Lightweight Model Using Graph Neural Networks for Air Quality Impact Assessment on Human Health

Tang Nguyen-Tan[1,2]
[1]*Faculty of Computer Networks and Communications, University of Information Technology, Ho Chi Minh city, Vietnam*
[2]*Vietnam National University, Ho Chi Minh City, Vietnam*
19522181@gm.uit.edu.vn

Quan Le-Trung[1,2]
[1]*Faculty of Computer Networks and Communications, University of Information Technology, Ho Chi Minh city, Vietnam*
[2]*Vietnam National University, Ho Chi Minh City, Vietnam*
quanlt@uit.edu.vn

***Abstract*—Current air monitoring stations can only tell us the purity or pollution of the air through the Air Quality Index (AQI). Using the Air Quality Index is not enough when we need to monitor the air environment in areas such as factories, industrial parks, and space areas where many people are working. In this article, we propose a lightweight deep learning model to assess the impact of air quality on human health using the following layers: i) The Timestamp Graph Message Passing layer for graph-based data aggregation affecting prediction results; ii) The Timestamp Graph Convolution layer for graph-based feature expansion affecting prediction results; iii) The Temporal Graph Message Passing layer, which is used to aggregate data based on time series graphs; iv) The Temporal Graph Convolution layer for feature extraction based on time series graphs; v) The Simple Artificial Neural Network for graph classification based on data of nodes. The input is four time series containing data on CO, NO_2, O_3, and PM 2.5 concentrations in the air. The output of the model is one of the following cases: Fresh, Polluted, Headaches, Pneumonia, Convulsions or Nausea, or Death. The results show that our model can be used for early warning of adverse human health impacts due to air pollution, which has achieved an accuracy of up to 98.15% for the samples in the training set and 91.86% when performing the evaluation with the samples in the test set.**

***Index Terms*—Edge computing, IoT, Deep Learning, Time series data, GNN, GCN, GGNN, Air Quality Index, TensorFlow Lite.**

I. Introduction

In recent decades, environmental pollution has been one of the most talked about issues in the media and at major economic, political, cultural, and social conferences around the world. In the world, there have been many air monitoring stations built to measure and calculate the Air Quality Index in the area where the station is located. We can look up the Air Quality Index very easily online or better yet, we can easily buy an Air Quality Index meter with just a matchbox on e-commerce sites to use. Until we designed a system to measure and monitor air quality in an area where many people were working, we realized that using the Air Quality Index alone was not enough because the most important thing in places like factories and industrial parks is to maintain a fresh air environment, good for human health. In fact, it seems that current air monitoring stations only give indicators of air quality and concentrations of harmful substances in the air. If we need to know the effect of these indicators on human health then we have to compare them with medical standards ourselves, which is quite time-consuming. Because of the above problems, we decided to carry out this study to build a lightweight deep learning model that helps predict the effects of CO, NO_2, O_3 and PM 2.5 concentrations in the air environment on human health. That model can be deployed on Gateway devices similar to Raspberry Pi in air monitoring systems at industrial parks, factories, or places where many people work for early warning of adverse human health impacts due to air pollution.

Artificial Intelligence-based models have been widely used in air quality forecasting. Air environment data collected through the Internet of Things system is time series data stored in a time series database. Currently, deep learning models for input data in time series format are usually based on Recurrent Neural Network (RNN), Long short-term memory (LSTM), or Gated Recurrent Unit (GRU). Using variations of the LSTM model can predict the Air Quality Index [1]–[4]. The combination of Gray Level Co-occurrence Matrix (GLCM), Support Vector Regression (SVR), and LSTM is another solution for deep learning methods to predict Air Quality Index [5]. In the above studies, the proposed deep learning models can all be built using TensorFlow and must then be converted to TensorFlow Lite before being deployed to Edge devices via the HTTP protocol, which will reduce the latency and increase the real-time for predictions [6]. Recently, there has been research using Graph Neural Networks (GNNs) to determine air quality by representing collected data about the air environment in relational graphs [10], [11]. Therefore, the Graph Neural Network is used to handle the relationships of the data in the model we propose to predict the impact of air quality on human health.

In this paper, we propose a new approach to predict air quality using Graph Neural Networks. Our Graph Neural networks use two graphs to handle the relationships of the

input data. The first is G_{EP} which describes each relationship between CO, NO_2, O_3, PM 2.5, and the predicted outcome at each timestamp. The second is that the G_{TS} represents the relationship between the timestamps of the input data in time series format. The input data for our model are four time series containing data on CO, NO_2, O_3, and PM 2.5 concentrations in the air. The output of the model is one of the following cases: Fresh, Polluted, Headaches, Pneumonia, Convulsions or Nausea, or Death.

This paper has the following structure: Section I introduces an overview of the current research, the rationale for this study, and the proposed Graph Neural Network model. Contextual analysis and the relationships of the input data to the predicted outcomes will be provided in Section II. In Section III, we propose our Graph Neural Network model. Section IV shows the results and evaluates the accuracy of the proposed model. Finally, Section V ends this paper with a conclusion and future development directions.

II. Contextual And Data Analysis

Edge computing plays an important role in IoT systems that integrate Artificial Intelligence for data exploration and intelligent control. Such systems are often referred to as Artificial Intelligence of Things systems. The growth of Edge computing has brought the possibility of bringing Artificial Intelligence-based models closer to where data is collected. That has opened up a huge potential for using Artificial Intelligence-based models to explore data from IoT systems [7]. The standard network topology for modern IoT systems typically consists of Cloud servers, Edge servers, IoT devices, and more. Based on that standard network topology we get the deployment context given below.

A. Implementation Context

The deployment context is mainly concerned with edge network components, so we remove the extraneous components to obtain the network topology model as shown in Figure 1. The data about CO, NO_2, O_3, and PM 2.5 concentrations are collected by the sensors mounted on front-end Internet of Things (IoTs) devices and then transmitted to the Raspberry Pi which acts as a Gateway through the wireless connection. The transmitted data frame contains all four data values of the four time series at the current timestamp. The Gateway that receives the data then processes and forwards that data to the Edge server through the internet connection.

Edge Server will receive the results returned from the Gateway and then store them. Users will use the Mobile App or Web Browser to access the Edge server to view the display data. The Graph Neural Network model will be trained on the Edge server and sent back to the Gateway via HTTP protocol. The model training process uses a combination of time series data stored on the Cloud and the Edge server. The model will be used by the Gateway to make predictions and return the results to the Edge server.

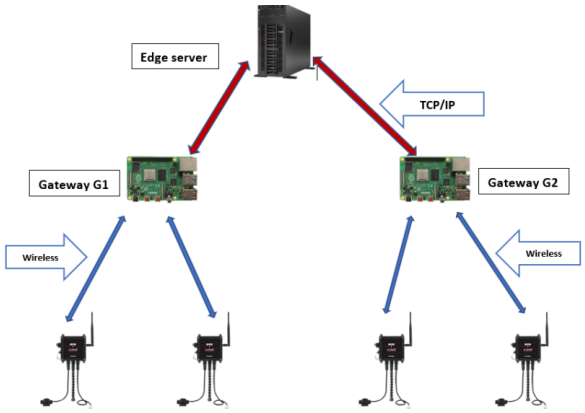

Fig. 1. The implementation context of the AQI prediction system.

B. Data Feature Analysis

In this research, four time series have been used to extract features, then aggregate them and make predictions about the impact on human health. We have made Table 1 based on Vietnam's ambient air quality standards in 2013 and the knowledge we have found about the effects of CO, NO_2, O_3, PM 2.5 concentrations in the air environment on human health. On that basis, the input data will be pre-processed during training, evaluation, and prediction.

TABLE I
The Effects Of CO, NO_2, O_3, PM 2.5 Concentrations In The Air Environment On Human Health

Sensor	Value	Range 0-1	Label	Measure
CO	0-30	0-0.16	Fresh	ppm
	30-100	0.16-0.32	Polluted	
	100-800	0.32-0.48	Headaches	
	800-1600	0.64-0.8	Convulsions or Nausea	
	1600 or more	0.8-1	Death	
NO_2	0-0.2	0-0.16	Fresh	
	0.2-50	0.16-0.32	Polluted	
	50-150	0.48-0.64	Pneumonia	
	150 or more	0.8-1	Death	
O_3	0-0.2	0-0.16	Fresh	
	0.2-1	0.16-0.32	Polluted	
	1 or more	0.8-1	Pneumonia	
PM 2.5	0-0.00005	0-0.16	Fresh	$\mu g/cm^3$
	0.00005 or more	0.8-1	Pneumonia	

There are two datasets used by us to train and evaluate the model: The first is Beijing Multi-Site Air-Quality Dataset provided by Kaggle. The second is the datasets collected by the air monitoring station using Libelium Smart Cities PRO v3.0 at the VNUHCM - University of Information Technology. Both datasets contain data that are the timestamps of the four time series used in this research. Based on Table 1, we performed relabeling for two datasets. The interval between two timestamps in the first dataset is one hour while in the second dataset it is five minutes, so when labeling the first

dataset, we inserted seven new timestamps after each existing timestamp.

Despite using a combination of both datasets, the labels are still numerically heterogeneous, because some labels fall into the rare case while others are the frequent case in real life. We solved the problem of data balancing in the training and evaluation sets by using Numpy to randomly generate the missing cases on the basis of Table 1. The time series data contains information about the concentrations of CO, NO_2, O_3, and PM 2.5 in the air with different units of measurement. It is for that reason that we perform data preprocessing to convert all input data to the range from 0 to 1 as shown in Table 1. After solving the above problems, we have a dataset with the ratio of labels as shown in Figure 2.

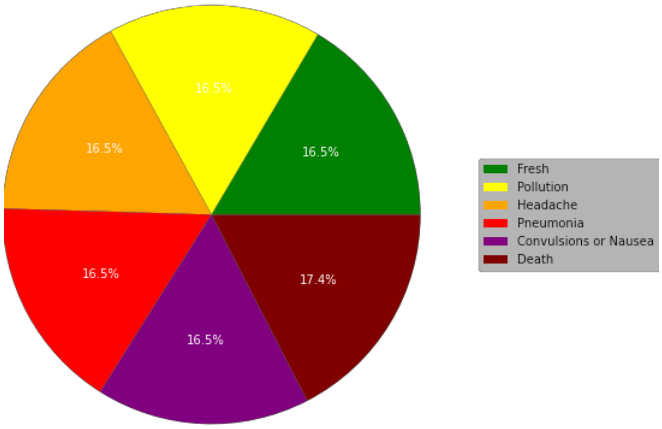

Fig. 2. The ratio of labels.

III. Proposed Model

The Graph Neural Networks model we propose is inspired by the Graph Convolutional Networks model (GCNs) and Gated Graph Sequence Neural Networks model (GGNNs) [12]. GCN is one of the best Graph Neural Networks architectures [8], [13]. GGNNs is a variant of GNNs for data of a sequential nature or time series format [14]–[16]. Our model is deployed on the Raspberry Pi acting as the Gateway with the network context shown in figure 1 for early warning of adverse human health impacts due to air pollution.

We designed a Graph Neural Network model that includes: The Timestamp Graph Message Passing layer, The Timestamp Graph Convolution layer, The Temporal Graph Message Passing layer and The Temporal Graph Convolution layer. The input to our model is a time series graph represented by two G_{EP} and G_{TS} graphs as shown in Figures 3 and 4. The graph data through a Graph Neural Network to synthesize information and extract features. Those features will be the input to a simple Artificial Neural Network to obtain the output of the following cases: Fresh, Polluted, Headaches, Pneumonia, Convulsions or Nausea, and Death.

A. Graphs Represent Data

The graph in Figure 3 is denoted by G_{EP}. At each timestamp, we have four nodes representing the four data values of CO, NO_2, O_3, and PM 2.5 concentrations in the air.

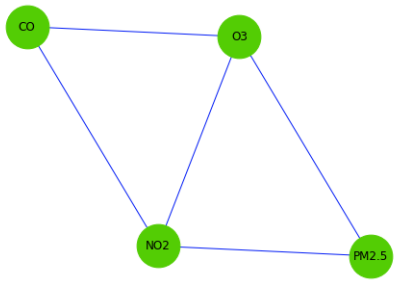

Fig. 3. The graph of the effect on prediction results at each timestamp.

If two data values represented by two nodes are likely to lead to the same prediction result, there will be an edge joining between them. Based on Table 1, we get the graph of the effect on prediction results at each timestamp as shown in Figure 3.

We used eight consecutive timestamps for the prediction, so for each prediction, we need to define a second graph to represent the temporal relationship of the eight timestamps. Eight timestamps are equal to using data over an interval of forty minutes to make predictions, which is a long enough time for air pollution to have adverse effects on human health. This second graph is denoted by G_{TS}. G_{TS} is a graph with nodes G_{EP} and edges connecting the nodes to represent a temporal relationship, which is a characteristic relationship of time series data. The graph in figure 4 is G_{TS}.

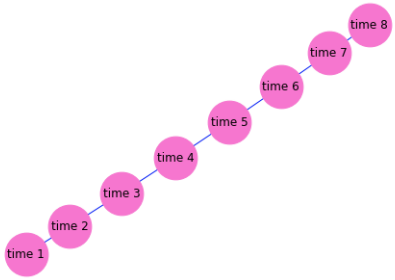

Fig. 4. The graph of the temporal relationship.

N is the number of nodes of G_{EP} and M is the number of nodes of G_{TS}:

- $A_{EP} \in R^N$ is the adjacency matrix of G_{EP}.
- $D_{EP} \in R^N$ is the degree matrix of G_{EP}.
- $A_{TS} \in R^M$ is the adjacency matrix of G_{TS}.
- $D_{TS} \in R^M$ is the degree matrix of G_{TS}.

B. Graph Message Passing

It is very necessary to perform matrix normalization before aggregating data on the nodes of the graph [9].

$$\widehat{A}_{\text{EP}} = A_{\text{EP}} + I_{\text{EP}} \quad \widehat{A}_{\text{TS}} = A_{\text{TS}} + I_{\text{TS}} \tag{1}$$

$$\widehat{D}_{\text{EP}}{}^{-\frac{1}{2}} = (D_{\text{EP}} + I_{\text{EP}})^{-\frac{1}{2}} \quad \widehat{D}_{\text{TS}}{}^{-\frac{1}{2}} = (D_{\text{TS}} + I_{\text{TS}})^{-\frac{1}{2}} \tag{2}$$

Where I_{EP} and I_{TS} are the identity matrices of G_{EP} and G_{TS}. From (1) and (2) we get the normalized matrix L_{EP} and L_{TS}.

$$L_{\text{EP}} = \widehat{D}_{\text{EP}}{}^{-\frac{1}{2}} \widehat{A}_{\text{EP}} \widehat{D}_{\text{EP}}{}^{-\frac{1}{2}} \tag{3}$$

$$L_{\text{TS}} = \widehat{D}_{\text{TS}}{}^{-\frac{1}{2}} \widehat{A}_{\text{TS}} \widehat{D}_{\text{TS}}{}^{-\frac{1}{2}} \tag{4}$$

There are two stages of information aggregation on the nodes corresponding to two graphs G_{EP} and G_{TS}. The first stage is to aggregate the information in each timestamp. This phase we named GMPEP. The second stage is to aggregate information on a graph of eight timestamps based on time series relationships. This phase we named GMPTS.

The GMPEP stage is the Timestamp Graph Message Passing layer, which aggregates the information of each time stamp, where W_{EP} is the weight matrix that will change during training.

$$U = NM \quad W_{\text{EP}} \in R^{\text{UxU}} \tag{5}$$

From (3) calculate L_{GMPEP} by the direct sum of M matrices L_{EP}.

$$L_{\text{GMPEP}} = L_{\text{EP}} \oplus L_{\text{EP}} \oplus L_{\text{EP}} \oplus \dots \tag{6}$$

Where $b \in R^{\text{U}}$ is biased, H is the matrix containing the input data of the graph, $H \in R^{\text{U}}$. From (5) (6) calculate $\text{H}_{\text{i}+1}$ as follows:

$$H_{\text{i}+1} = Relu(W_{\text{EP}} L_{\text{GMPEP}} H_{\text{i}} + b) \tag{7}$$

The GMPTS stage is the Temporal Graph Message Passing layer, which aggregates information based on time series relationships. Where K is a matrix containing the features of M timestamps, $K \in R^{\text{MxF}}$. From (4) calculate K_{i} as follows:

$$K_{\text{i}} = Relu(W_{\text{TS}} L_{\text{TS}} H_{\text{i}+2} + H_{\text{i}+2} W_{\text{U}} + b) \tag{8}$$

In formula (8), $b \in R^{\text{MxF}}$ is bias and $W_{\text{TS}} \in R^{\text{MxM}}$, $W_{\text{U}} \in R^{\text{MFxMF}}$ is the weight matrix that will change during training. F is the number of features of each graph G_{EP} after performing feature expansion. $\text{H}_{\text{i}+2}$ is calculated in formula (10) then reshaped to R^{MxF}.

C. Graph Convolutional

After each data update for the graph nodes through either GMPEP or GMPTS, we need to perform feature expansion or feature extraction for those nodes. That means there will be two different layers to perform feature expansion and feature extraction.

- The Timestamp Graph Convolution layer for graph-based feature expansion affecting prediction results:

$$W_{\text{GMPEP}} = RS_{\text{NxF}} \oplus RS_{\text{NxF}} \oplus \dots \oplus RS_{\text{NxF}} \tag{9}$$

  W_{GMPEP} is the direct sum of M random matrices. RS_{NxF} is a function that returns a random matrix of shape NxF. The output of this layer is the graph G_{TS}.

$$H_{\text{i}+2} = \tanh(H_{\text{i}+1} W_{\text{GMPEP}} + b_{\text{GMPEP}}) \tag{10}$$

- The Temporal Graph Convolution layer for feature extraction based on time series graph:

$$W_{\text{GMPTS}} = RS_{\text{FxZ}} \oplus RS_{\text{FxZ}} \oplus \dots \oplus RS_{\text{FxZ}} \tag{11}$$

  W_{GMPTS} is the direct sum of M random matrices. RS_{FxZ} is a function that returns a random matrix of shape FxZ and Z is the number of features of a node after performing feature extraction. The output of this layer is the feature vector from G_{TS}.

$$K_{\text{i}+1} = \tanh(K_{\text{i}} W_{\text{GMPTS}} + b_{\text{GMPTS}}) \tag{12}$$

In formulas (10) and (12), $b_{\text{GMPEP}} \in R^{\text{B}}$ and $b_{\text{GMPTS}} \in R^{\text{C}}$ are bias (B = MF; C = MZ), W_{GMPEP} and W_{GMPTS} are the weight matrix that will change during training. K_{i} is calculated in formula (8) then reshaped to R^{B}.

D. Artificial Neural Network

The output is obtained through a simple Artificial Neural Network (ANN) consisting of an input layer, a hidden layer, and an output layer. The input of ANN is K calculated in a formula (12).

$$\widehat{z} = Relu(KW + b) \quad W \in R^{\text{Ux64}} \quad b \in R^{32} \tag{13}$$

Dropout is an approach to regularization in Neural Networks which helps reduce interdependent learning amongst the neurons, because of that, we have to dropout 0.2 for $\widehat{z}$.

$$OUTPUT = Softmax(\widehat{z}W + b) \quad W \in R^{\text{64x6}} \quad b \in R^{6} \tag{14}$$

E. Summary Model

Figure 5 is a summary diagram of the graph-based deep learning model that we propose in this study.

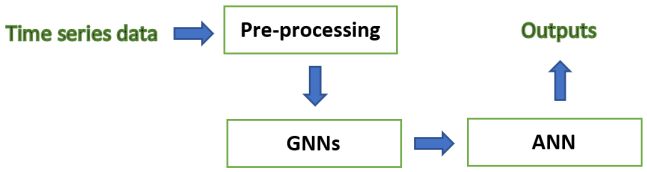

Fig. 5. The summary model.

The process of synthesizing information, expanding, and extracting features on the graph is summarized by the diagram in Figure 6. When encapsulation this diagram into a block with only input and output, the block is a GNNs block.

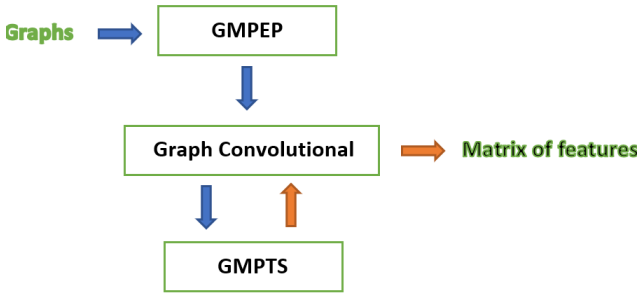

Fig. 6. The diagram of GNNs block.

The loss function used is Categorical Cross Entropy:

$$J(w) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y_i}) + (1 - y_i)\log(1 - \hat{y_i})] \quad (15)$$

The optimization function is Adam:

$$m_w^{(t+1)} \leftarrow \beta_1 m_w^{(t)} + (1 - \beta_1)\nabla_w L^{(t)}$$
$$v_w^{(t+1)} \leftarrow \beta_2 v_w^{(t)} + (1 - \beta_2)(\nabla_w L^{(t)})^2$$

$$\hat{m}_w = \frac{m_w^{(t+1)}}{1 - \beta_1^{t+1}}$$
$$\hat{v}_w = \frac{v_w^{(t+1)}}{1 - \beta_2^{t+1}}$$

$$w^{(t+1)} \leftarrow w^{(t)} - \eta \frac{\hat{m}_w}{\sqrt{\hat{v}_w} + \epsilon}$$

Our model is a lightweight Graph Neural Networks model to assess the impact of air quality on human health, which can be used for early warning of adverse human health impacts due to air pollution in areas such as factories, industrial parks, and space areas where many people are working. We converted it from TensorFlow to TensorFlow Lite for deployment at Gateways to help ensure real-time predictive results.

IV. Experimental Results

We have successfully built the proposed model with TensorFlow using Google Colab. The six labels used to relabel the dataset are Fresh, Polluted, Headaches, Pneumonia, Convulsions or Nausea, and Death. After re-labeling the dataset based on Table 1, we split the dataset into two at a ratio of 70% for the training set and 30% for the validation set. Time series data in IoT will often be missing values at some timestamp. These missing value errors usually occur due to an error in reading data from the sensor or an error in transmitting data from the node sensor to the Gateway. Therefore, we randomly assign a zero value to 10 percent of the data for each sample in our dataset to get a test set with missing samples value. The metrics related to the training set, validation set, and test set are shown in Table 2.

TABLE II
The metrics related to the training set, validation set, and test set

	Number of samples	Dropout element of samples
Training set	149478	0%
Validation set	64062	0%
Test set	213540	10%

The training process results in accuracy up to 98.15% on the training set and 92.86% on the validation set. Figures 7 and 8 show the variation in accuracy and loss level during training. We changed the learning rate every 25 epochs, so the training history graph has a steep slope as shown in Figures 7 and 8. The initial learning rate is 1x10 $^{-5}$, every 25 epochs we will divide it by 10.

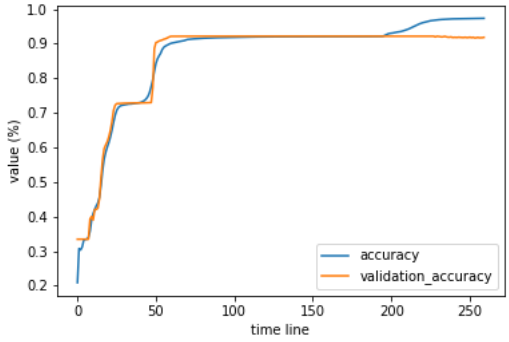

Fig. 7. The accuracy history.

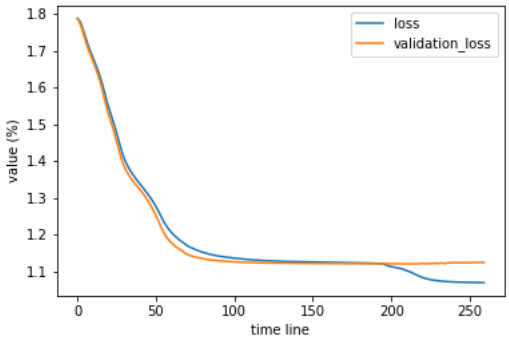

Fig. 8. The loss history.

The model evaluation process is carried out according to two scenarios as follows: In the first scenario, the data from the node sensor is fully transmitted at each timestamp and no errors occur during transmission. In that case, we reuse all

the samples in the dataset to perform the model evaluation. The second scenario is that the data of some timestamp is lost during transmission or due to an error of not reading the value from the sensor. The samples in the test set will be used to evaluate the model in case of missing data.

We used two other deep learning models to compare with our proposed model in terms of accuracy. Those two Models are LSTM and GRU because it is one of the most commonly used deep learning models for time series data. After training the LSTM model and the GRU model with the dataset used in this study, we obtained the results as shown in Table 3 for the case of full-valued samples.

TABLE III
THE ACCURACY OF OUR GNNS, GRU, AND LSTM MODEL IN THE CASE OF FULL-VALUED SAMPLES

Model	Training Accuracy	Validation Accuracy	Test Accuracy	Trainable Params	epochs
GNNs	98.15%	92.86%	97.35%	34,886	260
LSTM	94.43%	91.77%	94.56%	78,822	
GRU	96.61%	90.68%	95.42%	62,534	

We compared our model with LSTM model and GRU model to find that our model gives better prediction results in most cases of missing input data. The results of our evaluation when generating 20 different test sets from the training set are shown in Figure 9 and Table 4.

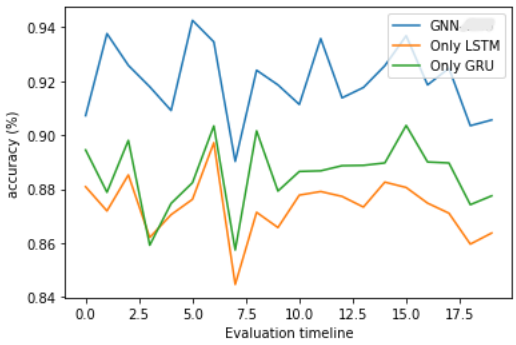

Fig. 9. History of model evaluation in case of missing values.

TABLE IV
THE ACCURACY OF OUR GNNS, GRU, AND LSTM MODEL IN CASE OF MISSING VALUES

Model	Max Accuracy	Min Accuracy	Average Accuracy	Median Accuracy
GNNs	94.25%	89.03%	92%	91.86%
LSTM	89.72%	84.47%	87.33%	87.41%
GRU	90.36%	85.75%	88.53%	88.77%

V. CONCLUSION AND FUTURE WORK

In this paper, we have introduced a new approach that uses Graph Neural Networks to build deep learning models for time series data. The model proposed in this study helps to predict the impacts of air pollution on human health. There are four time series containing information about CO, NO_2, O_3, and PM 2.5 concentrations in ambient air that are combined for prediction. We converted our model to TensorFlow Lite for deployment on the Gateway. The converted model is only 48 KByte in size and the accuracy is slightly reduced. In the future, we will research and deploy a solution to automate the retraining process and load the model proposed in this study to the Gateway via HTTP protocol. We hope our research will help create smart air monitoring stations in the trend of Edge computing and the Internet of Things.

ACKNOWLEDGEMENT

This research is funded by the Faculty of Computer Networks and Communications, University of Information Technology, Vietnam National University Ho Chi Minh City, Vietnam.

REFERENCES

[1] Abirami, S., Chitra, P. (2021). Regional air quality forecasting using spatiotemporal deep learning. Journal of Cleaner Production, 283, 125341.
[2] Navares, R., Aznarte, J. L. (2020). Predicting air quality with deep learning LSTM: Towards comprehensive models. Ecological Informatics, 55, 101019.
[3] Du, S., Li, T., Yang, Y., Horng, S. J. (2019). Deep air quality forecasting using hybrid deep learning framework. IEEE Transactions on Knowledge and Data Engineering, 33(6), 2412-2424.
[4] Soh, P. W., Chang, J. W., Huang, J. W. (2018). Adaptive deep learning-based air quality prediction model using the most relevant spatial-temporal relations. Ieee Access, 6, 38186-38199.
[5] Janarthanan, R., Partheeban, P., Somasundaram, K., Elamparithi, P. N. (2021). A deep learning approach for prediction of air quality index in a metropolitan city. Sustainable Cities and Society, 67, 102720.
[6] Sudharsan, B., Breslin, J. G., Tahir, M., Ali, M. I., Rana, O., Dustdar, S., Ranjan, R. (2022). Ota-tinyml: over the air deployment of tinyml models and execution on iot devices. IEEE Internet Computing, 26(3), 69-78.
[7] Huh, J. H., Seo, Y. S. (2019). Understanding edge computing: Engineering evolution with artificial intelligence. IEEE Access, 7, 164229-164245.
[8] Chen, M., Wei, Z., Huang, Z., Ding, B., Li, Y. (2020, November). Simple and deep graph convolutional networks. In International Conference on Machine Learning (pp. 1725-1735). PMLR.
[9] Kipf, T. N., Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.
[10] Han, J., Liu, H., Xiong, H., Yang, J. (2022). Semi-Supervised Air Quality Forecasting via Self-Supervised Hierarchical Graph Neural Network. IEEE Transactions on Knowledge and Data Engineering.
[11] Chen, L., Xu, J., Wu, B., Qian, Y., Du, Z., Li, Y., Zhang, Y. (2021). Group-aware graph neural network for nationwide city air quality forecasting. arXiv preprint arXiv:2108.12238.
[12] Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R. (2015). Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493.
[13] Manessi, F., Rozza, A., Manzo, M. (2020). Dynamic graph convolutional networks. Pattern Recognition, 97, 107000.
[14] Narwariya, J., Malhotra, P., TV, V., Vig, L., Shroff, G. (2020). Graph neural networks for leveraging industrial equipment structure: An application to remaining useful life estimation. arXiv preprint arXiv:2006.16556.
[15] Wang, Y., Zheng, J., Du, Y., Huang, C., Li, P. (2022). Traffic-GGNN: Predicting Traffic Flow via Attentional Spatial-Temporal Gated Graph Neural Networks. IEEE Transactions on Intelligent Transportation Systems.
[16] Chen, M., Zhang, J., Zhang, Z., Du, L., Hu, Q., Wang, S., Zhu, J. (2020). Inference for network structure and dynamics from time series data via graph neural network. arXiv preprint arXiv:2001.06576.