The Analysis of Airport Passengers Flow by Using Spatial Temporal Graph Neural Networks and Resolving Efficient Dominating Set

A. Muklisin[1], I. M. Tirta[3], Dafik[1,2(✉)], R. I. Baihaki[2], and A. I. Kristiana[1,2]

[1] Departement of Mathematics Education Postgraduate, University of Jember, Jember, Indonesia
210220101010@mail.unej.ac.id, {d.dafik,arika.fkip}@unej.ac.id
[2] PUI-PT Combinatorics and Graph, CGANT, University of Jember, Jember, Indonesia
[3] Departement of Mathematics, University of Jember, Jember, Indonesia
itirta.fmipa@unej.ac.id

Abstract. The advanced development of airport infrastructure is intended to improve airport services for air flight costumers. Thus, it is compulsory to maintain the sustainability system for service establishment. The purpose of this research is to apply the concept of Spatial Temporal Graph Neural Network (STGNN) integrated with Resolving Efficient Dominating Set (REDS) to analyze density of airport passengers flow. This research method includes the analytical and the experimental techniques. The input data involves the number of in-coming and out-coming airline, number of schedule, number of passengers and the weather. The results shows that the use of the Spatial Temporal Graph Neural Network and Resolving Efficient Dominating Set are effective tools for Airport Passengers Flow analysis, with cascadeforwardnet ANN (665) get MSE TEST $1.0328 \times {}^{-09}$.

Keywords: Spatial Temporal Neural Networks · Resolving Efficient Dominating Set · Airport Passengers Flow

1 Introduction

The domestic aviation network is a massive system consisting of components of passengers, goods and flight routes [1]. The world aviation system is a relatively tight structured level consisting of a small group of dominant airports that function as airports of origin or airports of destination [2]. The global air transport network is responsible for carrying millions of domestic and international passengers every year. Not surprisingly, the relationship between airports varies greatly, due to various geographical, economic, political, and historical settings [3]. In addition, given the dynamic nature of the many influences acting on the air transport system, the relationship between airports and the structure of the global air network as a whole is also constantly changing and influencing each other [4].

Experimental learning and analytic studies are always followed by the use of computational techniques, and it takes a lot of time to arrive at accurate results from the problem

I. H. Agustin (Ed.): ICONNSMAL 2022, AISR 177, pp. 3–20, 2023.
https://doi.org/10.2991/978-94-6463-174-6_2

to be solved [5]. The use of Artificial Neural Networks (ANN) is able to save time and also provide key information patterns in multidimensional information domains and therefore, ANN techniques are becoming increasingly popular in the fields of Science and Engineering [6], especially in the field of Mechanical Engineering applications in recent years [7].

The application of Artificial Neural Networks (ANN) is an effective way to analyze the flow of passenger density at airports as a preventive measure. From the perspective of ANN input data, passenger density at airports is likened to a non-Euclidean data source [8]. To solve this type of data source, the use of graph neural networks or often referred to as (GNN) is considered a breakthrough in machine learning. With a graph in this case the structure $G = (\mathrm{V(G)}, \mathrm{E(G)}$ where $V(G)$ is a finite non-empty set of elements called vertices, and $E(G)$ is the set (possibly empty) of unordered pairs (u, v) of vertices u, v in $V(G)$ are called edges The number of vertices of a graph G is the order of$G$, usually denoted by $|V(G)|$ The number of edges is the size of G, often denoted $|E(G)|$ a graph G has order $p = |V(G)|$ and size $q = |E(G)|$ is something called a graph (p, q) suppose u, v in $V(G)$, a node u is said to be a neighbor of v if there is an edge e between u and$v$, i.e. $e = uv$ The node v is called the neighbor of u The set of all neighbors of u is called the neighbor of u and is denoted by $N(u)$ We also say that u and v side by side e [9].The adjacency matrix of a graph G and vertex set $V(G) = \{v_1, v_2, v_3 \ldots v_n\}$ is the $n \times n$ is the $n \times n$ matrix $\mathrm{A} = [a_{ij}]$ where

$$\mathrm{D} = \begin{cases} 1 \; if \;\; v_i v_j \in E(G) \\ 0 \; otherwise \ldots \end{cases}$$

The GNN technique adopts the Convolutional Neural Networks (CNN) framework [10] many methods of applying CNN are useful for studying discriminatory features, which have played an important role in increasing the accuracy of the classification of a problem [11]. Some uses of CNN introduce hidden convolution and layer pooling to identify spatially localized features via a set of receptive fields in the kernel form [12] (Fig. 1).

In this image, what happens is that the middle node of the middle pixel gathers information from its neighbors and it self, which aims to generate a new value. To deal with this problem we need to use a Graph Neural Network (GNN), this is because CNN has a graph size that is always changing and a relatively more complicated topology, this means there is no spatial locality. In addition, CNN has an installation of vertices that

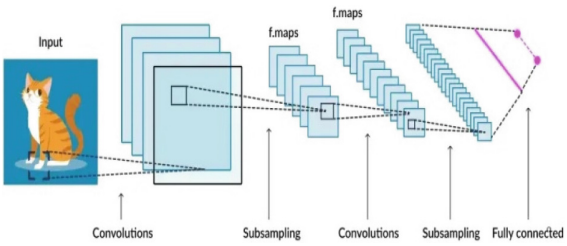

Fig. 1. The illustration of CNN on image. https://lamiae-hana.medium.com

are not fixed. For example, if we first label them F, G, H, I, J and the second time we label them K, L, M, N, O, then what will happen is that the graph will not change [13].

GNN is a method in advanced learning classes designed to make inferences on data described by graphs. GNNs are neural networks that can be applied to graphs and provide an easy way to perform forecasting or prediction tasks. In graph theory, we apply the concept of insertion of vertices. It means providing node feature cues to storage space. Graph Repetitive Network (GRCN) [14].

GNN is a method in advanced learning classes designed to make inferences on data described by graphs. GNN is a neural network that can be applied to graphics and provides an easy way to perform a forecasting or prediction task. In graph theory, we apply the concept of inserting vertices. This means providing node feature cues to the storage space. See Fig. 2 [15].

GNN has several types including graph convolutional network or (GCN), simple graph convolution (SGC), Graph Attention Network (GAN), graph recurrent network (GRCN). Graph residual network (GRAN) and Spatial Temporal Graph Neural Network (STGNN) whereas in this study it will focus more on the Spatial Temporal Graph Neural Network (STGNN). This research was first proposed by Liu and Zhon (2020) and looked at the Graph Learning Survey written by Xia et al. (2021) [16].

Spatial-Temporal Graph Neural Network (STGNN) is one type of deep machine learning in which the spatial-temporal graph structure has dynamic properties as vertex/edge features change over time. Air transport network analysis is a typical STGNN, where the airport network can be modeled with a graph structure. Specifically, each vertex represents a location that monitors route data, number of routes, days, number of passengers and airlines that change from time to time. Those are the vertex characteristics of each airport [17].

In this discussion, u and v are defined as two vertices in a graph, x_u and x_v, which are two feature vectors. Now, we will define the encoder functions $Enc(u)$ and$Enc(v)$, which transform the feature vectors into Z_u andZ_v.The current problem raises the encoder function or commonly called the encoder function, this function must be able to perform: (i) Locality (local network environment), (ii) Aggregate information, (iii) Compose many layers (calculations). Computational graphics Locality information can be achieved by using a computational graph. After the locality information maintains the computational graph, we start aggregating it using an artificial neural network, see Fig. 3. Artificial

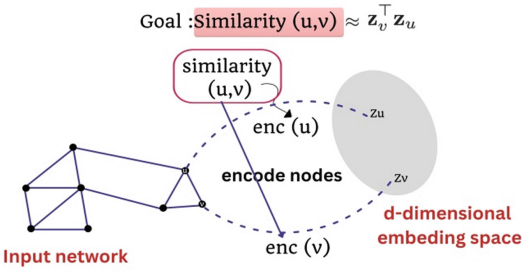

Fig. 2. The illustration of CNN on image. https://wandb.ai/syllogismos/machine

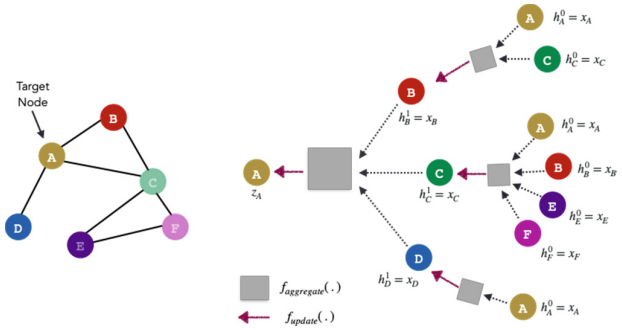

Fig. 3. The ilustration of node embedding with two layers dept. Source https://lamiae-hana.medium.com

neural networks are presented in blue and gray squares. They require aggregations to be order-invariant, such as*sum*(.), *average*(.), *maximum*(.), *concat*(.), because they are a function of GNN origin.

The purpose of this research is to apply the Spatial Temporal Graph Neural Network (STGNN) which is integrated with the Resolving Efficient Dominating Set (REDS) to analyze Airport Passengers Flow anomalies as the best and curative measure to predict airport density by Resolving Efficient Dominating Set (REDS). In Fig. 4 is the result of the comb product operation ($\triangleright$) of the complete bipartite graph (K_{33}) and the path graph (P_5). Based on Fig. 4, we can see that the resolving efficient domination number of the graph is 12. The resolving efficient dominating set of the graph ($K_{33} \triangleright P_5$) is W ($K_{33} \triangleright P_5$) = $\{x_{11},x_{14},x_{21},x_{24},x_{31},x_{34},y_{11},y_{14},y_{21}, y_{24},y_{31},y_{34}\}$. Every vertex on the set W is not adjacent to each other, and every vertex on V ($K_{33} \triangleright P_5$) W is adjacent to exactly one vertex in W, so that the set W ($K_{33} \triangleright P_5$) satisfies the definition of the efficient dominating set. The representation of each vertex on the graph ($K_{33} \triangleright P_5$) is the distance of each vertex on the graph ($K_{33} \triangleright P_5$) to the set W. It can be easily seen that each vertex in the graph ($K_{33} \triangleright P_5$) has a different representation of the distance between one another. From this explanation, it can be concluded that the graph ($K_{33} \triangleright P_5$) satisfies the definition of the resolving efficient dominating set.

2 Methods

This study uses analytical and experimental methods. In the analytical study, we use a mathematical deductive approach to describe the findings, whereas in the experimental method, we use computer programming to carry out the simulation. We will analyze airport density at 12 points in Indonesia. First, we will show the process of inserting single layer GNN nodes from a given graph with seven data features, namely location data, number of aircraft, airlines, days and number of passengers for 20 months of observation. Second, we will develop the STGNN programming, train the model using 70% of the input data obtained from the node insertion process, testing, and finally airport density forecasting.

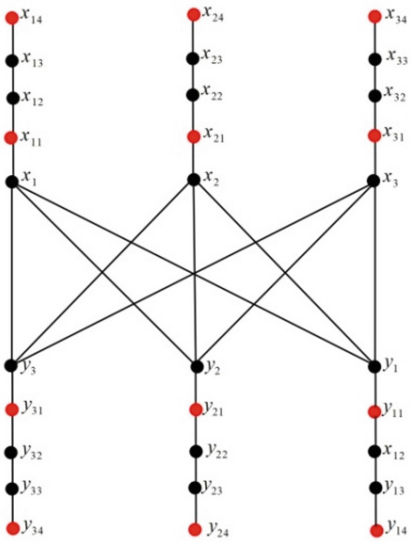

Fig. 4. The resolving efficient dominating set of ($K_{33} \triangleright P_5$).

The following is an algorithm for studying airport density anomalies using STGNN combined with Resolving Efficient Dominating Set (REDS).

Single Layer GNN Algorithm

Step 0. Given that a graph $G(V, E)$ of order n and feature matrix $Hn \times m$ of n vertices and m features and give a tolerance ε.

Step 1. Determine the matrix adjacency A of graph G and set a matrix $B = A + I$, where I is an identity matrix.

Step 2. Initialize weights W, bias β, learning rate α. (For simplicity, set $W_{m\times 1} = [W_1\ W_2 \ldots W_m]$, where $0 < W_j\ j < 1$, bias $\beta = 0$ and $0 < \alpha < 1$).

Step 3. Multiply weight matrix with vertex features, by setting a message function $m_u^1 = MSG^1(h_u^{l-1})$, for linear layer $\boldsymbol{m}_u^l = W^l\left(h_u^{l-1}\right)$.

Step 4. Aggregate the messages from vertex v's neighbors, by setting function $h_v^l = AGG^l\{m_u^{l-1},\ u \in N(v)\}$, and by applying the sum(·) function $h_u^l = SUM^1\{m_u^{l-1},\ u \in N(v)\}$ in regards with matrix B.

Step 5. Determine the error, by setting $error^l = \frac{|h_{vi} - h_{vi}||_2}{|E|}$, where v_i, v_j are any two adjacent vertices.

Step 6. Observe whether error $\leq \epsilon$ or not. If yes then stop, if not then do Step 7 to update the learning weight matrix W.

Step 7. Update the learning weight matrix by setting $W^{l+1} = w_j^l + \alpha \times z_j \times e^1$ where z_j is the sum of each column in the H_{vi}^l and divide by the number of nodes.

Step 8. Do Step 3–6 till the error $\leq \varepsilon$.

Step 9. Save the embedding results into a vector, by naming the vector file with embedding data.mat. When the data is a time series data, then do the same proses for the next time data observation.

Step 10. Load the embedding data.mat then use the time series machine learning to do forecasting.

Step 11. Have the best training, testing and forecasting results, then STOP

3 Research Findings

In discussing this topic, we will discuss the research results and explain them as follows. First, we will show analytically manually the process of inserting vertex features and coloring the Resolving Efficient Dominating Set graph and finally using data at airports to then get the STGN model to then do time series forecasting on airport density anomaly.

Observation 1. Graph G of order n. We suppose that vertex and edge sets are V (G) = {{$v_1, v_2, v_3 \ldots v_{n-1}, v_n$}} and E(G) = {$v_i v_j | v_i v_j \in$ V(G)}, respectively. Given that vertex features.

$$as follows\, H_{vi} = \begin{bmatrix} s_{1,1} & s_{1,2} & \ldots & s_{1,m} \\ s_{2,1} & s_{2,2} & \ldots & s_{2,m} \\ \ldots & \ldots & \ldots & \ldots \\ s_{n,1} & s_{n,2} & \ldots & s_{n,m} \end{bmatrix}$$

The vertex embedding can determined using the messages passing from vertex v's neighbors $h_v^l = AGG^1\{m_u^{l-1}, u \in N(v)\}$ under the aggregation **sum(·)**, thus $h_v^l = AGG^1\{m_u^{l-1}, u \in N(v)\}$ in regard to the matrix B = A + I where A, I are adjacency and identity matrix, respectively.

Proof. With graph G, we can find the neighbor matrix A. Therefore, we need to consider the proximity between each graph G therefore we need to add A to the identity matrix I and we get matrix B as follows.

$$B = A + 1 = \begin{bmatrix} b_{1,1} & b_{1,2} & \ldots & b_{1,n} \\ b_{2,1} & b_{2,2} & \ldots & b_{2,n} \\ \ldots & \ldots & \ldots & \ldots \\ b_{n,1} & b_{n,2} & \ldots & b_{n,n} \end{bmatrix}$$

Based on the existing algorithm on single layer GNN, we need to initialize the learning weights as $W^1 = [W_1 W_2 \ldots . W_m]$.The weight will be used to find the value of mvi and update the new weight in the next iteration. The process of vertex embedding of GNN can be divided into two stages, namely message passing and aggregation. In the first step we do message passing $mu^l = \mathrm{MSG}^l(h_u^{l-1})$. For linear layer $\boldsymbol{m_u^l} = w^l(h_u^{l-1})$, we have

$$\boldsymbol{m_u^l} = W^1 \cdot H_{v_1}^0$$

$$m_{v_1}^1 = \left[w_{1,1} w_{1,2} \ldots w_{1,m}\right].\left[s_{1,1} s_{1,2} \ldots s_{1,m}\right] = w_{1,} \times w_{2,1} + w_{1,2} \times s_{1,2} + \ldots w_m \times s_{1,m}$$

$$m_{v_2}^1 = \left[w_{1,1} w_{1,2} \ldots w_{1,m}\right].\left[s_{2,1} s_{2,2} \ldots s_{2,m}\right] = w_{1,} \times w_{2,1} + w_{1,2} \times s_{2,2} + \ldots w_{2,m} \times s_{2,m}$$

$$\vdots$$

$$\vdots$$

$$m^1_{v_2} = \left[w_{1,1}w_{1,2}\ldots w_{1,m}\right].\left[s_{2,1}s_{2,2}\ldots s_{2,m}\right]$$

After the process is complete, proceed to the next stage aggregation in regards with v's neighbors. By applying the aggregation **sum**$(\cdot)$, for $h^l_v = AGG^1\{m^{l-1}_u, u \in N(v)\}$ we have $h^l_v = SUM^1\{m^{l-1}_u, u \in N(v\}$ in regards with matrixB $=$ A $+$ I. The embedding vector $h^1_{v_i}$ can written as follows$h^1_{v_i}[m_{v1}; m_{v2}\cdots : m_{vn}]$. Next, we need to get an error value that shows how close the two are adjacent nodes in the embedding. The smaller the error on a value, the closer the distance between the two vertices. Error can formulate the following.

$$error^1 = error^l = \frac{|\mathrm{h_{vi}} - \mathrm{h_{vi}}||_{\mathrm{inf}}}{|\mathrm{E}|}\mathrm{where}i, j \in \{1, 2, \ldots, n\}.$$

next iteration, is updating H_i wear H_{v_i} and H_{v_i} in the previous iteration.

$$H^2_{vi} = \frac{\left[S_{1,1}S_{1,2}.\cdots S_{1,m}\right]}{\sum\left[S_{1,1}S_{1,2}.\cdots S_{1,m}\right]} \times h_{v2}$$
$$= \frac{[\left(s_{1,1} \times h_{v1}\right) + \left(s_{1,2} \times h_{v1}\right) + \cdots + \left(s_{1,m} \times h_{vl}\right)|}{\sum[s_{1,1}s_{1,2}\cdots s_{1,m}}$$
$$H^2_{vi} = \frac{\left[S_{1,1}S_{1,2}.\cdots S_{1,m}\right]}{\sum\left[S_{2,1}S_{2,2}.\cdots S_{2,m}\right]} \times h_{v2}$$
$$= \frac{[\left(s_{1,1} \times h_{v1}\right) + \left(s_{1,2} \times h_{v1}\right) + \cdots + \left(s_{1,m} \times h_{vl}\right)|}{\sum[s_{1,1}s_{1,2}\cdots s_{1,m}}$$

$$\vdots$$

$$H^2_{vi} = \frac{\left[S_{n,1}S_{n,2}.\cdots S_{n,m}\right]}{\sum\left[S_{n,1}S_{n,2}.\cdots S_{n,m}\right]} \times h_{vn}$$
$$= \frac{[\left(s_{n,1} \times h_{vn}\right) + \left(s_{n,2} \times h_{vn}\right) + \cdots + \left(s_{n,m} \times h_{vn}\right)|}{\sum[s_{n,1}s_{n,2}\cdots s_{n,m}]}$$

H_{vi} can be written like following

$$H_{vi} = [H^2_{vi}; H^2_{vi}; \cdots ; H^2_{vn}]$$

At this iteration stage, the learning weight needs to be updated. The goal is to obtain the best learning model need to remember before updating the file weights we need to find the value of z_k.

$$z_1 = \frac{\sum\left[H^2_{(1,1)}; H^2_{(2,1)}; \ldots ; H^2_{(n,1)}\right]}{n}$$

$$z_2 = \frac{\sum\left[H^2_{(1,2)}; H^2_{(2,2)}; \ldots; H^2_{(n,2)}\right]}{n}$$

$$\vdots$$

$$z_m = \frac{\sum\left[H^2_{(1,m)}; H^2_{(2,m)}; \ldots; H^2_{(n,m)}\right]}{n}$$

z_k can be written like following.

$$z_k = [z_1 z_2 \ldots z_m]$$

By initializing the learning rate (α), the learning weights can be updated as follows.

$$W^2_1 = W^1_1 + \alpha \times z_1 \times e$$

$$W^2_2 = W^1_2 + \alpha \times z_2 \times e$$

$$\vdots$$

$$W^2_m = W^1_m + \alpha \times z_m \times e$$

Thus, the learning weights can be updated as follows.

$$W^2 = W^2_1 W^2_2 \ldots W^2_m$$

In order to better understand about graph neural networks, let's give some technical examples about a particular chart with specifics features of each node/node.

Example 1. Given that a graph G of order five. Suppose that vertex and edge sets are $V(G) = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$ and $(G) = \{v_1v_3, v_1v_4, v_2v_5, v_2v_6, v_2v_7, v_4v_8\}$, respectively. Given that feature node as

$$H^0_{v_i} = \begin{bmatrix} 0.8 & 2 & 4 & 2.8 \\ 2 & 2.4 & 0.16 & 2.6 \\ 4 & 2.4 & 0.18 & 2.2 \\ 6 & 4 & 2 & 2.4 \\ 6 & 4 & 0.18 & 2 \end{bmatrix}$$

Get embedded nodes with one hidden layer with one neuron, and with minimal loss function.

Solution. Based on the graph above, we will determine adjacency matrices, Identity, and loop-adjacency as following (Fig. 5).

$$A(G) = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}, I = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

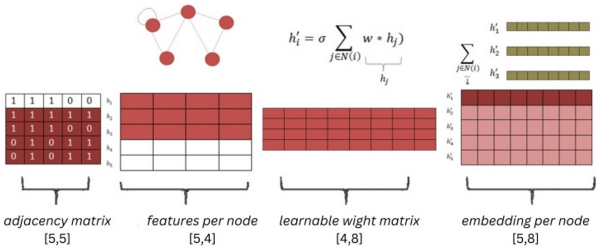

Fig. 5. The example of GNN architectures

$$B = A + I = \begin{bmatrix} 1\,1 & 1 & 1\,0 \\ 1\,1 & 1 & 1\,1 \\ 1\,1 & 1 & 1\,1 \\ 1\,1 & 1 & 1\,0 \\ 0\,1 & 1 & 0\,1 \end{bmatrix}$$

We can start manual calculations by initializing the weights on learning $W^1 = \begin{bmatrix} 0.05\ 0.05 & 0.05\ 0.05 \end{bmatrix}$ from (1, 4)-matrix.

See Image Fig. 5, in this figure the learning weight is (4, 8)-matrix, that is the number of neurons in the hidden layer is eight. First iteration the equation can be described as follows:

$$m_{v_i}^1 = W^1.H_{v_i}^{1-1}, wherei = 1, 2, 3, 4, 5, 6, 7, 8$$

$$\begin{aligned} m_{v_1}^1 &= W^1.H_{v_1}^0 \\ &= \begin{bmatrix} 0.05\ 0.05 & 0.05\ 0.05 \end{bmatrix}.\begin{bmatrix} 0.8\ 2 & 4\ 2.8 \end{bmatrix} \\ &= 0.0521 \end{aligned}$$

$$\begin{aligned} m_{v_2}^1 &= W^1.H_{v_2}^0 \\ &= \begin{bmatrix} 0.05\ 0.05 & 0.05\ 0.05 \end{bmatrix}.\begin{bmatrix} 2\ 2.4 & 0.16\ 2.6 \end{bmatrix} \\ &= 0.0517 \end{aligned}$$

$$\begin{aligned} m_{v_3}^1 &= W^1.H_{v_3}^0 \\ &= \begin{bmatrix} 0.05\ 0.05 & 0.05\ 0.05 \end{bmatrix}.\begin{bmatrix} 4\ 2.4 & 0.16\ 2.2 \end{bmatrix} \\ &= 0.0507 \end{aligned}$$

$$\begin{aligned} m_{v_4}^1 &= W^1.H_4^0 \\ &= \begin{bmatrix} 0.05\ 0.05 & 0.05\ 0.05 \end{bmatrix}.\begin{bmatrix} 6\ 2 & 4\ 2.4 \end{bmatrix} \end{aligned}$$

$$= 0.0516$$

thus, we have $m_{v_i}^1$:

$$m_{v_i}^1 = \left[\ 0.0521;\ 0.0517;\ \ 0.0507;\ 0.0516\ \right]$$

By considering the matrix B, we only include the non zeros element of $m_{v_i}^1$, thus we have

$$m_{v_1}^1 = \left[\ 0.4800;\ 0.4800;\ \ 0.4800\ \right]$$

$$m_{v_2}^1 = \left[\ 0.3580;\ 0.3580;\ \ 0.3580;\ 0.3600;\ 0.2150\ \right]$$

$$m_{v_3}^1 = \left[\ 0.4800;\ 0.4800;\ \ 0.4800\ \right]$$

$$m_{v_4}^1 = \left[\ 0.7200;\ 0.7200;\ \ 0.7200\ \right]$$

$$m_{v_5}^1 = \left[\ 0.2150;\ 0.6090;\ \ 0.6090\ \right]$$

Take the sum of the elements of each nodes embedding are as follows: $h_{v_1}^1 = 19970, h_{v_2}^1 = 2, 6060, h_{v_3}^1 = 0.7150, h_{v_4}^1 = 1.9970, h_{v_5}^1 = 1.4060$. Thus, we have the first iteration of aggregation.

$h_{v_1}^1$ =[199702, 60600.7150 1.99701.4060] *where* $i = 1, 2, 3, 4, 5$

The loss (e) can be calculated as

$$e^1 = \frac{||h_{vi}^l - h_{vj}^l||inf}{|\mathrm{E(G)}|^2} wherei, j = 1, 2, 3, 4, 5$$

$$e^1 = \frac{\|h_{v1}^1 - h_{v2}^1\| + \|h_{v1}^1 - h_{v3}^1\| + \|h_{v2}^1 - h_{v3}^1\|}{|\mathrm{E(G)}|^2}$$

$$\frac{\|h_{v2}^1 - h_{v5}^1\| + \|h_{v5}^1 - h_{v4}^1\| + \|h_{v4}^1 - h_{v2}^1\|}{|\mathrm{E(G)}|^2}$$

$$= 0.028$$

In the second iteration, we need update H_{vi}^{l-1} first:

$$H_{vi}^{l-1} = \frac{H_{vi}^{l-2}}{\Sigma(H_{vi}^{l-2})} \mathrm{x} H_{vi}^{l-1} wherei = 1, 2, 3, 4, 5$$

$$H_{v1}^1 = \frac{[0.8242.8]}{\Sigma[0.8242.8]} \times 1.9970 = [0.16640.4160 = [0.16640.41600.83210.5825]$$

$$H_{v2}^1 = \frac{[42.40.162.6]}{\Sigma[42.40.162.6]} \times 2.6060 == [0.16640.41600.83210.5825]$$

$$H_{v3}^{1} = \frac{[42.40.182.2]}{\Sigma[42.40.182.2]} \times 0.7150 == [1.18720.71230.05340.6530]$$

$$H_{v4}^{1} = \frac{[6422.4]}{\Sigma[6422.4]} \times 0.7150 == [0.83210.55470.27740.3328]$$

$$H_{v5}^{1} = \frac{[640.182]}{\Sigma[640.182]} \times 1.4060 == [0.69260.02080.12000.2309]$$

thus, we have: H_{vi}^{1}:

$$H_{vi}^{1} := \left[H_{vi}^{1}; H_{v2}^{1}; H_{v3}^{1}; H_{v4}^{1}; H_{v5}^{1}\right]$$

$$= \begin{bmatrix} 0.05960.14900.29790.2085 \\ 0.16640.41600.83210.5825 \\ 0.72790.87350.05820.9463 \\ 1.18720.71230.05340.6530 \\ 0.83210.55470.27740.3328 \\ 0.69260.46170.02080.2309 \end{bmatrix}$$

Now, we need to update the learning weight. Before that, we need take the sum of each column in the H_{vi}^{1} and divide them by the number of nodes as follows: $z_1 = 0.7213$, $z_2 = 0.6037$, $z_3 = 0.2484$, $z_4 = 0.5491$. Thus, we have $z_k = [0.72130.60370.24840.5491]$ where $k = 1, 2, 3, 4$

Given that the learning rate α, we can update the weight ω as

$$w^{1} = W_{k}^{l-1} + a \times z_k \times e^{l-1}, where k = 1, 2, 3, 4$$

$$w^{2} = W_{k}^{1} + a \times z_k \times e^{1} for a = 0, 1$$

$$w_1^2 = W_k^{l-1} + a \times z_1 \times e^1 = 0.05 + 0.1 \times 0.7213 \times 0.0286 = 0.0521$$

$$w_2^2 = W_k^{l-1} + a \times z_1 \times e^1 = 0.05 + 0.1 \times 0.6037 \times 0.0286 = 0.0517$$

$$w_3^2 = W_k^{l-1} + a \times z_1 \times e^1 = 0.05 + 0.1 \times 0.2484 \times 0.0286 = 0.0507$$

$$w_4^2 = W_k^{l-1} + a \times z_1 \times e^1 = 0.05 + 0.1 \times 0.5491 \times 0.0286 = 0.0516$$

Thus, we have W^2 :

$$W^2 = \left[\ W_1^2\ W_2^2\ \ W_3^2\ W_4^2\ \right]$$

$$= \left[\ 0.0521\ 0.0517\ \ 0.0507\ 0.0516\ \right]$$

By this new W^2 in hand, we can calculate the second iteration as follows.

$$m_{v_i}^{l} = W^1.H_{v_i}^{l-1}, wherei = 1, 2, 3, 4, 5$$

$$\begin{aligned} m_{v_1}^{2} &= W^2.H_{v_1}^{1} \\ &= \begin{bmatrix} 0.0521 & 0.0517 & 0.0507 & 0.0516 \end{bmatrix}. \\ &\quad \begin{bmatrix} 0.1664 & 0.4160 & 0.8321 & 0.5825 \end{bmatrix} \\ &= 0.1482 \end{aligned}$$

$$\begin{aligned} m_{v_2}^{2} &= W^2.H_{v_2}^{1} \\ &= \begin{bmatrix} 0.0521 & 0.0517 & 0.0507 & 0.0516 \end{bmatrix}. \\ &\quad \begin{bmatrix} 0.7279 & 0.8735 & 0.0582 & 0.9463 \end{bmatrix} \\ &= 0.1955 \end{aligned}$$

$$\begin{aligned} m_{v_3}^{2} &= W^2.H_{v_3}^{1} \\ &= \begin{bmatrix} 0.0521 & 0.0517 & 0.0507 & 0.0516 \end{bmatrix}. \\ &\quad \begin{bmatrix} 1.1872 & 0.7123 & 0.0534 & 0.6530 \end{bmatrix} \\ &= 0.1958 \end{aligned}$$

$$\begin{aligned} m_{v_4}^{2} &= W^2.H_{4}^{1} \\ &= \begin{bmatrix} 0.0521 & 0.0517 & 0.0507 & 0.0516 \end{bmatrix}. \\ &\quad \begin{bmatrix} 0.8321 & 0.5547 & 0.2774 & 0.3328 \end{bmatrix} \\ &= 0.1497 \end{aligned}$$

$$\begin{aligned} m_{v_5}^{2} &= W^2.H_{4}^{1} \\ &= \begin{bmatrix} 0.0521 & 0.0517 & 0.0507 & 0.0516 \end{bmatrix}. \\ &\quad \begin{bmatrix} 0.6926 & 0.4617 & 0.0208 & 0.2309 \end{bmatrix} \\ &= 0.1057 \end{aligned}$$

thus, we have $m_{v_i}^{2}$:

$$m_{v_i}^{2} = \begin{bmatrix} 0.1482 & 0.1955 & 0.1958 & 0.1497 & 0.1057 \end{bmatrix}$$

By considering the matrix B, we only include the non zeros element of $m_{v_i}^{2}$, thus we have

$$m_{v_1}^{2} = \begin{bmatrix} 0.1482 & 0.1482 & 0.1482 \end{bmatrix}$$

$$m_{v_2}^2 = \left[\ 0.1955\ 0.1955\ \ 0.1955\ \ 0.1955\ 0.1958\ \right]$$

$$m_{v_3}^2 = \left[0.1958\ 0.1958\ 0.1958\right]$$

$$m_{v_4}^2 = \left[0.1497\ 0.1497\ 0.1497\right]$$

$$m_{v_5}^2 = \left[0.1057\ 0.1057\ 0.1057\right]$$

Take the sum of the elements of each nodes embedding are as follows: $h_{v_1}^2 = 0.1482$, $h_{v_2}^2 = 0.1955$, $h_{v_3}^2 = 0.1958$, $h_{v_4}^2 = 0.1497$, $h_{v_5}^2 = 0.1057$. Thus, we have the second iteration of aggregation

$$h_{v_i}^2 = \left[\ 0.1482;\ 0.1955;\ \ 0.1958;\ \ 0.1497;\ 0.1057\ \right]$$

$$wherei = 1, 2, 3, 4, 5$$

The loss (e) can be calculated as

$$e^2 = \frac{\|h_{v_i}^l - h_{v_j}^l\|_{inf}}{|E(G)|} wherei, j \in \{1, 2, 3, 4, 5\} = 0.0060$$

3.1 Nodes Embedding

Research uses data without normalization as well as data with normalization. We use alpha = 0.1, iterations 20, 25 and 30, the initial weights are 0.5, 0.3 and 0.1. This experiment can be seen in Table 1. In the table, the results show that the normalized arrival has a relatively decreasing error at each iteration whereas on data that is not normalized, it can be concluded that the error always increases at each iteration. The error comparison can be seen through the image below.. In addition, the accuracy in choosing the weight must also be considered. Based on 3 weight values that we use, the initial weight of 0.1 is the weight that produces the smallest error value.

3.2 Time Series Forecasting Analysis

Now, we will provide computer simulations on two neural network architectures to train, test and density data at airports, the data will be in the form of a time series obtained through GNN embedding in the previous stage, to make this step easier we use mathlab tools perform numerical simulations. Can be seen in Fig. 6. The architecture used in this study is ANN-746 and ANN-665. While the ANN models used are Feedforwardnet, Paternnet, Fitnet, and Cascadeforwardnet. Parameters used to train the network training function Lavenberg- Marquadt, sigmoid log transfer.

Function and sigmoid hyperbolic tangent, epoch count of 750, and learning rate 0.1. The results of this training and testing are presented in Table 2. Best Indicator the model

Table 1. The results of GNN nodes embedding for finding the best loss

Data Type	Iteration Numbers	Learning Weight	Error Value
Non Normalization	12	0.7	3.1131×10^{11}
	16	0.3	1.6137×10^{31}
	20	0.1	3.9989×10^{12}
		0.7	1.5637×10^{95}
		0.3	1.3563×10^{151}
		0.1	2.3769×10^{72}
		0.7	18×10^{168}
		0.3	1.9×10^{151}
		0.1	4.4×10^{112}
Normalization	12	0.7	9.6404×10^{06}
	16	0.3	0.9139
	20	0.1	0.8134
		0.7	98.1097
		0.3	0.9139
		0.1	0.0185
		0.7	6.9588×10^{202}
		0.3	5.8892×10^{125}
		0.1	0.0024×10^{-5}

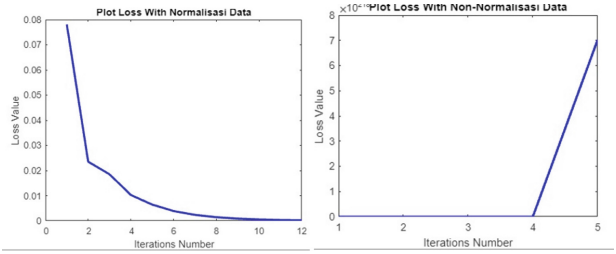

Fig. 6. (left) Plot loss with normalize data and (right) plot loss with non- normalize data

is the model that produces the smallest mean square error (MSE). So, based on the Table 2, the best model is produced by ANN-746 with the cascadeforwardnet model, with an MSE test of 1.0328×10^{-09}.

In addition to performance indicators, we also show plots from training, testing, and forecasting to find out the location of data anomalies. The results of our research are shown in Figs. 7, 8, and 9. Based on Fig. 8, in this figure it can be seen that in the following week there will be data anomalies is in the 2nd data. The regression that we produce is also 0.9917. Plots from This regression can be seen in Fig. 9.

Table 2. The performance indicator of ANN architectures and models on training and testing Airport Passengers Flow data set.

ANN Model	ANN Architecture	MSE TRAIN	Regression Train	Time Train	MSE Test
Feed Forwartnet	ANN-(a) 466	5.5129×10^{-10}	0.9997	1.7693*s*	1.4871×10^{-08}
	ANN-(a) 567	$3.9490 \times 10^{-0,8}$	0.9998	1.3222*s*	11.4500×10^{-08}
Fitnet	ANN- (b) 466	4.6468×10^{-08}	0.99928	2.4292*s*	4.6468×10^{-08}
	ANN- (b) 567	4.6022×10^{-08}	0.99929	1.2286*s*	4.6022×10^{-08}
Paternet	ANN- (c) 466	1.7993×10^{-08}	0.99916	4.1392*s*	1.7993×10^{-08}
	ANN- (c) 567	2.7797×10^{-08}	0.99917	2.4391*s*	2.7797×10^{-08}
Cascadeforwartnet	ANN- (d) 466	1.2386×10^{-08}	0.99981	8.2378*s*	1.2386×10^{-08}
	ANN- (d) 567	1.4500×10^{-08}	0.99977	2.8274*s*	1.0328×10^{-09}

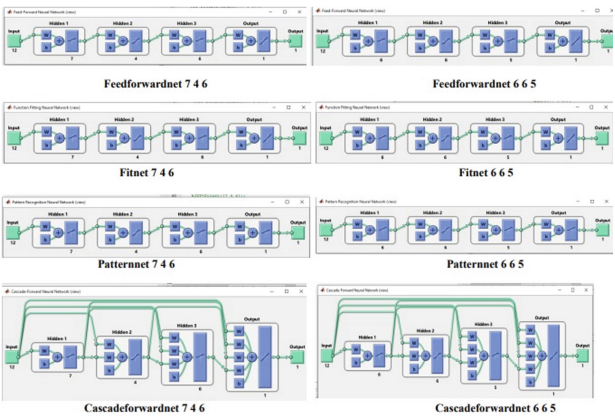

Fig. 7. The comparation of ANN architecture which used to get the training model.

4 Discussion

In this study, we have obtained a density anomaly at the airport. in this process the features that were previously 6 will change to 1 this stage is to get the embedding results, we need to process aggregation and message passing. So we use matrix B as a reference next, when the message has been sent, we sum the messages in aggregation process. To measure how close two vertices have been in aggregation, we use error values as a benchmark. In the study, the best.

Error value is 0.0024×10^{-5} generate using weights of 0.1 and 20 iterations. Furthermore, after processing the input data, we predict the data in the form of a time series using ANN. This stage has three stages, the first is training, the second is testing and the last is forecasting. There are four models and two ANN architectures that we use during the training. At the training stage will be obtained in network training.

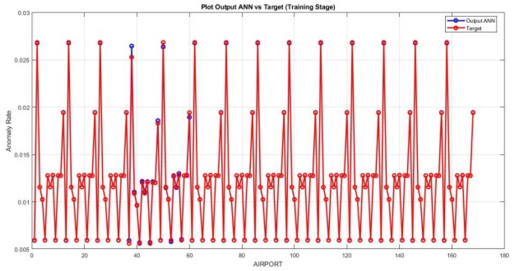

Fig. 8. The comparison of ANN output and target data of the Airport Passengers Flow on training stage. The illustration shows that ANN output interpolates uniformly on the data target

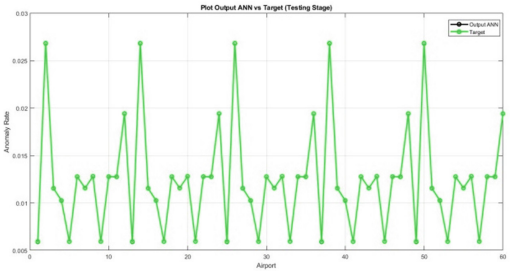

Fig. 9. The comparison of ANN output and target data of the flood flow river on testing stage. The illustration shows that ANN output almost interpolates uniformly on the data target.

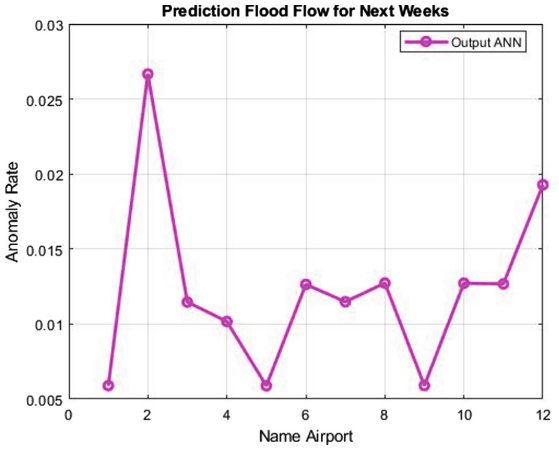

Fig. 10. The forecasting of flood flow river for determining the biggest river flow. The illustration shows that the biggest river flow is in river number 13, or in other words is Bedadung River.

A model is obtained which is then used in the testing phase. The model that has been built is then tested at the testing stage to measure ANN performance. As a benchmark, we use the mean square error (MSE). Based on the test results, the best ANN model is

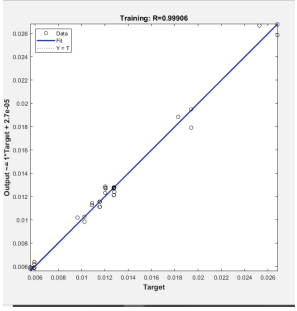

Fig. 11. This regression shows that the model we use in this study is accurate

Cascadeforwardnet with the 756 architecture. Next By using the training model, data on flood discharge forecasts for the following week is obtained. This forecast is shown in Fig. 10. Based on the figure, the anomaly point is the 2nd airport (Fig. 11).

5 Concluding Remarks

There are many uses of machine learning, one of which is used in agriculture, especially in sensor planting to facilitate watering of plants, this is important to do as an effort to minimize the use of human labor, in line with this, machine learning is felt to be able to present a new breakthrough, especially in agriculture, So, the use of IoT in agriculture will support the Sustainable Development Goals (SDGs) program.

Acknowledgment. For supporting this research, we are delighted to PUI-PT Combinatorics and Graph, CGANT, University of Jember in 2023.

References

1. BUDAK, Ümit; ŞENGÜR, Abdulkadir; HALICI, Uğur. Deep convolutional neural networks for airport detection in remote sensing images. In: 2018 26th Signal Processing and Communications Applications Conference (SIU). Ieee, 2018. p. 1–4.
2. CHEN, Zhuyun, et al. A deep learning method for bearing fault diagnosis based on cyclic spectral coherence and convolutional neural networks. Mechanical Systems and Signal Processing, 2020, 140: 106683.
3. Choi, Sun, and Young Jin Kim. "Artificial neural network models for airport capacity prediction." Journal of Air Transport Management 97 (2021): 102146.
4. Dafik, I. H Agustin, M. Hasan, R. Adawiyah, R. Alfarisi and D. A. R Wardani. On the Locating Edge Domination Number of Comb Product of Graphs. In Journal of Physics: Conference Series. Vol. 1022. No. 1. p. 012003. IOP Publishing. 2018.
5. Deng, Yun-Ping, et al. Efficient dominating sets in circulant graphs. Discrete Mathematics, 2017, 340.7: 1503–1507.
6. S.,Fernández-González, P. Bolgiani,Fernández-Villares, González, García-Gil, A. Suárez & A. Merino (2019). Forecasting of poor visibility episodes in the vicinity of Tenerife Norte Airport. Atmospheric Research, 223, 49–59, (2019).

7. Guo, Xiaojia, Yael Grushka-Cockayne, and Bert De Reyck. "Forecasting airport transfer passenger flow using real-time data and machine learning." Manufacturing & Service Operations Management 24.6 (2022): 3193–3214
8. R. A., Hakim, et al. "On resolving efficient domination number of comb product of special graphs." Journal of Physics: Conference Series. Vol. 1832. No. 1. IOP Publishing, 2021.
9. Tien, Shin-Lai, et al. "Using ensemble weather forecasts for predicting airport arrival capacity." Journal of Air Transportation 26.3 (2018): 123–132.
10. Jin, Guangyin, et al. "STGNN-TTE: Travel time estimation via spatial–temporal graph neural network." Future Generation Computer Systems 126 (2022): 70–81.
11. I., Kusumawardani, et al. On resolving efficient domination number of path and comb product of special graph. In: Journal of Physics: Conference Series. IOP Publishing, 2022. p. 012012.
12. LI, Yansheng, et al. Multi-label remote sensing image scene classification by combining a convolutional neural network and a graph neural network. Remote Sensing, 2020, 12.23: 4003.
13. Orsini, Federico, et al. Neural networks trained with WiFi traces to predict airport passenger behavior. In: 2019 6th international conference on models and Technologies for Intelligent Transportation Systems (MT-ITS). IEEE, 2019. p. 1–7.
14. Peng, Liang, et al. Reverse graph learning for graph neural network. IEEE Transactions on Neural Networks and Learning Systems, 2022.
15. SHAO, Zezhi, et al. Pre-training Enhanced Spatial-temporal Graph Neural Network for Multivariate Time Series Forecasting. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2022. p. 1567–1577.
16. Tran, Viet-Linh; Thai, Duc-Kien; Kim, Seung-Eock. Application of ANN in predicting ACC of SCFST column. Composite Structures, 2019, 228: 111332.
17. Wang, Zhanwei, and Woon-Kyung Song. "Sustainable airport development with performance evaluation forecasts: A case study of 12 Asian airports." Journal of Air Transport Management 89 (2020): 101925