On the Spatial Temporal Graph Neural Network Analysis Together with Local Vertex Irregular Reflexive Coloring for Time Series Forecasting on Passenger Density at Bus Station

Adidtiya Dwi Harliyuni[1], Dafik[1,2](✉), Slamin[2,5], Zainur Rasyid Ridlo[2,3], and Ridho Alfarisi[2,4]

[1] Department of Mathematics Education Postgraduate, University of Jember, Jember, Indonesia
adidtiyadwiharliyuni@gmail.com, d.dafik@unej.ac.id
[2] PUI-PT Combinatorics and Graph, CGANT, University of Jember, Jember, Indonesia
{slamin,zainur.fkip,alfarisi.fkip}@unej.ac.id
[3] Department of Science Education, University of Jember, Jember, Indonesia
[4] Department of Elementary Education, University of Jember, Jember, Indonesia
[5] Department of Computer Science, University of Jember, Jember, Indonesia

Abstract. The transportation problem that occurs in urban areas is how to meet the demand for the increasing number of trips and avoid traffic jams on the highway. In Indonesia, traffic density occurs during office hours, holidays, and national holidays. The solution to this problem is to use an effective public transportation service, one of which is the bus. Infrastructure for bus transportation includes roads, bridges, bus stops, and bus station. Bus station is one of the transportation systems that has the main function as a place to stop public transportation to pick up and drop passengers to the final destination of the trip. In this paper, we discuss the application of the concept of Spatial Temporal Graph Neural Network (STGNN) together with Local Vertex Irregular Reflexive Coloring (LVIRC) to analyze the passengers density anomaly of in bus station. The results shows that the use of the Spatial Temporal Graph Neural Network with Local Vertex Irregular Reflexive Coloring (LVIRC) are effective tools for forcasting the passengers density anomaly with the best model is generated by ANN-657 cascadeforwardnet, with a test MSE of 9.4982 $\times 10^{-9}$.

Keywords: spatial temporal graph neural network · local vertex irregular reflexive coloring · time series forecasting · passengers density anomaly

1 Introduction

The transportation problem that occurs in Indonesia's urban areas is how to meet the increasing demand for travel, which does not cause traffic jams on the roads. This is in line with economic development and the growing number of middle and upper middle class people in urban areas as seen from the development of existing urban transportation. The solution to this problem is to use effective public transportation services, one of which is the bus [13].

The bus is one of the public transportation media for traveling medium and long distances. A prediction/forecasting system is needed that can identify the surge in passengers that may occur during the holidays. Therefore, the handling of these problems must be carried out systematically starting from prevention and curative action.

The application of Artificial Neural Networks is one way to analyze and can prevent the large number of passengers not being carried by the fleet by providing the number of fleets according to the known number of passengers. In the Artificial Neural Networks input data perspective, the bus transport flows are considered to be a Non-Euclidean data sources. To handel with this type of data source, the use of Graph Neural Network (GNN) is considered to be a breakthrough in the machine learning. By a *graph*, we mean a structure $G = (V(G), E(G))$, where $V(G)$ is a finite nonempty set of elements called *vertices*, and $E(G)$ is a set (possibly empty) of unordered pairs $\{u, v\}$ of vertices $u, v \in V(G)$, called *edges* [8]. The number of vertices of a graph G is the *order* of G, commonly denoted by $|V(G)|$. The number of edges is the *size* of G, often denoted by $|E(G)|$. A graph G that has order $p = |V(G)|$ and size $q = |E(G)|$ is sometimes called a (p, q)-*graph*. Let $u, v \in V(G)$, vertex u is said to be *adjacent* to v if there is an edge e between u and v, that is, $e = uv$. Vertex v is then called a *neighbor* of u. The set of all neighbors of u is called the *neighborhood* of u and is denoted by $N(u)$. We also say that u and v are *incident* with edge e. The *adjacency matrix* of a graph G and vertex-set $V(G) = \{v_1, v_2, \ldots, v_n\}$ is the $n \times n$ matrix $A = [a_{ij}]$, where $a_{ij} = \begin{cases} 1 \text{ if } v_iv_j \in E(G), \\ 0 \text{ otherwise.} \end{cases}$

The Graph Neural Network (GNN) technique adopts the framework of the Convolutional Neural Networks (CNN). Convolutional Neural Network (CNN) is a well-known deep learning architecture inspired by the natural visual perception mechanism of the living creatures [9]. The core concept behind CNN introduces hidden convolution and pooling layers to identify spatially localized features via a set of receptive fields in kernel form. The convolution takes a little sub-patch of the image (a little rectangular part of the image), applies a function to it, and produces a new part (a new pixel) (Fig. 1).

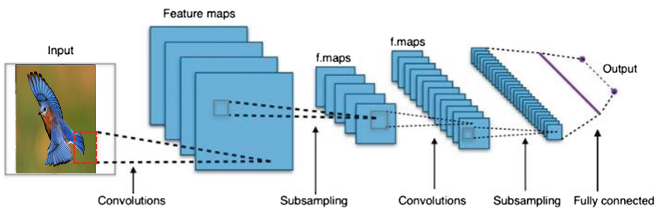

Fig. 1. The illustration of CNN Source: https://commons.wikimedia.org

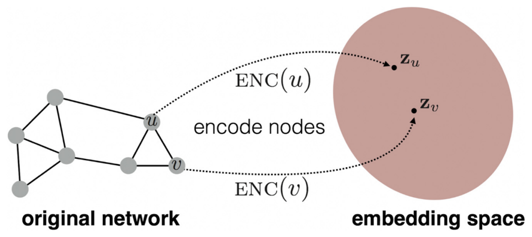

Fig. 2. The illustration of node embedding with single layer Source: shorturl.at/fjM39

Graph Neural Networks are a type of machine learning algorithm that can extract important information from a network to make useful predictions [14] with graphics, it becomes more pervasive and richer with information, and Artificial Neural Networks also become more capable. GNN has become a powerful tool for many important applications other than ANNs. GNN is a class of deep learning methods designed to perform inferences on data explained by graphs. As well as a neural network that can be directly applied to graphs, and provides an easy way to do this node-level, edge-level, and graph-level prediction tasks. In this theory, we apply the *node embedding* concept, it means maps vertex features to the $d-$dimensional (low dimensional space rather than the actual dimensions of the graph). The goal is to map the vertices so that they are similar in embedding space approximates similarity in the graphs (Fig. 2).

The graph neural networks (GNN) has been introduced as a new Depth Learning paradigm for learning non-Euclidean data by applying graph analysis methods. GNN's can be categorized into four groups, namely Recurrent GNNs (RecGNNs), ConvulaTION GNNs (ConvGNNs), Graph Autoencoders (GAEs), and Spatial Temporal GNNs (STGNNs). Spatial Temporal Graph Neural Networks (STGNNs) is a type of GNN in which the concept is based on simultaneously spatial model and temporal dependencies to deal with a dynamic graph problem. In other words, the Spatial Temporal Graph Neural Network (STGNN) is one of the deepest machine learning type in which the spatial-temporal graph structure resides dynamic because node/edge features change over time

Traffic flow forecasting problem, flood flow forecasting problem, supply chain management problem, and precision agriculture problem, etc. are some of the examples of STGNN cases. A bus station analysis of transport flow networks is a typical application of STGNN, where bus station network can be modeled with a graphical structure. In particular, each node represents a bus station location that monitors the number of lanes, number of buses, type of bus, time of day, weather, and number of passengers, which changes from time to time. Those are the node feature of each bus station.

Let us define u and v as two vertices in a graph, x_u and x_v are two feature vectors. Now, we will define the encoder function $Enc(u)$ and $Enc(v)$, which convert the feature vectors to z_u and z_v. The challenge now is how to come up with the encoder function? The encoder function should be able to perform: (i) Locality (local network neighborhoods), (ii) Aggregate information, (iii) Stacking multiple

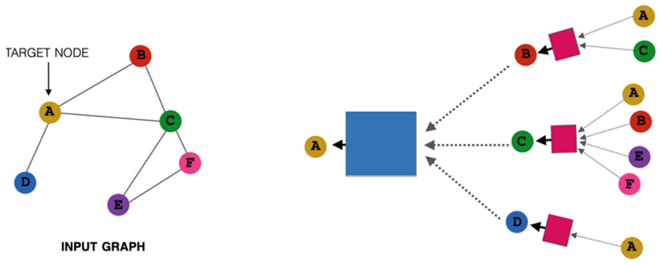

Fig. 3. The illustration of node embedding with two layers Source: shorturl.at/fjM39

layers (computation). Locality information can be achieved by using a computational graph. Once the locality information preserves the computational graph, we start aggregating [10]. This is basically done using neural networks, see Fig. 3. Neural Networks are presented in blue and pink boxes. They require *aggregations* to be order-invariant, like **sum(.), average(.), maximum(.), concat(.)**, because they are permutation-invariant functions [15] [16].

In this study we apply the Spatial Temporal Graph Neural Network (STGNN) integrated with the Local Vertex Irregular Reflexive Coloring (LVIRC) to analyze bus transportation flow anomalies as a preventive and curative actions to overcome the increase in the number of passengers. By local vertex irregular reflexive coloring of graph, that a function $f : V(G) \rightarrow \{0, 2, ..., 2k \in N_v\}$ and $f : E(G) \rightarrow \{1, 2, ..., k \in N_e\}$, where $k = max\{k \in N_e, 2k \in N_v\}$ for $k \in N_v, k \in N_e$ are natural number. The associated weight of a vertex $u, v \in V(G)$ under f is $w(u) = f(u) + \Sigma \in N_{uv \in E(G)} f(uv)$. The function f is called a local vertex irregular reflexive k-labelling if every two adjacent vertices has distinct weight. When we assign each vertex of G with a color of the vertex weight $w(uv)$, thus we say the graph G admits a local vertex irregular reflexive coloring. The smallest number of vertex weights needed to color the vertices of G such that no two adjacent vertices share the same color is called a local vertex irregular reflexive chromatic number, denoted by $\chi_{lrvs}(G)$. Furthermore, the minimum k required such that $\chi_{lrvs}(G) = \chi(G)$ is called a local reflexive vertex color strength, denoted by $lrvcs(G)$ [1]- [7], [11]- [12], [17]. In Fig. 4, we give an illustration of local vertex irregular reflexive coloring of windmill graph.

2 Methods

This research uses analytical and experimental methods. In the analytical study, we use mathematical deductive approach to describe the findings, whilst in the experimental method, we use a computer programming to do simulation. We will analyze the anomaly of the bus transportation flow of twelve bus stations in East Java, Indonesia. First, we will show the vertex embedding process of single layer GNN of a given graph with six features data, namely number of lines, number of buses, type of bus, time of day, weather, and number of passengers for 17 weeks

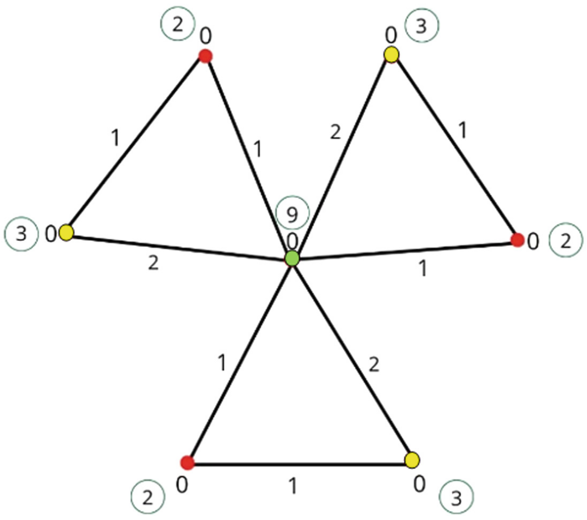

Fig. 4. The illustration of local vertex irregular reflexive coloring of windmill graph

observation. Second, we will develop the STGNN programming, train a model using 70% data input obtained from the vertex embedding process, testing and finally forecast passengers density anomaly.

The following is the algorithm for studying bus transportation flow anomalies using STGNN combined with local vertex irregular reflexive coloring.

Single Layer GNN Algorithm

```
Step 0. Given that a graph $G(V, E)$ of order
    $n$ and feature matrix $H_{n\times m}$ of $n$
    vertices and $m$ features, and give
    a tolerance $\epsilon$.
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 $\beta$,
    learning rate $\alpha$. (For simplicity,
    set $W_{m\times 1} = [w_1\ w_2\ \ldots\ w_m]$, where $0 <$
    $w_j < 1$, bias $\beta = 0$ and $0 < \alpha < 1$)
Step 3. Multiply weight matrix with vertex
    features, by setting a message
    function $\mathbf{m_u}^l = MSG^l(h_u^{l-1})$, for
    linear layer $\mathbf{m_u}^l = W^l(h_u^{l-1})$.
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**$(\cdot)$ function $h_v^l = SUM^l\{m_u^{l-1}, u \in N(v)\}$ in regards with matrix $B$.
*Step 5.* Determine the error, by setting $error^l = \frac{||h_{v_i} - h_{v_j}||_2}{|E|}$, 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 $W^{l+1} = W_j^l + \alpha \times z_j \times e^l$ where $z_j$ is the sum of each column in the $H_{v_i}^l$ and divide by the number of nodes.
*Step 8.* Do Step 3-6 till the $error \leq \epsilon$.
*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

We will discuss the research result and describe in the following. We first show analytically the embedding process of vertex features and the local vertex irregular reflexive coloring of graph, and the last use the bus transportation data to obtain the STGNN model to do time series forecasting on bus transportation flow anomaly.

Observation 1 Given that a graph G of order n. Suppose that vertex and edge sets are $V(G) = \{v_1, v_2, \ldots, v_{n-1}, v_n\}$ and $E(G) = \{v_iv_j | v_i, v_j \in V(G)\}$, respectively. Given that vertex features as follows

$$H_{v_i} = \begin{bmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,m} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n,1} & s_{n,2} & \cdots & s_{n,m} \end{bmatrix}.$$

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

Proof. By graph G, we can determine the matrix adjacency A. Since, we need to consider the self adjacency for each vertex of G then we need to add A by identity matrix I and we have matrix B as follows.

$$B = A + I = \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n,1} & b_{n,2} & \cdots & b_{n,n} \end{bmatrix}$$

According to the single layer GNN algorithm, we need to initialize the learning weights as $W^1 = [w_1 \; w_2 \; \ldots \; w_m]$. This weight will be used to obtain the value of m_{v_i} 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 $m_u^l = MSG^l(h_u^{l-1})$. For linear layer $m_u^l = W^l(h_u^{l-1})$, we have $m_{v_i}^1 = W^1 \cdot H_{v_i}^0$

$$\begin{aligned} m_{v_1}^1 &= [w_{1,1} \; w_{1,2} \; \ldots \; w_{1,m}] \cdot [s_{1,1} \; s_{1,2} \; \ldots \; s_{1,m}] \\ &= w_1 \times s_{1,1} + w_2 \times s_{1,2} + \cdots + w_m \times s_{1,m} \\ m_{v_2}^1 &= [w_{1,1} \; w_{1,2} \; \ldots \; w_{1,m}] \cdot [s_{2,1} \; s_{2,2} \; \ldots \; s_{2,m}] \\ &= w_{1,1} \times s_{2,1} + w_{1,2} \times s_{2,2} + \cdots + w_{1,m} \times s_{2,m} \\ &\vdots \\ m_{v_n}^1 &= [w_{1,1} \; w_{1,2} \; \ldots \; w_{1,m}] \cdot [s_{n,1} \; s_{n,2} \; \ldots \; s_{n,m}] \end{aligned}$$

Once the above process has been done we do the second step, namely aggregation in regards with v's neighbors. By applying the aggregation **sum**$(\cdot)$, for $h_v^l = AGG^l\{m_u^{l-1}, u \in N(v)\}$ we have $h_v^l = SUM^l\{m_u^{l-1}, u \in N(v)\}$ in regards with matrix $B = A + I$. The embedding vector h_{v_i} can written as follows: $h_{v_i}^1 = [m_{v_1}; m_{v_2}; \ldots; m_{v_n}]$. Next, we need to obtain the error value that indicates how close the two adjacent vertices in the embedding space. The smaller the error value, the closer the distance between the two vertices. The error can be formulated as follows.

$$error^l = \frac{||h_{v_i} - h_{v_j}||_{inf}}{|E(G)|} \quad \text{where } i, j \in \{1, 2, ..., n\}$$

In the next iteration, we need to update H_{v_i} using H_{v_i} and h_{v_i} in the previous iteration.

$$\begin{aligned}
H^2_{v_1} &= \frac{[s_{1,1}\ s_{1,2}\ \dots\ s_{1,m}]}{\sum[s_{1,1}\ s_{1,2}\ \dots\ s_{1,m}]} \times h_{v_1} \\
&= \frac{[(s_{1,1} \times h_{v_1}) + (s_{1,2} \times h_{v_1}) + \dots + (s_{1,m} \times h_{v_1})]}{\sum[s_{1,1}\ s_{1,2}\ \dots\ s_{1,m}]} \\
H^2_{v_2} &= \frac{[s_{2,1}\ s_{2,2}\ \dots\ s_{2,m}]}{\sum[s_{2,1}\ s_{2,2}\ \dots\ s_{2,m}]} \times h_{v_2} \\
&= \frac{[(s_{2,1} \times h_{v_2}) + (s_{2,2} \times h_{v_2}) + \dots + (s_{2,m} \times h_{v_2})]}{\sum[s_{2,1}\ s_{2,2}\ \dots\ s_{2,m}]} \\
&\vdots \\
H^2_{v_n} &= \frac{[s_{n,1}\ s_{n,2}\ \dots\ s_{n,m}]}{\sum[s_{n,1}\ s_{n,2}\ \dots\ s_{n,m}]} \times h_{v_n} \\
&= \frac{[(s_{n,1} \times h_{v_n}) + (s_{n,2} \times h_{v_n}) + \dots + (s_{n,m} \times h_{v_n})]}{\sum[s_{n,1}\ s_{n,2}\ \dots\ s_{n,m}]}
\end{aligned}$$

H_{v_i} can be written like following.

$$H_{v_i} = [H^2_{v_1};\ H^2_{v_2};\ \dots;\ H^2_{v_n}]$$

In this iteration the learning weights need to be updated. The goal is to get the best learning model. However, before updating the weights we need to find the z_k value.

$$\begin{aligned}
z_1 &= \frac{\sum[H^2_{(1,1)}; H^2_{(2,1)}; \dots; H^2_{(n,1)}]}{n} \\
z_2 &= \frac{\sum[H^2_{(1,2)}; H^2_{(2,2)}; \dots; H^2_{(n,2)}]}{n} \\
&\vdots \\
z_m &= \frac{\sum[H^2_{(1,m)}; H^2_{(2,m)}; \dots; H^2_{(n,m)}]}{n}
\end{aligned}$$

z_k can be written like following.

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

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

$$\begin{aligned} W_1^2 &= W_1^1 + \alpha \times z_1 \times e \\ W_2^2 &= W_2^1 + \alpha \times z_2 \times e \\ &\vdots \\ W_m^2 &= W_m^1 + \alpha \times z_m \times e \end{aligned}$$

Thus, the learning weights can be updated as follows.

$$W^2 = [W_1^2 \; W_2^2 \; \dots \; W_m^2]$$

To have more understanding about the graph neural network, let us give some technical example about a specific graph with a specific feature of each vertex/node.

Example. Given that a graph G of order five. Suppose that vertex and edge sets are $V(G) = \{v_1, v_2, v_3, v_4, v_5\}$ and $V(G) = \{v_1v_2, v_1v_3, v_2v_3, v_2v_4, v_2v_5, v_4v_5\}$, respectively. Given that feature node as

$$H_{v_i}^0 = \begin{bmatrix} 0.61 & 0.69 & 0.50 & 0.10 \\ 0.27 & 0.44 & 0.10 & 0.10 \\ 0.50 & 0.61 & 0.10 & 0.10 \\ 0.50 & 0.61 & 0.90 & 0.10 \\ 0.10 & 0.10 & 0.10 & 0.10 \end{bmatrix}$$

Obtain the nodes embedding with one hidden layer with one neuron, and with minimal loss function.

Solution. By the above graph, we can determine the adjacency, Identity, and loop-adjacency matrices as follows.

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

We can start the technical calculation by initiating the learning weight $W^1 = \begin{bmatrix} 0.05 & 0.05 & 0.05 & 0.05 \end{bmatrix}$ of $(1, 4)$-matrix. See Fig. 5, in this figure learning weight is $(4, 8)$-matrix, meaning the number on neuron in the hidden layer is eight. The first iteration of Equation Fig. 5 can be described as follows:

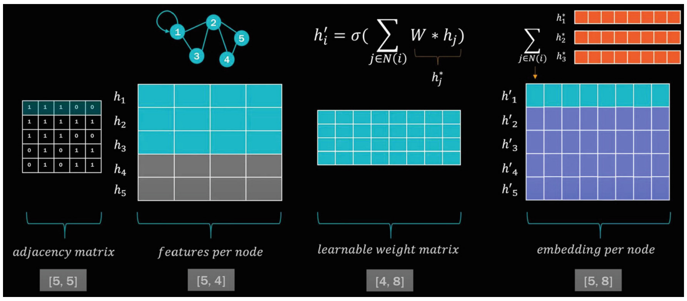

Fig. 5. The example of GNN architectures: https://www.youtube.com/watch?v=A-yKQamf2Fc

$$\begin{aligned}
m^l_{v_i} &= W^l \cdot H^{l-1}_{v_i}, \quad \text{where } i = 1, 2, 3, 4, 5 \\
m^1_{v_1} &= W^1 \cdot H^0_{v_1} \\
&= [0.05\ 0.05\ 0.05\ 0.05] \cdot [0.61\ \ 0.69\ \ 0.50\ \ 0.10] \\
&= 0.0950 \\
m^1_{v_2} &= W^1 \cdot H^0_{v_2} \\
&= [0.05\ 0.05\ 0.05\ 0.05] \cdot [0.27\ \ 0.44\ \ 0.10\ \ 0.10] \\
&= 0.0455 \\
m^1_{v_3} &= W^1 \cdot H^0_{v_3} \\
&= [0.05\ 0.05\ 0.05\ 0.05] \cdot [0.50\ 0.61\ 0.10\ 0.10] \\
&= 0.0655 \\
m^1_{v_4} &= W^1 \cdot H^0_{v_4} \\
&= [0.05\ 0.05\ 0.05\ 0.05] \cdot [0.50\ 0.61\ 0.90\ 0.10] \\
&= 0.1055 \\
m^1_{v_5} &= W^1 \cdot H^0_{v_5} \\
&= [0.05\ 0.05\ 0.05\ 0.05] \cdot [0.10\ 0.10\ 0.10\ 0.10] \\
&= 0.0200
\end{aligned}$$

thus, we have $m^1_{v_i}$:

$$m^1_{v_i} = [0.0950;\ \ 0.0455;\ \ 0.0655;\ \ 0.1055;\ \ 0.0200]$$

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

$$\begin{aligned}
m_{v_1}^1 &= [0.0950;\ 0.0455;\ 0.0655]\\
m_{v_2}^1 &= [0.0950;\ 0.0455;\ 0.0655;\ 0.1055;\ 0.0200]\\
m_{v_3}^1 &= [0.0950;\ 0.0455;\ 0.0655]\\
m_{v_4}^1 &= [0.0455;\ 0.1055;\ 0.0200]\\
m_{v_5}^1 &= [0.0455;\ 0.1055;\ 0.0200]
\end{aligned}$$

Take the sum of the elements of each nodes embedding are as follows: $h_{v_1}^1 = 0.2060, h_{v_2}^1 = 0.3315, h_{v_3}^1 = 0.2060, h_{v_4}^1 = 0.1710, h_{v_5}^1 = 0.1710$. Thus, we have the first iteration of aggregation

$$h_{v_i}^1 = [0.2060\ 0.3315\ \ 0.2060\ \ 0.1710\ \ 0.1710]$$
$$\text{where } i = 1, 2, 3, 4, 5.$$

The loss (e) can be calculated as

$$\begin{aligned}
e^l &= \frac{||h_{v_i}^l - h_{v_j}^l||_{inf}}{|E(G)|^2} \quad \text{where } i, j \in \{1, 2, 3, 4, 5\}\\
e^1 &= \frac{||h_{v_1}^1 - h_{v_2}^1|| + ||h_{v_1}^1 - h_{v_3}^1|| + ||h_{v_2}^1 - h_{v_3}^1||}{|E(G)|}\\
e^1 &= \frac{+||h_{v_2}^1 - h_{v_5}^1|| + ||h_{v_5}^1 - h_{v_4}^1|| + ||h_{v_4}^1 - h_{v_2}^1||}{|E(G)|}\\
&= 0.0047
\end{aligned}$$

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

$$\begin{aligned}
H_{v_i}^{l-1} &= \frac{H_{v_i}^{l-2}}{\sum(H_{v_i}^{l-2})} \times h_{v_i}^{l-1}, \quad \text{where } i = 1, 2, 3, 4, 5\\
H_{v_1}^1 &= \frac{[0.61\ 0.69\ 0.50\ 0.10]}{\sum[0.61\ 0.69\ 0.50\ 0.10]} \times 0.2060\\
&= [0.0661\ 0.0748\ 0.0542\ 0.0108]\\
H_{v_2}^1 &= \frac{[0.27\ 0.44\ 0.10\ 0.10]}{\sum[0.27\ 0.44\ 0.10\ 0.10]} \times 0.3315\\
&= [0.0984\ 0.1603\ 0.0364\ 0.0364]\\
H_{v_3}^1 &= \frac{[0.50\ 0.61\ 0.10\ 0.10]}{\sum[0.50\ 0.61\ 0.10\ 0.10]} \times 0.2060\\
&= [0.0786\ 0.0959\ 0.0157\ 0.0157]\\
H_{v_4}^1 &= \frac{[0.50\ 0.61\ 0.90\ 0.10]}{\sum[3\ 2\ 1\ 1.2]} \times 0.1710\\
&= [0.0405\ 0.0494\ 0.0729\ 0.0081]\\
H_{v_5}^1 &= \frac{[0.10\ 0.10\ 0.10\ 0.10]}{\sum[0.10\ 0.10\ 0.10\ 0.10]} \times 0.1710\\
&= [0.0428\ 0.0428\ 0.0428\ 0.0428]
\end{aligned}$$

thus, we have $\mathfrak{H}^1_{v_i}$:

$$H^1_{v_i} = [H^1_{v_1};\ H^1_{v_2};\ H^1_{v_3};\ H^1_{v_4};\ H^1_{v_5}]$$

$$= \begin{bmatrix} 0.0661\ 0.0748\ 0.0542\ 0.0108 \\ 0.0984\ 0.1603\ 0.0364\ 0.0364 \\ 0.0786\ 0.0959\ 0.0157\ 0.0157 \\ 0.0405\ 0.0494\ 0.0729\ 0.0081 \\ 0.0428\ 0.0428\ 0.0428\ 0.0428 \end{bmatrix}$$

Now, we need to update the learning weight. before that, we need take the sum of each column in the $H^1_{v_i}$ and divide them by the number of nodes as follows: $z_1 = 0.0652, z_2 = 0.0846, z_3 = 0.0444, z_4 = 0.0227$. Thus, we have $z_k = [0.0652\ 0.0846\ 0.0444\ 0.0227]$ where $k = 1, 2, 3, 4$. Given that the learning rate α, we can update the weight w as

$$\begin{aligned} W^l &= W^{l-1}_k + \alpha \times z_k \times e^{l-1}, \quad \text{where } k = 1, 2, 3, 4 \\ W^2 &= W^1_k + \alpha \times z_k \times e^1, \quad \text{for } \alpha = 0.1 \\ W^2_1 &= W^1_1 + \alpha \times z_1 \times e^1 \\ &= 0.05 + 0.1 \times 0.0652 \times 0.0047 = 0.0500 \\ W^2_2 &= W^1_2 + \alpha \times z_2 \times e^1 \\ &= 0.05 + 0.1 \times 0.0846 \times 0.0047 = 0.0500 \\ W^2_3 &= W^1_3 + \alpha \times z_3 \times e^1 \\ &= 0.05 + 0.1 \times 0.0444 \times 0.0047 = 0.0500 \\ W^2_4 &= W^1_4 + \alpha \times z_4 \times e^1 \\ &= 0.05 + 0.1 \times 0.0227 \times 0.0047 = 0.0500t \end{aligned}$$

Thus, we have W^2:

$$\begin{aligned} W^2 &= [W^2_1\ W^2_2\ W^2_3\ W^2_4] \\ &= [0.0500\ 0.0500\ 0.0500\ 0.0500] \end{aligned}$$

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

$$\begin{aligned}
m^l_{v_i} &= W^l \cdot H^{l-1}_{v_i}, \quad \text{where } i = 1,2,3,4,5 \\
m^2_{v_1} &= W^2 \cdot H^1_{v_1} \\
&= [0.0500\ 0.0500\ 0.0500\ 0.0500] \cdot \\
&\quad [0.0661\ 0.0748\ 0.0542\ 0.0108] \\
&= 0.0103 \\
m^2_{v_2} &= W^2 \cdot H^1_{v_2} \\
&= [0.0500\ 0.0500\ 0.0500\ 0.0500] \cdot \\
&\quad [0.0984\ 0.1603\ 0.0364\ 0.0364] \\
&= 0.0166 \\
m^2_{v_3} &= W^2 \cdot H^1_{v_3} \\
&= [0.0500\ 0.0500\ 0.0500\ 0.0500] \cdot \\
&\quad [0.0786\ 0.0959\ 0.0157\ 0.0157] \\
&= 0.0103 \\
m^2_{v_4} &= W^2 \cdot H^1_{v_4} \\
&= [0.0500\ 0.0500\ 0.0500\ 0.0500] \cdot \\
&\quad [0.0405\ 0.0494\ 0.0729\ 0.0081] \\
&= 0.0086 \\
m^2_{v_5} &= W^2 \cdot H^1_{v_5} \\
&= 0.0500\ 0.0500\ 0.0500\ 0.0500] \cdot \\
&\quad [0.0428\ 0.0428\ 0.0428\ 0.0428] \\
&= 0.0086
\end{aligned}$$

thus, we have $m^2_{v_i}$:

$$m^2_{v_i} = [0.0103\ 0.0166\ 0.0103\ 0.0086\ 0.0086]$$

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

$$\begin{aligned}
m^2_{v_1} &= [0.0103\ 0.0166\ 0.0103] \\
m^2_{v_2} &= [0.0103\ 0.0166\ 0.0103\ 0.0086\ 0.0086] \\
m^2_{v_3} &= [0.0103\ 0.0166\ 0.0103] \\
m^2_{v_4} &= [0.0166\ 0.0086\ 0.0086] \\
m^2_{v_5} &= [0.0166\ 0.0086\ 0.0086]
\end{aligned}$$

Take the sum of the elements of each nodesZ embedding are as follows: $h^2_{v_1} = 0.0372, h^2_{v_2} = 0.0543, h^2_{v_3} = 0.0372, h^2_{v_4} = 0.0337, h^2_{v_5} = 0.0337$. Thus, we have the second iteration of aggregation$h^2_{v_i}$=[0.0372; 0.0543; 0.0372; 0.0337;

0.0337]
where i=1,2,3,4,5.

The loss (e) can be calculated as

$$e^2 = \frac{||h^l_{v_i} - h^l_{v_j}||_{inf}}{|E(G)|} \text{ where } i, j \in \{1, 2, 3, 4, 5\} = 0.0062$$

3.1 Nodes Embedding

In this study, we used data with normalization. We use alpha = 0.1, iterations of 4, 7, and 10, initial weights are 0.5, 0.3, and 0.1. The results of this experiment can be seen in Table 1. Based on the table, it can be seen that in the normalized data the error value tends to decrease in each iteration. Based on the 3 weight values that we use, the initial weight of 0.1 is the weight that produces the smallest error value (Fig. 6).

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

Data Type	Iteration Numbers	Learning Weight	Error Value
Non-Normalization	4	0.5	6.9×10^{14}
		0.3	6.8×10^{12}
		0.1	3.4×10^{8}
	7	0.5	17×10^{128}
		0.3	1.3×10^{120}
		0.1	3.3×10^{104}
	10	0.5	17×10^{128}
		0.3	1.3×10^{120}
		0.1	3.3×10^{104}
Normalization	4	0.5	1.6194
		0.3	0.2255
		0.1	0.0067
	7	0.5	9.3124
		0.3	1.8634
		0.1	0.0062
	10	0.5	1.7×10^{28}
		0.3	2.4×10^{27}
		0.1	**1.6234** $\times 10^{-6}$

ANN Model	ANN Architecture	MSE Train	Regression Train	Time Train	MSE Test
Feedforwardnet	ANN-(a)657	8.8447×10^{-8}	0.38273	0.64514 s	1.1782×10^{-7}
	ANN-(a)746	6.3391×10^{-8}	0.6495	0.48781 s	9.5504×10^{-8}
Fitnet	ANN-(b)657	8.2661×10^{-8}	0.45934	0.82325 s	1.1201×10^{-8}
	ANN-(b)746	6.3380×10^{-8}	0.60173	0.54041 s	1.0428×10^{-7}
Paternet	ANN-(c)657	1.0719×10^{-8}	0.46934	0.86498 s	9.9737×10^{-8}
	ANN-(c)746	3.7654×10^{-8}	0.61173	0.86515 s	7.2084×10^{-8}
Cascadeforwardnet	ANN-(d)657	3.2651×10^{-9}	0.98338	2.2961 s	$\mathbf{9.4982} \times 10^{-9}$
	ANN-(d)746	4.2894×10^{-9}	0.9779	1.5937 s	1.0431×10^{-8}

Fig. 6. The performance indicator of ANN architectures and models on training and testing bus transportation flow data set.

3.2 Time Series Forecasting Analysis

Now, we will perform computer simulation on two neural network architectures to train, test, and forecast passengers density anomaly data in East Java. The data on the time series is obtained from the GNN embedding process in the previous stage. We use Matlab programming to perform numerical simulations. A comparison of this architecture can be seen in Fig. 7. The neural network architecture that we use is ANN-657 and ANN-746. While the ANN models that we use are Feedforwardnet, Paternnet, Fitnet, and Cascadeforwardnet. The parameters that we use in training the network are the Lavenberg-Marquadt training function, the sigmoid log transfer function and the sigmoid hyperbolic tangent, the number of epochs 750, and the learning rate 0.1. The results of this training and testing are presented in Table 1. The best model indicator is the model that produces the smallest mean square error (MSE). So, based on the table, the best model is generated by ANN-657 cascadeforwardnet, with a test MSE of 9.4982×10^{-9} (Fig,9).

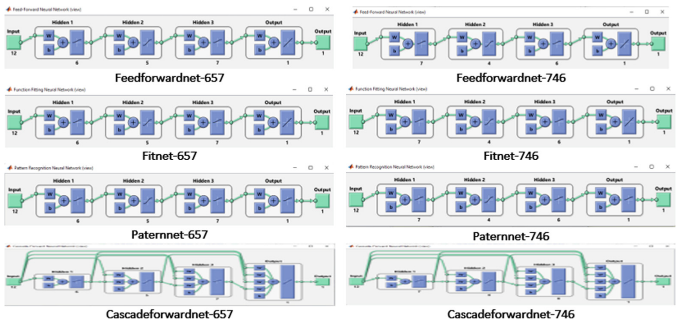

Fig. 7. The comparison of ANN architecture which we used to get the training model.

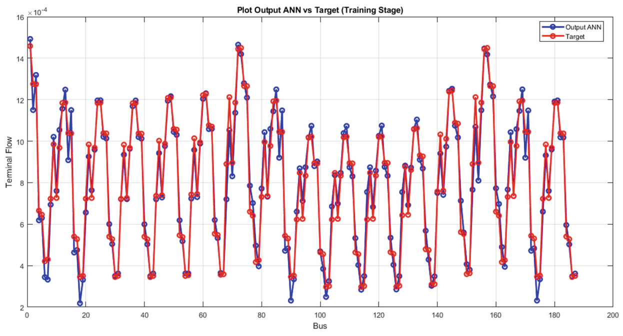

Fig. 8. The comparison of ANN output and target data of the passengers density of bus transportation flow on training stage.

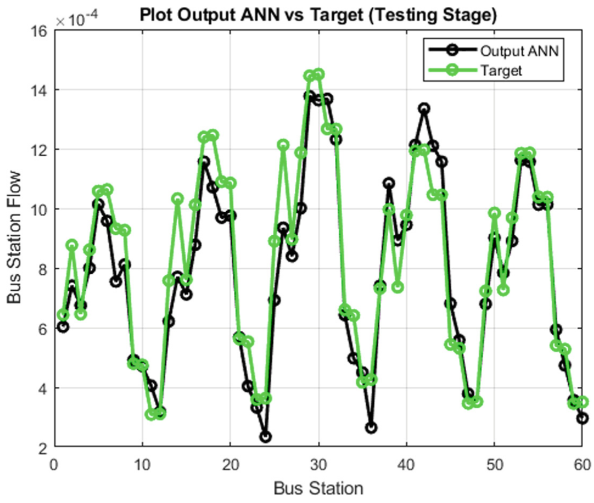

Fig. 9. The comparison of ANN output and target data of the passengers density of bus transportation flow on testing stage.

In addition to performance indicators, we also show plots from training, testing, and forecasting to find out where the anomalous data is located. The results of our study are shown in Fig. 8 - Image 10. Based on Fig. 10, we know that in the following week the anomaly data is in the 6th data. The regression that we produce is also good, namely 0.98338. The plot of this regression can be seen in Fig. 11.

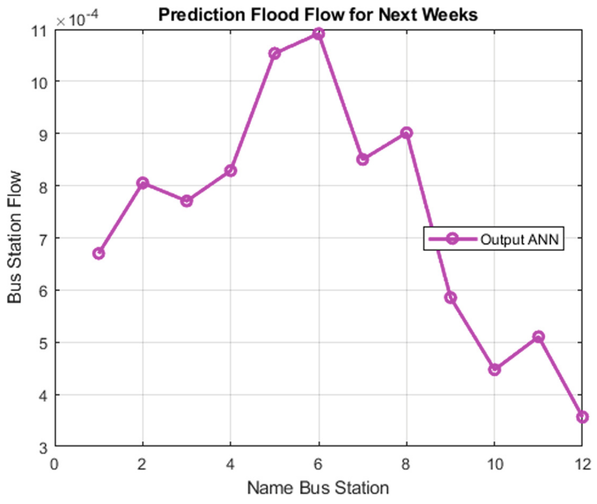

Fig. 10. The forecasting of bus transportation flow for determining the biggest passengers density. The illustration shows that the biggest passengers density is in number 6, or in other words is Probolinggo Station.

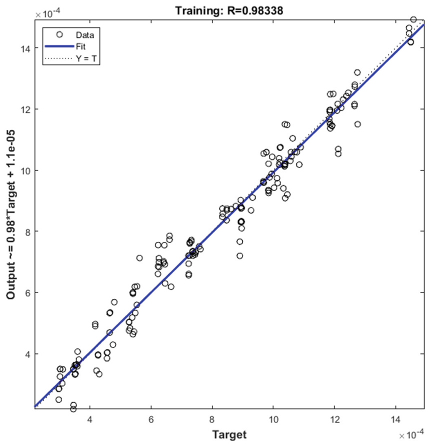

Fig. 11. This regression shows that the model we use in this study is accurate.

Discussion

In this research, we have succeeded in embedding the bus transportation flow graph in East Java. This embedding process can reduce the original feature dimension from 6 to 1. Of course, to get the embedding results, we need to process message passing and aggregation. In the message passing process, we assume that each node has information or messages that will be sent to other nodes. So in this process we consider the adjacency matrix. However, we do not only consider the relationship of one node with adjacent vertices. But also the relationship of the node to itself. So we use matrix B as a reference in this message passing process. Next, when the messages have been sent, we add up the messages in the aggregation process. To measure how close two vertices have been processed by aggregation, we use error value as a benchmark. In this study, the best error value is 1.6234 $\times 10^{-6}$ generated using a weight of 0.1 and 10 iterations.

Furthermore, after obtaining the embedded data, we forecast the time series data using ANN. At this stage there are three parts, namely training, testing, and forecasting. There are four models and two ANN architectures that we used during the training. In network training we obtained a model which was then used in the testing phase. The model that has been built is then tested at the testing stage to measure the performance of the ANN. As a benchmark, we use the mean square error (MSE). Based on the test results, the best ANN model is Cascadeforwardnet with 657 architecture. Furthermore, by using the training model we obtain bus transportation flow data forecasting in the following week. This forecast is shown in Fig. 10. Based on the picture, the anomaly point is on the 6th bus station.

Concluding Remarks

Transportation is an important and strategic means of development in facilitating the wheels of the economy and the smooth running of community activities. The bus is transportation with multiple comparative advantages such as being mass in nature, adaptive to the main tasks, and the function of mobilizing the flow of passengers and goods. Then, one of the public transportation media for traveling medium and long distances. Another thing that is no less important is the need for smooth transportation. Thus, a prediction/forecasting system is needed that can identify the surge in passengers. In line with this, machine learning is felt to be able to present a new breakthrough, especially in transportation. So, the use of IoT in transportation will support the system transportation

Acknowledgment. We would like to thank PUI-PT Combinatorics and Graph, CGANT-University of Jember 2023 for the funding towards this research.

References

1. Agustin I H, Dafik, Utoyo M I, Slamin, and Venkatachalam M 2021 The Relexive Edge Strength on Some Almost Regular Graphs Heliyon 7
2. Agustin I H, Utoyo M I, Dafik, and Venkatachalam M 2020 Edge irregular reflexive labeling of some tree graphs Journal of Physics: Conference Series 1543
3. Agustin I H, Utoyo M I, Dafik, Venkatachalam M and Surahmat 2020 On the construction of the refexive vertex k-labeling of any graph with pendant vertex International Journal of Mathematics and Mathematical Sciences 2020 7812812

4. Alfarisi R, Ryan J, Siddiqui M K, Dafik and Agustin I H 2021 Vertex irregular reflexive labeling of disjoint union of gear and book graphs Asian-European Journal of Mathematics 14(5) 2150078
5. Ali, A., Baca, M.: On Vertex Irregular Total Labelings Ars Combinatoria **112**, 129–139 (2013)
6. A'yun Q, Dafik, Adawiyah R, Agustin I H and Albirri E R 2021 On the irregular coloring of bipartite graph and tree graph families J. Phys.: Conf. Ser. 1836 012024
7. Baca M, Irfan M, Ryan J, Semanicova-Fenovcikova A, Tanna D 2017 On Edge Irregular Reflexive Labellings for the Generalized Friendship Graphs Mathematics 5(67)
8. Gallian, J.A.: A Dynamic Survey of Graph Labeling The Electronic Journal of Combinatorics **20**, 1–432 (2017)
9. Gu, J., et al.: Recent Advances in Convulational Neural Networks Science. Direct **77**, 354–377 (2018)
10. Levin, O.: Discrete Mathematics an Open Introduction, 3rd edn. University of Northern Colorado, Greeley (2019)
11. Mursyidah I L, Dafik, Adawiyah R, Kristiana A I and Agustin I H 2021 On local irregularity vertex coloring of comb product on star graphs J. Phys.: Conf. Ser. 1836 012023
12. Nisviasari R, Dafik and Agustin I H 2020 The total H-irregularity strength of triangular ladder graphs Journal of Physics: Conference Series 1465 012026
13. Munawar A 2006 Perencanaan Angkutan Umum Perkotaan Berkelanjutan UNISIA 59 XXIX
14. Rahman R F 2015 Forecasting Rail Passenger Volume in Java-Sumatera Island Using Backprogation Artificial Neural Network Method. Universitas Pendidikan Indonesia. upi.edu perpustakaan.upi.edu
15. Sarker, I.H.: Machine Learning: Algorithms, Real-World Applications and Research Directions. SN Computer Science **2**(3), 1–21 (2021). https://doi.org/10.1007/s42979-021-00592-x
16. Sarker IH, Kayes ASM. 2020. Abc-ruleminer: user behavioral rulebased machine learning method for context-aware intelligent services. J Netw Comput Appl, page 102762
17. Tanna D, Ryan J, Semanicova-Fenovcikova A and Baca M 2018 Vertex Irregular Refexive Labeling of Prisms and Wheels AKCE International Journal of Graphs and Combinatorics