compared to prior studies employing purely ANN based predictions, the REDS-ANN framework demonstrates superior robustness and interpretability. The methodological novelty lies in grounding a data intensive neural model upon an analytically optimal graph topology. Future expansions may integrate wireless network constraints and IoT automation for adaptive irrigation control. The results shows that the best architecture for phase 1 is obtained by Feedforwardnet ANN-567 model, with MSE of 0.3810. The best architecture for phase 2 is obtained by Patternnet ANN-567 model with an MSE of 1.1374×10^{-9}. The best architecture for phase 3 is obtained by Cascadeforwardnet ANN-567 model with an MSE of 1.1232×10^{-10}. Finally, the best architecture for phase 4 is obtained by model Cascadeforwardnet ANN-567 with an MSE of 1.0780×10^{-17}. Based on the computer simulation, we know that for best watering time of aloe vera plantation is in 3 days after the last watering, during Phase 4, at 16.45 - 17.15.

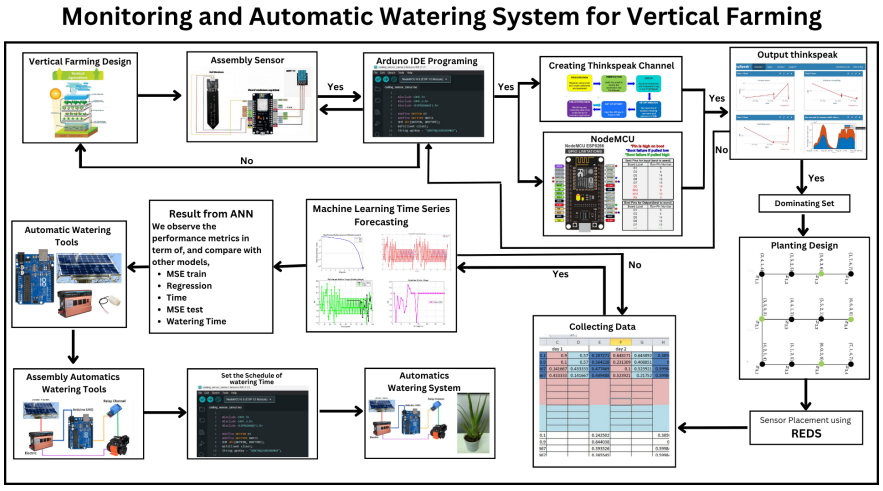

Figure 14. The Illustration of implementation of REDS for sensor placement in vertical farming

Furthermore, how to place those sensors to capture such data since the watering time is based on the soil moisture? The answer is not only focusing to the data analyzing generated from the sensors, but also how to design the most effective vertical farming design in such way we can place all sensors needed effectively and efficiently. The most effective way how to distribute the sensors or even LED, we can use the 2-Resolving Efficient Dominating Set. The mechanism of using this concept is the following: (1) Obtain a good vertical farming design based on the space availability, (2) Consider the associating graph structure on the vertical farming design, (3) Apply resolving efficient dominating set of distance one, two, three or k to place the sensors, (4) Grow specific plantation, for instance Aloe Vera, to place the sensors, (6) Collect the input data in regards with the number of sensors placement and do the ANN simulation by using machine learning. See Figure 14 for illustration of practical implementaion of REDS using ANN for time series forecasting for vertical farming.

5. Conclusion

The two ANN architectures across four ANN models in time series forecasting on soil moisture in the vertical farming have been utilized to determine the watering time on Aloe Vera plantation. The results show that the effectiveness of two architectures varies in each ANN models. Furthermore, the integrated REDS-ANN framework for soil-moisture forecasting in vertical farming, the sensors placement will be effectively to collect the temperature, air humidity and soil moisture data by considering the domination of resolvin set on their neighborhood plants.

By unifying sensor-network optimization and machine-learning prediction, the approach improves both spatial coverage and temporal accuracy. The combined methodology offers a scalable foundation for IoT based irrigation management. Future work will extend the framework to multi-crop environments and realtime deployment on embedded hardware.

6. Acknowledgement

The authors gratefully acknowledge the support from LPPM Universitas Jakarta, LP2M Universitas Jember, the PUI-PT Combinatorics and Graphs (CGANT), and the Directorate of Research and Community Service under the 2026 research grant. Their continuous assistance, encouragement, and collaborative efforts have been essential to the successful advancement of this study.

7. Appendix

Illustration 1
Given that an input data $\mathbf{x} = [x_1, x_2]$ and the target Y. The input and target data are given as follows

$$\mathbf{x} = \begin{bmatrix} 54 & 9.200.000 \\ 46 & 8.500.000 \\ 58 & 8.900.000 \\ 67 & 10.500.000 \end{bmatrix}, \text{ and } Y = \begin{bmatrix} 1.050.000 \\ 850.000 \\ 1.250.000 \\ 1.750.000 \end{bmatrix}$$

Show the calculation of two iterations of the first input data of the neural network architecture with no hidden layer.

Explication. Consider the neural network with no layer. To solve the problem above, we need to start by giving the initial weight and bias. Suppose $w^{0,1} = \begin{bmatrix} 1 \\ 0.8 \end{bmatrix}$, $\beta_{0,1} = 1$, and $\alpha = 0.5$. First, we need to normalize the input data by using $\mathbf{x}' = a + \frac{(x_i - x_{min})(b-a)}{x_{max} - x_{min}}$ within the range $[0.1 \ 0.9]$. Thus, we have the following new data scaling.

$$\mathbf{x} = \begin{bmatrix} 0.405 & 0.380 \\ 0.100 & 0.100 \\ 0.557 & 0.260 \\ 0.900 & 0.900 \end{bmatrix}, \text{ and } Y = \begin{bmatrix} 0.278 \\ 0.100 \\ 0.456 \\ 0.900 \end{bmatrix}$$

By using Perceptron Algorithm 1, the first iteration of the output unit of the input data $\mathbf{x} = [0.405 \ \ 0.380]$ can be calculated as follows:

$$\begin{aligned} \hat{Y}^1 &= \beta_{0,1} + w_1^{0,1} x_1 + w_2^{0,1} x_2 \\ &= 1 + 1 \cdot 0.405 + 0.8 \cdot 0.380 = 1.709 \end{aligned}$$

By using the linear activation function, we have $\hat{Y}^1_{linear} = \hat{Y}^1$. Since $\hat{Y}^1_{linear} \neq Y$, we need to update W and β by using Perceptron Algorithm 1,

$$\begin{aligned} w^{0,2} &= w^{0,1} + \alpha \cdot Y \cdot X \\ &= \begin{bmatrix} 1 \\ 0.8 \end{bmatrix} + 0.5 \cdot 0.278 \cdot \begin{bmatrix} 0.405 \\ 0.380 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.8 \end{bmatrix} - \begin{bmatrix} 0.056 \\ 0.052 \end{bmatrix} = \begin{bmatrix} 1.056 \\ 0.852 \end{bmatrix} \text{ and} \\ \beta_{0,2} &= \beta_{0,1} + \alpha \cdot Y = 1 + 0.5 \cdot 0.278 = 1.139 \end{aligned}$$

Thus, the first iteration of the linear activation function of the input data $\mathbf{x} = [0.100 \ \ 0.100]$ can be calculated as follows:

$$\begin{aligned} \hat{Y}^2_{linear} &= \beta_{0,2} + w_1^{0,2} x_1 + w_2^{0,2} x_2 \\ &= 1.139 + 1.056 \cdot 0.100 + 0.852 \cdot 0.100 = 1.435 \end{aligned}$$

By using the linear activation function, we have $\hat{Y}^2_{linear} = \hat{Y}^2$. Since $\hat{Y}^2_{linear} \neq Y$, again we need to update W and β before continuing to calculate the linear activation function for $\mathbf{x} = [0.557\ \ 0.260]$ and $\mathbf{x} = [0.900\ \ 0.900]$. Once we have calculated for all input data $\mathbf{x}$, we can continue to the second iteration by the same manner. □

Illustration 2
Given that an input data $\mathbf{x} = [x_1, x_2]$ and the target Y. The input and target data are given as above in Illustration 1. Show the calculation of two iterations of the first input data of the neural network architecture with one layer of two nodes.

Explication. Consider the neural network with one layer of two nodes. To solve the problem above, we need to start by giving the initial weight and bias, suppose $w^{1,1} = \begin{bmatrix}0.5\\0.2\end{bmatrix}$, $w^{0,1} = \begin{bmatrix}0.5 & 0.7\\1 & 0.4\end{bmatrix}$, $\beta_0^1 = \begin{bmatrix}0.5\\0.3\end{bmatrix}$, $\beta_{1,1} = 0.8$, and $\alpha = 0.7$. First, we need to normalize the input data by using $\mathbf{x}' = a + \frac{(x_i - x_{min})(b-a)}{x_{max} - x_{min}}$ within the range $[0.1\ 0.9]$. Thus, we have the following new data scaling.

$$\mathbf{x} = \begin{bmatrix}0.405 & 0.380\\0.100 & 0.100\\0.557 & 0.260\\0.900 & 0.900\end{bmatrix}, \text{ and } Y = \begin{bmatrix}0.278\\0.100\\0.456\\0.900\end{bmatrix}$$

Thus, the first iteration of the linear activation function can be calculated as follows:

$$\hat{Y}^1_{linear} = \beta_{1,1} + w_1^{1,1}h_1^1 + w_2^{1,1}h_2^1$$

where

$$\begin{bmatrix} h_1^1 \\ h_2^1 \end{bmatrix} = \begin{bmatrix} \beta_{0,1}^1 \\ \beta_{0,2}^1 \end{bmatrix} + \begin{bmatrix} w_{1,1}^{0,1} & w_{1,2}^{0,1} \\ w_{2,1}^{0,1} & w_{2,2}^{0,1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix}0.5\\0.3\end{bmatrix} + \begin{bmatrix}0.5 & 0.7\\1 & 0.4\end{bmatrix} \begin{bmatrix}0.405\\0.380\end{bmatrix} = \begin{bmatrix}0.969\\0.857\end{bmatrix}$$

$$\hat{Y}^1_{linear} = \beta_{1,1} + w_1^{1,1}h_1^1 + w_2^{1,1}h_2^1 = 0.8 + 0.5 \cdot 0.969 + 0.2 \cdot 0.857 = 1.456$$

By using the linear activation function, we have $\hat{Y}^1_{linear} = \hat{Y}^1$. Since $\hat{Y}^1_{linear} \neq Y$, we need to update W and β by using Perceptron Algorithm 1,

$$\begin{aligned} w^{0,2} &= w^{0,1} + \alpha \cdot Y \cdot X = w^{0,1} + \alpha \cdot h^1 \cdot X' \\ &= \begin{bmatrix}0.5 & 0.7\\1 & 0.4\end{bmatrix} + 0.7 \cdot \begin{bmatrix}0.969\\0.857\end{bmatrix} \cdot [0.405\ \ 0.380] = \begin{bmatrix}0.5 & 0.7\\1 & 0.4\end{bmatrix} + \begin{bmatrix}0.28 & 0.26\\0.24 & 0.23\end{bmatrix} = \begin{bmatrix}0.78 & 0.96\\1.24 & 0.63\end{bmatrix} \\ \beta_0^2 &= \beta_0^1 + \alpha \cdot h^1 = \begin{bmatrix}0.5\\0.3\end{bmatrix} + 0.7 \cdot \begin{bmatrix}0.969\\0.857\end{bmatrix} = \begin{bmatrix}1.178\\0.899\end{bmatrix} \end{aligned}$$

and

$$\begin{aligned} w^{1,2} &= w^{1,1} + \alpha \cdot Y \cdot X \\ &= \begin{bmatrix}0.5\\0.2\end{bmatrix} + 0.7 \cdot 0.278 \cdot \begin{bmatrix}0.405\\0.380\end{bmatrix} = \begin{bmatrix}0.5\\0.2\end{bmatrix} + \begin{bmatrix}0.079\\0.074\end{bmatrix} = \begin{bmatrix}0.579\\0.274\end{bmatrix} \\ \beta_{1,2} &= \beta_{1,1} + \alpha \cdot Y = 0.8 + 0.7 \cdot 0.278 = 0.995 \end{aligned}$$

Thus, the first iteration of the linear activation function of the input data $\mathbf{x} = [0.100\ \ 0.100]$ can be calculated as follows:

$$\hat{Y}^1_{linear} = \beta_{1,2} + w_1^{1,2}h_1^2 + w_2^{1,2}h_2^2$$

where

$$\begin{bmatrix} h_1^2 \\ h_2^2 \end{bmatrix} = \begin{bmatrix} \beta_{0,1}^2 \\ \beta_{0,2}^2 \end{bmatrix} + \begin{bmatrix} w_{1,1}^{0,2} & w_{1,2}^{0,2} \\ w_{2,1}^{0,2} & w_{2,2}^{0,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix}1.178\\0.899\end{bmatrix} + \begin{bmatrix}0.78 & 0.96\\1.24 & 0.63\end{bmatrix} \begin{bmatrix}0.100\\0.100\end{bmatrix} = \begin{bmatrix}1.352\\1.086\end{bmatrix}$$

Thus,

$$\hat{Y}^1_{linear} = \beta_{1,2} + w_1^{1,2}h_1^2 + w_2^{1,2}h_2^2 = 0.995 + 0.579 \cdot 1.352 + 0.274 \cdot 1.086 = 2.075$$

By using the linear activation function, we have $\hat{Y}^2_{linear} = \hat{Y}^2$. Since $\hat{Y}^2_{linear} \neq Y$, again we need to update W and β before continuing to calculate the linear activation function for $\mathbf{x} = [0.557 \;\; 0.260]$ and $\mathbf{x} = [0.900 \;\; 0.900]$. Once we have calculated for all input data $\mathbf{x}$, we can continue to the second iteration by the same manner. □

Illustration 3
Given that an input data $\mathbf{x} = [x_1, x_2]$ and the target Y. The input and target data are given as above in Illustration 1. Show the calculation of two iterations of the first input data of the neural network architecture with two hidden layers of respectively two and three nodes.

Explication. Consider the neural network architecture with two hidden layers of respectively two and three nodes. To solve the problem above, we need to start by giving the initial weight and bias, suppose $w^{2,1} = \begin{bmatrix} 0.3 \\ 0.4 \\ 0.1 \end{bmatrix}$, $w^{1,1} = \begin{bmatrix} 0.2 & 0.3 \\ 0.3 & 0.1 \\ 0.1 & 0.1 \end{bmatrix}$, $w^{0,1} = \begin{bmatrix} 0.4 & 0.6 \\ 0.7 & 0.8 \end{bmatrix}$, $\beta_{2,1} = 0.9$, $\beta_1^1 = \begin{bmatrix} 0.3 \\ 0.4 \\ 0.2 \end{bmatrix}$, $\beta_0^1 = \begin{bmatrix} 0.3 \\ 0.4 \end{bmatrix}$, and $\alpha = 0.7$. First, we need to normalize the input data by using $\mathbf{x}' = a + \frac{(x_i - x_{min})(b-a)}{x_{max} - x_{min}}$ within the range $[0.1 \;\; 0.9]$. Thus, we have the following new data scaling.

$$\mathbf{x} = \begin{bmatrix} 0.405 & 0.380 \\ 0.100 & 0.100 \\ 0.557 & 0.260 \\ 0.900 & 0.900 \end{bmatrix}, \text{ and } Y = \begin{bmatrix} 0.278 \\ 0.100 \\ 0.456 \\ 0.900 \end{bmatrix}$$

By this architecture, we can derive the first iteration of the linear activation function as $\hat{Y}^1_{linear} = \beta_{2,1} + w_1^{2,1}h_1^2 + w_2^{2,1}h_2^2 + w_3^{2,1}h_3^2 = \beta_2 + \sum_{i=1}^3 w_i^2 h_i^2$, where

$$\begin{bmatrix} h_1^1 \\ h_2^1 \end{bmatrix} = \begin{bmatrix} \beta_{0,1}^1 \\ \beta_{0,2}^1 \end{bmatrix} + \begin{bmatrix} w_{1,1}^{0,1} & w_{1,2}^{0,1} \\ w_{2,1}^{0,1} & w_{2,2}^{0,1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.4 \end{bmatrix} + \begin{bmatrix} 0.4 & 0.6 \\ 0.7 & 0.8 \end{bmatrix} \begin{bmatrix} 0.405 \\ 0.380 \end{bmatrix} = \begin{bmatrix} 0.690 \\ 0.986 \end{bmatrix}$$

and

$$\begin{bmatrix} h_1^2 \\ h_2^2 \\ h_3^2 \end{bmatrix} = \begin{bmatrix} \beta_{1,1}^1 \\ \beta_{1,2}^1 \\ \beta_{1,3}^1 \end{bmatrix} + \begin{bmatrix} w_{1,1}^{1,1} & w_{1,2}^{1,1}1 \\ w_{2,1}^{1,1} & w_{2,2}^{1,1} \\ w_{3,1}^{1,1} & w_{3,2}^{1,1} \end{bmatrix} \begin{bmatrix} h_1^1 \\ h_2^1 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.4 \\ 0.2 \end{bmatrix} + \begin{bmatrix} 0.2 & 0.3 \\ 0.3 & 0.1 \\ 0.1 & 0.1 \end{bmatrix} \begin{bmatrix} 0.690 \\ 0.986 \end{bmatrix} = \begin{bmatrix} 0.734 \\ 0.706 \\ 0.368 \end{bmatrix}$$

Thus,

$$\hat{Y}^1_{linear} = \beta_{2,1} + w_1^{2,1}h_1^2 + w_2^{2,1}h_2^2 + w_3^{2,1}h_3^2 = 0.9 + 0.3 \cdot 0.734 + 0.4 \cdot 0.706 + 0.1 \cdot 0.368 = 1.439$$

By using the linear activation function, we have $\hat{Y}^1_{linear} = \hat{Y}^1$. Since $\hat{Y}^1_{linear} \neq Y$, we need to update W and β by using Perceptron Algorithm 1,

$$\begin{aligned} w^{0,2} &= w^{0,1} + \alpha \cdot Y \cdot X = w^{0,1} + \alpha \cdot h^1 \cdot X' \\ &= \begin{bmatrix} 0.4 & 0.6 \\ 0.7 & 0.8 \end{bmatrix} + 0.7 \cdot \begin{bmatrix} 0.690 \\ 0.986 \end{bmatrix} \cdot [0.405 \;\; 0.380] = \begin{bmatrix} 0.4 & 0.6 \\ 0.7 & 0.8 \end{bmatrix} + \begin{bmatrix} 0.19 & 0.18 \\ 0.28 & 0.26 \end{bmatrix} = \begin{bmatrix} 0.59 & 0.78 \\ 0.98 & 1.06 \end{bmatrix} \\ \beta_0^2 &= \beta_0^1 + \alpha \cdot h^1 = \begin{bmatrix} 0.3 \\ 0.4 \end{bmatrix} + 0.7 \cdot \begin{bmatrix} 0.690 \\ 0.986 \end{bmatrix} = \begin{bmatrix} 0.783 \\ 1.090 \end{bmatrix} \end{aligned}$$

and

$$\begin{aligned} w^{1,2} &= w^{1,1} + \alpha \cdot Y \cdot X = w^{1,1} + \alpha \cdot h^2 \cdot X' \\ &= \begin{bmatrix} 0.2 & 0.3 \\ 0.3 & 0.1 \\ 0.1 & 0.1 \end{bmatrix} + 0.7 \cdot \begin{bmatrix} 0.734 \\ 0.706 \\ 0.368 \end{bmatrix} \cdot [0.405 \;\; 0.380] = \begin{bmatrix} 0.2 & 0.3 \\ 0.3 & 0.1 \\ 0.1 & 0.1 \end{bmatrix} + \begin{bmatrix} 0.21 & 0.19 \\ 0.20 & 0.18 \\ 0.10 & 0.10 \end{bmatrix} = \begin{bmatrix} 0.41 & 0.49 \\ 0.50 & 0.28 \\ 0.20 & 0.20 \end{bmatrix} \\ \beta_1^2 &= \beta_1^1 + \alpha \cdot Y = \beta_{1,1} + \alpha \cdot h^2 = \begin{bmatrix} 0.3 \\ 0.4 \\ 0.2 \end{bmatrix} + 0.7 \cdot \begin{bmatrix} 0.734 \\ 0.706 \\ 0.368 \end{bmatrix} = \begin{bmatrix} 0.813 \\ 0.894 \\ 0.458 \end{bmatrix} \end{aligned}$$

and

$$\begin{aligned} w^{2,2} &= w^{2,1} + \alpha \cdot Y \cdot X = w^{1,1} + \alpha \cdot Y \cdot h^2 \\ &= \begin{bmatrix} 0.3 \\ 0.4 \\ 0.1 \end{bmatrix} + 0.7 \cdot 0.278 \cdot \begin{bmatrix} 0.734 \\ 0.706 \\ 0.368 \end{bmatrix} = \begin{bmatrix} 0.843 \\ 0.537 \\ 0.172 \end{bmatrix} \\ \beta_{2,2} &= \beta_{2,1} + \alpha \cdot Y = 0.9 + 0.7 \cdot 0.278 = 1.095 \end{aligned}$$

Thus, the first iteration of the linear activation function of the input data $\mathbf{x} = [0.100 \;\; 0.100]$ can be calculated as follows:

$$\hat{Y}^1_{linear} = \beta_{2,2} + w_1^{2,2} h_1^3 + w_2^{2,2} h_2^3 + w_3^{2,2} h_3^3$$

Where

$$\begin{bmatrix} h_1^2 \\ h_2^2 \end{bmatrix} = \begin{bmatrix} \beta_{0,1}^2 \\ \beta_{0,2}^2 \end{bmatrix} + \begin{bmatrix} w_{1,1}^{0,2} & w_{1,2}^{0,2} \\ w_{2,1}^{0,2} & w_{2,2}^{0,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.738 \\ 1.090 \end{bmatrix} + \begin{bmatrix} 0.59 & 0.78 \\ 0.98 & 1.06 \end{bmatrix} \begin{bmatrix} 0.100 \\ 0.100 \end{bmatrix} = \begin{bmatrix} 0.875 \\ 1.294 \end{bmatrix}$$

and

$$\begin{bmatrix} h_1^3 \\ h_2^3 \\ h_3^3 \end{bmatrix} = \begin{bmatrix} \beta_{1,1}^2 \\ \beta_{1,2}^2 \\ \beta_{1,3}^2 \end{bmatrix} + \begin{bmatrix} w_{1,1}^{1,2} & w_{1,2}^{1,2} \\ w_{2,1}^{1,2} & w_{2,2}^{1,2} \\ w_{3,1}^{1,2} & w_{3,2}^{1,2} \end{bmatrix} \begin{bmatrix} h_1^2 \\ h_2^2 \end{bmatrix} = \begin{bmatrix} 0.813 \\ 0.894 \\ 0.458 \end{bmatrix} + \begin{bmatrix} 0.41 & 0.49 \\ 0.50 & 0.28 \\ 0.20 & 0.20 \end{bmatrix} \begin{bmatrix} 0.875 \\ 1.294 \end{bmatrix} = \begin{bmatrix} 1.8058 \\ 1.6938 \\ 0.8918 \end{bmatrix}$$

Thus,

$$\begin{aligned} \hat{Y}^1_{linear} &= \beta_{2,2} + w_1^{2,2} h_1^3 + w_2^{2,2} h_2^3 + w_3^{2,2} h_3^3 \\ &= 1.095 + 0.843 \times 1.8058 + 0.537 \times 1.6938 + 0.172 \times 0.8918 = 3.682 \end{aligned}$$

REFERENCES

1. Al-Kodmany, K. (2018). The vertical farm: A review of developments and implications for the vertical city. *Buildings, 8*(2), 24. https://doi.org/10.3390/buildings8020024
2. Abdullah, N.-O. (2016). Vertical-Horizontal Regulated Soilless Farming via Advanced Hydroponics for Domestic Food Production in Doha, Qatar. Research Ideas and Outcomes, 2. https://doi.org/10.3897/rio.2.e8134
3. Kernbach, S. (2024). Biofeedback-Based Closed-Loop Phytoactuation in Vertical Farming and Controlled-Environment Agriculture. Biomimetics, 9(10), 640. https://doi.org/10.3390/biomimetics9100640
4. Heuvelink, E., Hemming, S., & Marcelis, L. F. M. (2024). Some recent developments in controlled-environment agriculture: on plant physiology, sustainability, and autonomous control. The Journal of Horticultural Science and Biotechnology, 100(5), 604–614. https://doi.org/10.1080/14620316.2024.2440592
5. Manuel, J., Rosario Martinez-Blanco, Ma. del, & Rene, H. (2011). Evolutionary Artificial Neural Networks in Neutron Spectrometry. Artificial Neural Networks - Application. https://doi.org/10.5772/15131
6. Jeyakrishnan, S., Vijayakumar, S., Naga Swapna Sri, M., & Anusha, P. (2024). An integration of RSM and ANN modelling approach for prediction of FSW joint properties in AA7178/AA5456 alloys. Canadian Metallurgical Quarterly, 64(1), 43–60. https://doi.org/10.1080/00084433.2024.2310344

7. Kardum, M. (2020). Rudolf Carnap–The Grandfather of Artificial Neural Networks: The Influence of Carnap's Philosophy on Walter Pitts. Guide to Deep Learning Basics, 55–66. https://doi.org/10.1007/978-3-030-37591-1-6
8. Dassanayake, J. K., Kulathunga, R., Baduge, G. A. A., & Vaezi, M. (2025). Unsupervised Learning-Based ISAC Waveforms. IEEE Wireless Communications Letters, 14(9), 2663–2667. https://doi.org/10.1109/lwc.2025.3560907
9. Kaur, G., Sharma, S., & Singh, P. (2024). Development of Neural Networks and Performance Appraisal of Supervised Learning Models for Predicting Organic Carbon in Soils Under Different Cropping Systems. Journal of Soil Science and Plant Nutrition, 25(1), 1029–1046. https://doi.org/10.1007/s42729-024-02182-1
10. Davies, S., Gait, A., Rowley, A., & Di Nuovo, A. (2025). Supervised learning of spatial features with STDP and homeostasis using Spiking Neural Networks on SpiNNaker. Neurocomputing, 611, 128650. https://doi.org/10.1016/j.neucom.2024.128650
11. Afzal, A., Buradi, A., Islam, Md. T., Asif, M., Fayaz, H., Park, S. G., Munimathan, A., & Bordas, S. P. (2025). Comparative analysis of ensemble, supervised, and deep learning regression algorithms for parametric modelling of solid-liquid fluidization. Journal of the Taiwan Institute of Chemical Engineers, 171, 106053. https://doi.org/10.1016/j.jtice.2025.106053
12. Sarker, I. H., & Kayes, A. S. M. (2020). ABC-RuleMiner: User behavioral rule-based machine learning method for context-aware intelligent services. *Journal of Network and Computer Applications, 168*, 102762. https://doi.org/10.1016/j.jnca.2020.102762
13. Hamzi, B., Hutter, M., & Owhadi, H. (2025). Bridging Algorithmic Information Theory and Machine Learning: Clustering, density estimation, Kolmogorov complexity-based kernels, and kernel learning in unsupervised learning. Physica D: Nonlinear Phenomena, 476, 134669. https://doi.org/10.1016/j.physd.2025.134669
14. Ördek, B., Coatanea, E., & Borgianni, Y. (2025). An auto hierarchical clustering algorithm to distinguish geometries suitable for additive and traditional manufacturing technologies: Comparing humans and unsupervised learning. Results in Engineering, 25, 104418. https://doi.org/10.1016/j.rineng.2025.104418
15. Bresso, E., Lacomblez, C., Duarte, K., Monzo, L., Baudry, G., Tromp, J., Sharma, A., & Girerd, N. (2025). Unsupervised machine learning for cardiovascular disease: A framework for future studies. European Journal of Heart Failure. Portico. https://doi.org/10.1002/ejhf.70076
16. Perumal, S., Sujatha, P. K., S., K., & Krishnan, M. (2025). Clusters in chaos: A deep unsupervised learning paradigm for network anomaly detection. Journal of Network and Computer Applications, 235, 104083. https://doi.org/10.1016/j.jnca.2024.104083
17. Wangguway, Y., Slamin, Dafik, Wardani, D. A. R., & Alfarisi, R. (2020). On resolving domination number of special family of graphs. *Journal of Physics: Conference Series, 1465*(1), 012015. IOP Publishing. https://doi.org/10.1088/1742-6596/1465/1/012015
18. Adawiyah, R., Agustin, I. H., Dafik, Slamin, & Albirri, E. R. (2018). Related wheel graphs and its locating edge domination number. *Journal of Physics: Conference Series, 1022*(1), 012007. IOP Publishing. https://doi.org/10.1088/1742-6596/1022/1/012007
19. Elshorbagy, A., & Parasuraman, K. (2008). On the relevance of using artificial neural networks for estimating soil moisture content. Journal of Hydrology, 362(1–2), 1–18. https://doi.org/10.1016/j.jhydrol.2008.08.012
20. Sanuade, O. A., Adetokunbo, P., Oladunjoye, M. A., & Olaojo, A. A. (2018). Predicting moisture content of soil from thermal properties using artificial neural network. Arabian Journal of Geosciences, 11(18). https://doi.org/10.1007/s12517-018-3917-4
21. Radha, M. H., & Laxmipriya, N. P. (2015). Evaluation of biological properties and clinical effectiveness of *Aloe vera*: A systematic review. *Journal of Traditional and Complementary Medicine, 5*(1), 21–26. https://doi.org/10.1016/j.jtcme.2014.10.006
22. Eshun, K., & He, Q. (2004). Aloe Vera: A Valuable Ingredient for the Food, Pharmaceutical and Cosmetic Industries—A Review. Critical Reviews in Food Science and Nutrition, 44(2), 91–96. https://doi.org/10.1080/10408690490424694
23. Ziemlewska, A., Zagórska-Dziok, M., Nowak, A., Muzykiewicz-Szymańska, A., Wójciak, M., Sowa, I., Szczepanek, D., & Nizioł-Łukaszewska, Z. (2025). Enhancing the Cosmetic Potential of Aloe Vera Gel by Kombucha-Mediated Fermentation: Phytochemical Analysis and Evaluation of Antioxidant, Anti-Aging and Moisturizing Properties. Molecules, 30(15), 3192. https://doi.org/10.3390/molecules30153192
24. Dal'Belo, S. E., Rigo Gaspar, L., & Berardo Gonçalves Maia Campos, P. M. (2006). Moisturizing effect of cosmetic formulations containing Aloe vera extract in different concentrations assessed by skin bioengineering techniques. Skin Research and Technology, 12(4), 241–246. Portico. https://doi.org/10.1111/j.0909-752x.2006.00155.x
25. Singh, A., Verma, K., Kumar, D., Nilofer, Lothe, N. B., Kumar, A., Chaudhary, A., Kaur, P., Singh, K. P., Singh, A. K., Kumar, R., A. T. M., & Singh, S. (2021). Optimized irrigation regime and planting technique improve yields and economics in *Aloe vera*. *Industrial Crops and Products, 167*, 113539. https://doi.org/10.1016/j.indcrop.2021.113539
26. Gentilini, R., Bozzini, S., Munarin, F., Petrini, P., Visai, L., & Tanzi, M. C. (2013). Pectins from Aloe Vera: Extraction and production of gels for regenerative medicine. Journal of Applied Polymer Science, 131(2). Portico. https://doi.org/10.1002/app.39760
27. Shakib, Z., Shahraki, N., Razavi, B. M., & Hosseinzadeh, H. (2019). Aloe vera as an herbal medicine in the treatment of metabolic syndrome: A review. Phytotherapy Research, 33(10), 2649–2660. Portico. https://doi.org/10.1002/ptr.6465
28. Munawwarah, M., Dafik, Kristiana, A. I., Kurniawati, E. Y., & Nisviasari, R. (2023). On Resolving Efficient Dominating Set of Cycle and Comb Product Graph. Proceedings of the 6th International Conference on Combinatorics, Graph Theory, and Network Topology (ICCGANT 2022), 3–16. https://doi.org/10.2991/978-94-6463-138-8(2)
29. Patil, A. D., Ghasemi, A., & de Meer, H. (2025). Optimal Redundant Sensor Placement for Protection Blinding in Active Distribution Grids. IEEE Transactions on Network and Service Management, 22(2), 1409–1419. https://doi.org/10.1109/tnsm.2024.3497296
30. Al Mamun, Md. R., Ahmed, A. K., Upoma, S. M., Haque, M., & Ashik-E-Rabbani, M. (2025). IoT-enabled solar-powered smart irrigation for precision agriculture. Smart Agricultural Technology, 10, 100773. https://doi.org/10.1016/j.atech.2025.100773
31. Telaumbanua, M., Noval, F. A., Erika, Y., Haryanto, A., Lanya, B., Wisnu, F. K., Ansari, A., & Indriyawati, A. (2024). Design of temperature-soil moisture control and monitoring system for chili cultivation in greenhouse. IOP Conference Series: Earth and Environmental Science, 1386(1), 012029. https://doi.org/10.1088/1755-1315/1386/1/012029
32. Júnior, J. L. dos S., Silva, L. H. P., Andrade, I. R., Rosa, R., & Abegao, L. (2025). Real-World Deployment of an Affordable Sub-Soil Temperature and Relative Humidity Sensing System for Precision Agriculture. Measurement Science and Technology. https://doi.org/10.1088/1361-6501/ae1c63
33. Karthick, S., Kumar, S. L. M., Vendan, M. V., Gowtham, S., & Sivasakthivel, M. (2025). Designing and Implementing a Cutting Edge Smart Irrigation System to Optimize Water Resources and Improve Crop Yield in Precision Farming. 2025 8th International Conference on Circuit, Power &Amp; Computing Technologies (ICCPCT), 400–404.

https://doi.org/10.1109/iccpct65132.2025.11176602
34. Jaramillo, I. E., Cocco, C., Kang, J. J., Cheng, C.-L., & Pereira, E. (2025). Turning Waste into Fertilizer: Aloe vera Leaf Shavings Improve Plant Growth and Support Soil Fertility in Organic Systems. Soil Systems, 9(4), 113. https://doi.org/10.3390/soilsystems9040113
35. Silva, H., Sagardia, S., Seguel, O., Torres, C., Tapia, C., Franck, N., & Cardemil, L. (2010). Effect of water availability on growth and water use efficiency for biomass and gel production in Aloe Vera (Aloe barbadensis M.). Industrial Crops and Products, 31(1), 20–27. https://doi.org/10.1016/j.indcrop.2009.08.001
36. Kim, Y.-S., Kim, M. K., Fu, N., Liu, J., Wang, J., & Srebric, J. (2025). Investigating the impact of data normalization methods on predicting electricity consumption in a building using different artificial neural network models. Sustainable Cities and Society, 118, 105570. https://doi.org/10.1016/j.scs.2024.105570
37. Jin, J., Li, M., & Jin, L. (2015). Data Normalization to Accelerate Training for Linear Neural Net to Predict Tropical Cyclone Tracks. Mathematical Problems in Engineering, 2015, 1–8. https://doi.org/10.1155/2015/931629
38. Ding, B., Qian, H., & Zhou, J. (2018). Activation functions and their characteristics in deep neural networks. 2018 Chinese Control And Decision Conference (CCDC), 1836–1841. https://doi.org/10.1109/ccdc.2018.8407425
39. Rubio, J. de J. (2021). Stability Analysis of the Modified Levenberg–Marquardt Algorithm for the Artificial Neural Network Training. IEEE Transactions on Neural Networks and Learning Systems, 32(8), 3510–3524. https://doi.org/10.1109/tnnls.2020.3015200
40. Ait Gougam, L., Tribeche, M., & Mekideche-Chafa, F. (2008). A systematic investigation of a neural network for function approximation. Neural Networks, 21(9), 1311–1317. https://doi.org/10.1016/j.neunet.2008.06.015
41. Lazcano, A., Jaramillo-Morán, M. A., & Sandubete, J. E. (2024). Back to Basics: The Power of the Multilayer Perceptron in Financial Time Series Forecasting. Mathematics, 12(12), 1920. https://doi.org/10.3390/math12121920
42. Ostovar, A., Davari, D. D., & Dzikuć, M. (2025). Determinants of Design with Multilayer Perceptron Neural Networks: A Comparison with Logistic Regression. Sustainability, 17(6), 2611. https://doi.org/10.3390/su17062611
43. Lyu, Z., Ororbia, A., & Desell, T. (2023). Online evolutionary neural architecture search for multivariate non-stationary time series forecasting. Applied Soft Computing, 145, 110522. https://doi.org/10.1016/j.asoc.2023.110522
44. Shomope, I., Tawalbeh, M., Al-Othman, A., & Almomani, F. (2025). Predicting biohydrogen production from dark fermentation of organic waste biomass using multilayer perceptron artificial neural network (MLP–ANN). Computers & Chemical Engineering, 192, 108900.
45. Rachmatullah, M. I. C., Santoso, J., & Surendro, K. (2021). Determining the number of hidden layer and hidden neuron of neural network for wind speed prediction. PeerJ Computer Science, 7, e724. Portico. https://doi.org/10.7717/peerj-cs.724
46. Liang, Z., & Sun, Y. (2024, February). Evolutionary neural architecture search for multivariate time series forecasting. In Asian Conference on Machine Learning (pp. 771-786). PMLR.
47. Lyu, Z., & Desell, T. (2022). ONE-NAS. Proceedings of the Genetic and Evolutionary Computation Conference Companion, 659–662. https://doi.org/10.1145/3520304.3528962
48. Jie, R., & Gao, J. (2021). Differentiable Neural Architecture Search for High-Dimensional Time Series Forecasting. IEEE Access, 9, 20922–20932. https://doi.org/10.1109/access.2021.3055555
49. Qiao, M., Liang, Y., Tavares, A., & Shi, X. (2023). Multilayer Perceptron Network Optimization for Chaotic Time Series Modeling. Entropy, 25(7), 973. https://doi.org/10.3390/e25070973
50. Rudenko, O., Bezsonov, O., & Romanyk, O. (2019). Neural network time series prediction based on multilayer perceptron. Development Management, 17(1), 23–34. https://doi.org/10.21511/dm.5(1).2019.03
51. Alfarisi, R., Dafik, & Kristiana, A. I. (2019). Resolving domination number of graphs. *Discrete Mathematics, Algorithms and Applications, 11*(6), 1950071. https://doi.org/10.1142/S1793830919500712
52. Kurniawati, S., Dafik, & Slamin. (2020). On resolving domination number of friendship graph and its operation. *Journal of Physics: Conference Series, 1465*(1), 012023. IOP Publishing. https://doi.org/10.1088/1742-6596/1465/1/012023
53. Deng, Y. P., Sun, Y. Q., Liu, Q., & Wang, H. C. (2017). Efficient dominating sets in circulant graphs. *Discrete Mathematics, 340*(6), 1503–1507. https://doi.org/10.1016/j.disc.2017.02.008
54. Kusumawardani, A., Prasetyo, E., & Rahayu, P. (2022). On resolving efficient domination number of path and comb product of special graph.*Journal of Physics: Conference Series, 2165*(1), 012014. IOP Publishing. https://doi.org/10.1088/1742-6596/2165/1/012014
55. Gembong, A. W., Slamin, Dafik, & Agustin, I. H. (2017). Bound of distance domination number of graph and edge comb product graph. *Journal of Physics: Conference Series, 855*(1), 012014. IOP Publishing. https://doi.org/10.1088/1742-6596/855/1/012014
56. Hayyu, A. N., Dafik, Tirta, I. M., Adawiyah, R., & Prihandini, R. M. (2020). Resolving domination number of helm graph and its operation. *Journal of Physics: Conference Series, 1465*(1), 012022. IOP Publishing. https://doi.org/10.1088/1742-6596/1465/1/012022
57. Mee, N., Subramanian, A., & Swaminathan, V. (2014). Strong efficient domination in graphs.*International Journal of Innovative Science, Engineering, and Technology, 1*(4), 23–28.
58. Zeidler, C., & Schubert, D. (2014). From bioregenerative life support systems for space to vertical farming on Earth — The 100%. *Acta Astronautica, 104*(1), 271–279. https://doi.org/10.1016/j.actaastro.2014.07.009
59. Lim, Y., Seo, M., Lee, J., Hong, S., An, J., Jeong, H., Choi, H., Hong, W., Lee, C., Park, S. J., & Kwon, C. (2025). Optimizing plant size for vertical farming by editing stem length regulators. Plant Biotechnology Journal, 23(8), 3041–3053. Portico. https://doi.org/10.1111/pbi.70129
60. Adla, S., Rai, N. K., Karumanchi, S. H., Tripathi, S., Disse, M., & Pande, S. (2020). Laboratory Calibration and Performance Evaluation of Low-Cost Capacitive and Very Low-Cost Resistive Soil Moisture Sensors. Sensors, 20(2), 363. https://doi.org/10.3390/s20020363
61. Prihandini, R. M., Rahmadani, M., & Dafik, D. (2024). Analysis of Resolving Efficient Dominating Set and Its Application Scheme in Solving ETLE Problems. BAREKENG: Jurnal Ilmu Matematika Dan Terapan, 18(3), 1615-1628. https://doi.org/10.30598/barekengvol18iss3pp1615-1628
62. Petroșanu, D. and Pîrjan, A. (2020). Electricity Consumption Forecasting Based on A Bidirectional Long-Short-Term Memory Artificial Neural Network. Sustainability, 13(1), 104. https://doi.org/10.3390/su13010104

63. Kaushik, S., Choudhury, A., Sheron, P. K., Dasgupta, N., Natarajan, S., Pickett, L. A., … & Dutt, V. (2020). AI in Healthcare: Time-Series Forecasting Using Statistical, Neural, and Ensemble Architectures. Frontiers in Big Data, 3. https://doi.org/10.3389/fdata.2020.00004
64. Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning. https://doi.org/10.48550/arxiv.1506.00019
65. Gündoğdu, S. and Elbir, T. (2021). Application ofFeed Forward and Cascade Forward Neural Network Models for Prediction of Hourly Ambient Air Temperature Based on Merra-2 Reanalysis Data in A Coastal Area of Turkey. Meteorology and Atmospheric Physics, 133(5), 1481-1493. https://doi.org/10.1007/s00703-021-00821-1
66. Nandagopal, M. S. G. and Selvaraju, N. (2016). Prediction of Liquid–Liquid Flow Patterns in A y-Junction Circular Microchannel Using Advanced Neural Network Techniques. Industrial & Engineering Chemistry Research, 55(43), 11346-11362. https://doi.org/10.1021/acs.iecr.6b02438
67. Ellahi, M., Usman, M. R., Arif, W., Usman, H. F., Khan, W. A., Satrya, G. B., … & Shabbir, N. (2022). Forecasting of Wind Speed and Power Through FFNN and CFNN Using HPSOBA and MHPSO-BAACS Techniques. Electronics, 11(24), 4193. https://doi.org/10.3390/electronics11244193
68. Gokmen, T. and Haensch, W. (2020). Algorithm for Training Neural Networks on Resistive Device Arrays. Frontiers in Neuroscience, 14. https://doi.org/10.3389/fnins.2020.00103
69. Jin, M., & Mullens, T. (2014). A Study of the Relations between Soil Moisture, Soil Temperatures and Surface Temperatures Using ARM Observations and Offline CLM4 Simulations. Climate, 2(4), 279–295. https://doi.org/10.3390/cli2040279
70. Leuschner, C., & Lendzion, J. (2009). Air humidity, soil moisture and soil chemistry as determinants of the herb layer composition in European beech forests. Journal of Vegetation Science, 20(2), 288–298. Portico. https://doi.org/10.1111/j.1654-1103.2009.05641.x
71. Pengtao, W. (2020). Based on Adam Optimization Algorithm: Neural Network Model for Auto Steel Performance prediction. Journal of Physics: Conference Series, 1653(1), 012012. https://doi.org/10.1088/1742-6596/1653/1/012012