Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.036830
Article

Graph Construction Method for GNN-Based Multivariate Time-Series Forecasting

Wonyong Chung, Jaeuk Moon, Dongjun Kim and Eenjun Hwang*

School of Electrical Engineering, Korea University, Seoul, 02841, Korea
*Corresponding Author: Eenjun Hwang. Email: ehwang04@korea.ac.kr
Received: 13 October 2022; Accepted: 10 March 2023

Abstract: Multivariate time-series forecasting (MTSF) plays an important role in diverse real-world applications. To achieve better accuracy in MTSF, time-series patterns in each variable and interrelationship patterns between variables should be considered together. Recently, graph neural networks (GNNs) has gained much attention as they can learn both patterns using a graph. For accurate forecasting through GNN, a well-defined graph is required. However, existing GNNs have limitations in reflecting the spectral similarity and time delay between nodes, and consider all nodes with the same weight when constructing graph. In this paper, we propose a novel graph construction method that solves aforementioned limitations. We first calculate the Fourier transform-based spectral similarity and then update this similarity to reflect the time delay. Then, we weight each node according to the number of edge connections to get the final graph and utilize it to train the GNN model. Through experiments on various datasets, we demonstrated that the proposed method enhanced the performance of GNN-based MTSF models, and the proposed forecasting model achieve of up to 18.1% predictive performance improvement over the state-of-the-art model.

Keywords: Deep learning; graph neural network; multivariate time-series forecasting

1 Introduction

Accurate predictions on multivariate time-series data (i.e., data with numerous input variables in a single step) are crucial for many real-world applications. For example, the spread of an infectious disease can be mitigated by predictions based on the infection status of surrounding areas [1], congested driving routes can be avoided with predictions based on traffic flow in different locations [2], and investment direction can be determined based on the stock price trend of various companies [3].

With the recent rapid development of machine/deep learning models, many deep learning models for multivariate time-series forecasting (MTSF) such as Recurrent Neural Networks (RNNs) have been proposed. Unlike conventional statistical models, RNNs are powerful for capturing nonlinear characteristics of temporal features within univariate time-series data (i.e., intra-series patterns) [4].

However, for an accurate MTSF, besides the intra-series patterns, inter-relationships between variables (i.e., inter-series patterns) should be considered because each variable depends not only on its own historical values, but also on the historical values of other variables. Hence, a recent trend to address this is to use Graph Neural Networks (GNNs), which learn both the intra-series and the inter-series patterns in the format of a graph [5–8].

In GNNs, a graph consists of nodes (entities or regions) and edges (interrelationships between nodes) [9], so setting an appropriate graph is important for modeling multivariate time-series data. The graph can be set in two ways. The first is to define a graph in advance based on location identification or distance-based neighbor node selection algorithm, which is called a pre-defined graph [10,11]. This works well for datasets with fixed edges, such as roads or flights connecting cities. However, this structure is not suitable for cases where the relationship between nodes changes dynamically, such as the economic growth rate and the number of confirmed infections by country [8,12]. In these cases, the graph must be continuously changed during training to reflect the relationships between nodes in the input data that change dynamically over time.

Therefore, many existing GNNs construct dynamically changed graphs based on the similarity between two nodes, such as embedding vector similarity [12] or attention score similarity [13]. However, there are three limitations to this approach.

1. As time-series data is a combination of frequency functions, evaluation of their similarity must be made in the spectral domain where periodic patterns can be reflected fundamentally.
2. Setting up a graph that only considers similarity cannot effectively represent the time delay between the two time-series data. Although the flow of the two data is similar, it is difficult to capture the similarity in the case of time delay.
3. Existing GNN-based models consider all nodes equally weighted. There is no weighting process based on the number of connected nodes in the graph. However, hub nodes that are connected with many other nodes exchange more information than other nodes, so it is important to consider the influence of these hub nodes in constructing the graph.

To solve these limitations, in this paper, we propose a novel graph construction method for GNN-based MTSF models. The proposed method consists of three modules: (1) a Spectral Similarity Calculation Module to construct a graph based on the spectral features of the time-series data, (2) a Time Delay Reflection Module to refine the graph by considering the time delay between nodes, and (3) a Node Weighting Module to assign a weight to each node of the constructed graph according to the number of connections with other nodes. Using the proposed method, we obtain a graph matrix that contains the dynamic interrelationships between nodes that can enhance the forecasting performance of GNN-based MTSF models. In addition, we propose a forecasting model by replacing the graph construction part of a state-of-the-art GNN model by our graph construction method (hereafter, referred to proposed model). In extensive experiments on various public multivariate time series datasets, we show that the proposed graph construction method can improve the forecasting performance of existing GNN-based MTSF models, and the proposed prediction model can achieve up to 18.1% predictive performance improvement over the state-of-the-art model. Also, we visualize the constructed graph and show that our method can learn dynamic interrelationships between nodes. Finally, we investigate the effect of hyperparameters on our method and demonstrate that all components in our method are indispensable.

The contributions of this paper are as follows:

1. We propose a graph construction method for GNN-based MTSF that reflects the relationship between nodes considering the similarity in a spectral domain, time delay, and hub node.
2. We compare various MTSF models and demonstrate that our model outperforms other existing MTSF models in various public datasets.
3. We prove that our graph construction method can enhance the forecasting accuracy of existing GNN-based MTSF models.
4. We show that nodes with similar data patterns have high edge weight in a generated graph.

The paper is organized as follows. In Section 2, we describe the related works of the methodologies used in this paper and the time-series forecasting field. In Section 3, we introduce the problem formulation of this research. Section 4 describes the details of the model used in this paper. Experimental settings, datasets, and evaluation metrics are covered in Section 5. In Section 6, we introduce the experimental results. Finally, we summarize our work and discuss future studies in Section 7.

2 Related Works

So far, diverse conventional statistical forecasting models have been used to forecast univariate time-series. For instance, Perone et al. [14] proposed the Auto Regressive Integrated Moving Average (ARIMA) model for forecasting future influenza outbreaks. They combined Auto Regression (AR) and Moving Average (MA) to show better forecasting performance compared to the existing statistical models. Furthermore, García-Ascanio et al. [15] proposed Vector Auto Regression (VAR) model, an extended model of the AR model, to capture the linear interdependencies among electric power time-series data. VAR achieved higher forecasting performance than interval Multi-Layer Perceptron (iMLP), a simple form of the neural network model. Senanayake et al. [16] proposed a non-parametric model based on Gaussian Process (GP) regression. GP model can be applied to multivariate time-series data by modeling multiple variables of a function using a Gaussian distribution. Accordingly, the GP regression model is applied to capture the spatial and spatial and temporal dependencies in influenza-like illness data. The model outperformed the predictive performance of the ARIMA model. However, these statistical forecasting models do not work well for rapidly changing multivariate time-series data [17]. This is because nonlinear relationships are not suitable for describing large datasets and focus on estimating mean values rather than data distributions.

Recently, deep learning models are demonstrating outstanding performance on MTSF. For instancec, Volkova et al. [18] proposed a long short-term memory (LSTM)-based deep learning model specialized for processing sequential time-series data [19–21] to forecast multivariate time-series data. They used topics, word n-grams, and stylistic patterns that were extracted from social media data. Zhu et al. [22] also proposed a LSTM-based deep neural model for influenza forecasting. The model had multi-channel LSTM layers that drew multiple pieces of information from multiple different inputs, including climate, pharmacy, and symptom data. Likewise, Tang et al. [23] proposed a LSTM-based deep learning model to predict stock prices. They denoised the data and passed the data through a model constructed by stacking LSTM in multiple layers. They improved the forecasting performance further by denoising. Khashei et al. [24] proposed a multi-layer perceptron (MLP)-based stock price forecasting model. They forecasted stock price through a hybrid model by combining ARIMA with MLP, and this model showed good performance compared to the existing statistical models [25].

Recently, GNN has been widely used for MTSF. The graph of GNN consists of a set of nodes representing each input variable in a multivariate time series and a set of edges representing the

relationship between the input variables [26]. In GNN, node information is propagated through the edge, and prediction of a node is performed by considering information from neighboring nodes. GNNs are better suited for multivariate prediction because they can effectively reflect information from other nodes. GNN receives the graph as an input and learns along the relationship between nodes. Hence, they showed good performance in various areas.

Bloemheuvel et al. [27] proposed a GNN model that predicts peak ground acceleration (PGA), peak ground velocity (PGV), and spectral acceleration (SA) from seismic wave data from various locations in Italy. The features of each seismic wave were extracted through 1d convolutional neural network (CNN) layer, and each node information was exchanged through a Graph Convolutional Network (GCN). In addition, the authors set the adjacency matrix to be inversely proportional to the distance between nodes. The proposed model showed superior performance compared to the comparative models.

For instance, Deng et al. [28] forecasted the number of confirmed cases of infectious diseases using a distance-based pre-defined graph. In this model, the input time-series data is processed in parallel by the RNN and CNN modules. The RNN module obtains a hidden state with embedded temporal features for the input, and the model calculates an interrelationship attention score based on the hidden state value and the pre-defined graph. Meanwhile, the CNN module extracts temporal features by performing temporal convolution to the input data. Then, graph message passing is established based on interrelationship attention scores and temporal features obtained from the two modules. After that, the model predicts the node information through the MLP layer. The model outperformed statistical models and other deep learning models. On the other hand, Yu et al. [29] used both temporal convolution and graph convolution for traffic flow forecasting. Temporal convolution captures temporal features, and graph convolution captures inter-series patterns. The core of the model, the ST-Conv block, was constructed in the form of a graph convolution sandwiched between two temporal convolution layers. This "sandwich" structure was created to achieve a bottleneck strategy for scale compression and feature squeezing with graph convolution. The model predicts traffic in each region through the input traffic data and a pre-defined road graph based on the distance between each area. Table 1 summarizes all the models mentioned so far.

Table 1: A summary of relevant works

Author (Year)	Forecasting target	Forecasting method	Description
Perone et al. [14] (2020)	Influenza	ARIMA	This approach showed better forecasting performance compared to the existing statistics-based models

(Continued)

Table 1: Continued

Author (Year)	Forecasting target	Forecasting method	Description
Garcia et al. [15] (2009)	Electric power demand	VAR	This approach showed better forecasting performance compared to iMLP
Senanayake et al. [16] (2016)	Influenza	GP	This approach captured the space-time dependencies by combining different kernels
Volcova et al. [18] (2017)	Influenza-like illness	LSTM	This approach used various linguistic signals extracted from social media data to forecast influenza-like illness dynamics
Zhu et al. [22] (2019)	Influenza	LSTM	This approach utilized multi-channel LSTM layers that can draw multiple information from multiple inputs and added attention mechanism to improve forecasting accuracy
Tang et al. [23] (2021)	Stock price	LSTM	This approach showed that using data denoised by the wavelet transform (WT) and singular spectrum analysis (SSA) can forecast more accurately

(Continued)

Table 1: Continued

Author (Year)	Forecasting target	Forecasting method	Description
Khashei et al. [24] (2019)	Stock price	ARIMA, MLP	This approach outperformed each component (ARIMA, MLP) and also showed better performance than ARIMA-MLP model
Deng et al. [28] (2020)	Infectious diseases	GNN	This approach outperformed statistical-based models and other deep learning models by using distance-based pre-defined graph with RNN and CNN module
Yu et al. [29] (2017)	Traffic flow	GNN	This approach effectively captured comprehensive spatio-temporal correlations by using both temporal convolution and graph convolution

However, graphs of existing GNN models have limitations in considering spectral similarity, time delay between nodes, and importance of hub nodes. In this paper, we propose a novel graph construction method that overcomes these limitations and construct a GNN-based MTSF model using the constructed graph. To consider spectral similarity, we utilize Fast Fourier transform (FFT) and to reflect the time delay, we identify the progress order of the two time-series data and reflect it in the graph. In addition, the weight of the hub node is increased by assigning a weight according to the number of connections each node has.

3 Problem Formulation

In this section, we first define a multivariate time-series forecasting problem based on a graph G. As described in Eq. (1), $X_{seq} = \{x_t^i\} \in \mathbb{R}^{N \times T}$ stands for the multivariate time-series input, where N means the number of time-series (nodes), and T means the sequence length of input time-series data. Here, a node can be a region or city in influenza forecasting, a company in stock price forecasting, and a building or factory in electrical load forecasting. We denote $x_t^i \in \mathbb{R}$ as time-series data in node i at

time t, and indicate $Xt = [x_t^1, x_t^2, x_t^3 \ldots x_t^N] \in \mathbb{R}^{N\times 1}$ and $X^i = [x_{t-T+1}^i, x_{t-T+2}^i, x_{t-T+3}^i \ldots x_t^i] \in \mathbb{R}^{1\times T}$ as a set of time-series data in each node at time t and a set of node i during the past T time stamps, respectively. Our goal is to predict $Y = [x_{t+h}^1, x_{t+h}^2, x_{t+h}^3 \ldots x_{t+h}^N]$, which is the value obtained after h time stamps. We denote by G the matrix representing connection information and interrelationships between multiple nodes. We also denote the edge weight of node i and node j in matrix G as Gij and the total connection strength of node i as Ci. Table 2 shows the key notations and descriptions. The undefined notations in this section are described in detail later.

$$X_{seq} = \begin{bmatrix} x_{t-T+1}^1 & \cdots & x_t^1 \\ \vdots & \ddots & \vdots \\ x_{t-T+1}^N & \cdots & x_t^N \end{bmatrix} \tag{1}$$

Table 2: Notations and descriptions

Notation	Description
N	Number of nodes
T	Input sequence length
$Xseq$	Input data
h	Horizon/lead time of forecasting
Y	Actual value
$\hat{Y}$	Forecasted value
Gij	Edge weight of graph adjacency matrix that connects node i and node j
Ci	Total connection strength of the node i

4 Proposed Method

This section describes the overall structure of the proposed model. Fig. 1 shows the overall structure of our model.

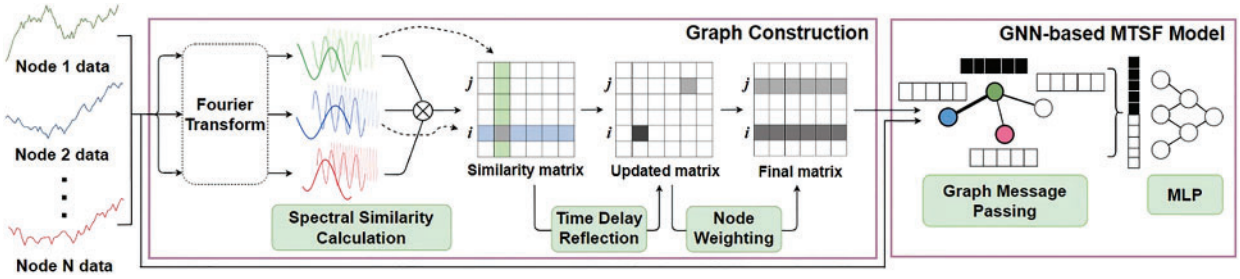


Figure 1: Overall architecture of the proposed method

4.1 Spectral Similarity Calculation Module

To determine the similarity of two time-series data, we compose the data into temporal frequencies and compare the intensity (amplitude) of each frequency to determine the similarity. FFT is a popular

frequency decomposition technique for time-series data [30] due to its outstanding capability of learning latent representations of multiple time-series data in the spectral domain.

FFT decomposes a sequence data into multiple frequency components ranging from 1 to $\lfloor (T-1)/2 \rfloor$, about half of the input time length. Low-frequency functions represent the overall trend of the time-series data, while high-frequency functions represent the noise in the data [31]. Therefore, low-frequency functions are suitable for similarity comparison of time-series data. For this reason, we calculate the similarity of time-series in this layer using only low-frequency functions. We use the lower 20% of the frequency range, so the frequency range becomes $1 \sim \lfloor \lfloor (T-1)/2 \rfloor * 0.2 \rfloor$.

We calculate the similarity of the time-series data of two nodes using the decomposed frequency functions based on two factors: (1) The similarity of two time-series data is high when they have the same phase for the same frequency, and (2) the similarity is higher when the amplitudes of the two time series data are similar. To satisfy these factors, the similarity is calculated by dot-producting the amplitudes of low-frequency periodic functions of the two time-series data. Here, the similarity G_{ij} of the time-series data of nodes i and j is calculated by Eqs. (2) and (3). In these equations, I_k and J_k represent the amplitude of k-th frequency function of X_i and X_j, which are the input time-series data of nodes i and j, respectively. In addition, t represents $\lfloor \lfloor (T-1)/2 \rfloor * 0.2 \rfloor$, which is the highest frequency among the frequencies to be calculated.

After calculating the similarity of the time-series data between all nodes by Eq. (2), we can construct an $N \times N$ similarity matrix composed of G_{ij} between nodes. The matrix values are arranged in normal distribution in order to filter values with high similarity. Similarity values with standard score greater than 0 are used, and values with standard score less than 0 are set to 0 to eliminate the edges.

$$FFT\,(X_i) \approx \sum_{k=1}^{t} I_k,\ FFT(X_j) \approx \sum_{k=1}^{t} J_k \tag{2}$$

$$G_{ij} = FFT(X_i) \cdot FFT(X_j) \tag{3}$$

4.2 Time Delay Reflection Module

Although the Spectral Similarity Calculation Module constructs a similarity matrix in the spectral domain, various time-series data in the real world have a time delay. For example, in the case of infectious diseases, as the infectious disease spreads from the affected area, the number of confirmed cases in other areas (usually adjacent areas) also increases over time [32]. In this case, the current time series data has a great influence on subsequent time series data. In addition, data before h timestamp is most important when forecasting after h timestamp. Such time delay property of the time series data is handled by the Time Delay Reflection Module, as described in Algorithm 1. In particular, the algorithm considers two factors: One is the time delay and the other is the horizon difference. To account for time delay, we set one of the two time-series data as fixed data and the other as moving data. Then, while moving the starting point of the moving data, we find the point where the distance between the two time-series data is minimal. Here, the distance is calculated using Dynamic Time Warning (DTW) [33], an algorithm that measures the similarity between two time-series data. DTW is easier to eliminate noise than Euclidean distance because it uses not only the same time data but also the surrounding time data as a comparison target. If the DTW distance is the minimum at the point where the starting point of the moving data is smaller than the starting point of the fixed data, we set the moving data as the preceding data. In addition, we consider horizon time step h when constructing the graph. To do this, we set G_{ij} large when the temporal difference (time delay) between the data at

preceding node i and the data at following node j is exactly h, and set smaller when the temporal difference is far from h. Accordingly, the edge weight is set to be inversely proportional to the absolute value of the difference between horizon time step and time delay. If the time delay is equal to the horizon time step, then the weight G_{ij} of the edge connecting two nodes i and j is set to 1. The graph edge weight from the following node to the preceding node is set to be inversely proportional to the time delay.

Algorithm 1: Reflection time delay

Input: ***Xseq***, G, input length T, moving length m, horizon h
Output: updated G
for each node i of *Xseq* **do**
  for each neighbor j of node i **do**
    for *each starting point of Xj* **do**
      Target data ← *Xj* [*starting point: starting point* + T]
      Distance ← DTW(Target data, *Xi*)
      if *Distance* < *min Distance* **do**
        min Distance, min starting point ← *Distance, starting point*
        starting point ← *starting point* + m
      if *min starting point* < *fixed data starting point* **do**
        preceding data, following data ← *Xj*, *Xi*
      else do
        preceding data, following data ← *Xi*, *Xj*
      time delay ← |*Xi starting point* – *min starting point*|
      Gij ← *Gij*/|time delay – h + 1|
      Gji ← *Gji*/time delay

4.3 Node Weighting Module

In dynamically changing data such as population movement or economic flow, the influence of hub nodes with many connected edges such as amount of movement is much greater than that of other ordinary nodes. For example, hub airports with many transfers are crowded with transfer passengers [34]. Therefore, it is important to reflect hub node information to the graph for the GNN model to learn dynamically changing data. To do that, we assigned a higher weight to nodes with many connections with other nodes.

Earlier, we constructed a graph G that defined the edge weights of the graph based on the similarity between nodes. The goal of this module is to reflect the weights according to the influence of nodes on the graph using Eq. (4). In the equation C_i represents the total amount of influence node i has and f represents a transformation function with the similarity value calculated in the earler step. Then, we update the original graph G_i by multiplying $f(C_i)$ to obtain the updated G_i. We multiplied edges from a node with a lower weight to another node by a lower weight. Conversely, we multiplied the edges from the node with a higher weight to another node by a higher weight.

$$G_i^{updated} = f(C_i) \cdot G_i^{original} \tag{4}$$

If we denote the sum of the influences node i has on its neighboring nodes as C_i, $\mathbb{C} = \{C_1, C_2, C_3, \ldots C_n\}$. Eq. (5) describes how to calculate C_i. Here, N_i denotes the set of neighboring nodes of node i. If we use the original similarity value in this step, the influence of nodes with few connections

can be completely removed. Hence, we applied a logarithmic function to the similarity so that small similarity values are not removed and have only minimal impact. Also, we set all C_is to be positive based on the largest and smallest connection weights, *Cmax* and *Cmin*, respectively.

$$C_i = \sum_{j \in N_i} \left(\ln\left(G_{ij} + 1\right) + 1\right) \tag{5}$$

$$f\left(C_i\right) = \left(\frac{1}{1 + e^{-a*(C_i - b)}}\right) * c + d \tag{6}$$

We generate a node weighting function f via C_i, and multiply f by *Goriginal*. In this way, we update *Goriginal* by applying node weight. We define the function f as Eq. (6). This function determines the weight for each node, such that hub nodes are given large weights. We describe the explanation of parameter $a \sim d$ in Table 3 that used in Eq. (6) to eliminate differences in expressions according to data characteristics. The node weight was multiplied by the row corresponding to each node of the $G^{original}$. We repeat this process for all nodes and obtain the $G^{updated}$.

Table 3: Parameters of the node weight function

$a = \left(\frac{10}{Cmax - Cmin}\right)$	This keeps the shape of f. The numerator value makes a function suitable for the hub node weighting task.
$b = \frac{Cmin + Cmax}{2}$	This moves f along the C_i axis so that the center of the f is located at average value of *Cmin* and *Cmax*.
$c = 1 - ln\left(1.35 * \left(1 + \frac{Cmin}{Cmax}\right)\right)$	This limits the range of f result values.
$d = ln\left(1.35 * \left(1 + \frac{Cmin}{Cmax}\right)\right)$	This moves $\boldsymbol{f}$ to the node weight axis so that the result value of *Cmax* converges to 1. If the numerical difference between *Cmin* and *Cmax* is small, $\boldsymbol{f}$ (***Cmin***) $\approx \ln$ (**2.70**) $\approx$ **1.00**.

4.4 GNN-Based MTSF Model

Using the graph constructed by the proposed method, we construct a GNN-based MTSF model (referred to "the proposed model" in the following sections) based on Cola-GNN [28]. This model was initially proposed for predicting infectious disease occurrences and used a pre-defined graph that considered only geological locations. We modified this pre-defined graph structure using the proposed graph construction method so that the graph can reflect dynamically changing interrelationships between the variables (nodes). Even though we apply our graph to Cola-GNN for forecasting, our graph structure can be applied to other GNN-based models (described in Section 6. D).

Now, we describe the process of learning the inter-relationship and the intra-relationship of multivariate time-series using the graph we proposed. Cola-GNN consists of three main components: (1) graph that considers inter-relationships between nodes, (2) node feature that represents the intra-relationship in each node, and (3) graph message passing that uses graph and node feature for learning information from neighboring nodes. We obtain the first component by attention score based on the last hidden state of the RNN. Then, we use this attention matrix along with the graph G generated by the graph construction layer. Next, we obtain the features of each node by applying dilated convolution

to each node of *Xseq*. Finally, we perform graph message passing and forecast future time-series with neighbor nodes data.

5 Experimental Settings

To evaluate the effectiveness of the proposed model, we performed extensive experiments using six public datasets. Before we present the experimental results in detail, we first describe the datasets we used, the evaluation metrics, and the experimental setup for the comparison models.

5.1 Datasets

We used six real-world datasets in the experiments: three influenza-like illness (ILI) data, exchange rate data, stock price data, and electrical load data. In the experiments, we normalized all datasets to a range between 0 and 1 for each node data using min-max normalization. Also, we split the data into training, validation, and test sets in chronological order at ratios of 60%, 20%, and 20%.

- **ILI US-States:** This contains weekly ILI for each state in the US from 2010 to 2017. From the original dataset, we utilized weekly data on new cases in each state in the United States from the Centers for Disease Control and Prevention (CDC) [35].
- **ILI US-Regions:** This dataset includes weekly ILI for ten regions of the US from 2002 to 2017. The ten regions were defined as geographically close states by the Department of Health and Human Services (HHS). Data collection proceeded in the same way as US-States.
- **ILI Japan-Prefectures:** This contains weekly ILI for each prefecture in Japan from 2012 to 2019. We collected weekly data on new cases in each prefecture in Japan from the Infectious Diseases Weekly Report (IDWR) [36].
- **Exchange rate:** This contains daily exchange rate data for 8 countries (Australia, British, Canada, Switzerland, China, Japan, New Zealand, and Singapore) from 1990 to 2016 [37].
- **US stock market price:** We collected daily stock prices of 50 major companies listed on the US stock market from 2007 to 2016 [38] and composed this dataset.
- **GEFCOM 2012:** This dataset includes the hourly electrical load data of a US utility from January 1, 2005 to December 31, 2008 for 20 zones, which were used in the load forecasting track of the Global Energy Forecasting Competition 2012 (GEFCOM 2012) hosted on Kaggle [39]. Among 20 zones, we utilized data from zone 1 to zone 11 since the other nine zones have different variable conditions.

5.2 Evaluation Metrics

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \tag{7}$$

$$PCC = \frac{\sum_{n=1}^{N}(\overline{y_i} - y_i)(\overline{\hat{y}_i} - \hat{y}_i)}{\sqrt{\sum_{n=1}^{N}(\overline{y_i} - y_i)^2}\sqrt{\sum_{n=1}^{N}(\overline{\hat{y}_i} - \hat{y}_i)^2}} \tag{8}$$

For performance comparison, we used two popular metrics for MTSF [28]: Root Mean Squared Error (RMSE) and Pearson Correlation Coefficient (PCC). The RMSE represents the difference between forecasted values and actual values. PCC is a numerical value in range $(-1, 1)$ that quantifies the linear correlation between forecasted and actual values. Since PCC is calculated as a value close to 1 when the predicted value is similar to the actual value, it is useful not only for comparison between different models but also for comparison between different datasets. On the other hand, RMSE is

represented by a lower value as the model's forecasting performance improves, and is not suitable for comparing the predictive performance of different datasets due to data scale dependence. We selected these two metrics because PCC is easy to compare since all results are calculated in the same range, and RMSE is effective in determining how far the predicted value differs from the actual value.

We denote the forecasted values, actual values, and number of data samples as $\hat{y}_i$, y_i, and N, respectively. We also denote the mean value of y_i as $\overline{y_i}$. RMSE and PCC are calculated using Eq. (7) [40] and (8) [41,42], respectively.

5.3 Experimental Settings

For performance comparison, we considered eight different MTSF models: AutoRegressive (AR), AutoRegressive and MovingAverage (ARMA), MLP, RNN, LSTM, LSTNet, ST-GCN, and Cola-GNN. We set epoch, batch size, learning rate, input sequence length T, and dropout probability to 500, 100, 0.001, 30, and 0.3 respectively. And we measure the forecasting performance by averaging the results of repeating the experiment 5 times. Depending on the setting of the [28], we evaluate our approach in short-term (horizon $h = 2, 3$) and long-term (horizon $h = 5, 10, 15$) settings. We ignore the case of h equals 1 because symptom monitoring data is usually delayed by at least one-time step. We measured the forecasting performance by averaging the results of repeating the forecasting 10 times. All experiments were done in the Python environment, and all models were implemented using PyTorch [43] library. The model-specific hyperparameters are empirically determined.

6 Experiments and Discussion

In this section, we first present the experimental results and analyze the graph we constructed. We then report on the ablation studies performed.

6.1 Forecasting Performance

Tables 4–9 show the results of comparative experiments using six datasets. In the tables, the bold values indicate the highest values of RMSE and PCC in each horizon. Tables 4–6 present the results of comparative experiments on the ILI datasets. Table 4 shows that our model outperforms other existing models for all cases on the ILI US-Regions dataset. In particular, since the graph generated by the proposed model better reflects the characteristics of dynamically changing time series data than the predefined graph of Cola-GNN, the proposed model shows better predictive performance.

Table 4: Comparison of forecasting performance on the ILI US-Regions dataset

Horizon	2		3		5		10		15	
Metric	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC
AR	570	0.927	757	0.878	997	0.792	1330	0.612	1404	0.527
ARMA	560	0.927	742	0.876	989	0.792	1322	0.614	1400	0.520
MLP	524	0.931	701	0.869	974	0.803	1312	0.608	1409	0.531
RNN	513	0.940	689	0.895	896	0.821	1328	0.587	1434	0.499
LSTM	507	0.943	688	0.895	975	0.812	1351	0.586	1477	0.488

(Continued)

Table 4: Continued

Horizon	2		3		5		10		15	
Metric	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC
LSTNet-skip	554	0.935	801	0.868	998	0.746	1157	0.609	1231	0.533
ST-GCN	697	0.879	807	0.840	1038	0.741	1290	0.644	1286	0.619
Cola-GNN	480	0.940	636	0.909	855	0.835	1134	0.717	1203	0.639
Ours	**478**	**0.946**	**634**	**0.922**	**726**	**0.890**	**932**	**0.815**	**1101**	**0.755**

Table 5: Comparison of forecasting performance on the ILI US-States dataset

Horizon	2		3		5		10		15	
Metric	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC
AR	161	0.940	204	0.909	251	0.863	306	0.773	327	0.723
ARMA	161	0.939	200	0.909	250	0.862	306	0.773	326	0.725
MLP	159	0.938	200	0.917	236	0.891	294	0.791	320	0.739
RNN	149	0.948	181	0.922	217	0.886	274	0.821	315	0.758
LSTM	150	0.948	180	0.922	213	0.889	276	0.820	307	0.771
LSTNet-skip	199	0.913	249	0.850	299	0.759	292	0.760	292	0.802
ST-GCN	189	0.907	209	0.778	256	0.823	289	0.769	292	0.774
Cola-GNN	**136**	**0.955**	167	0.933	205	0.887	241	0.822	237	0.856
Ours	141	0.947	**151**	**0.949**	**202**	**0.897**	**229**	**0.841**	**221**	**0.880**

Table 6: Comparison of forecasting performance on the ILI Japan-Prefectures dataset

Horizon	2		3		5		10		15	
Metric	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC
AR	1377	0.752	1705	0.579	2013	0.31	2107	0.238	2042	0.483
ARMA	1371	0.754	1703	0.579	2013	0.31	2105	0.238	2041	0.486
MLP	1190	0.817	1437	0.716	1712	0.46	1905	0.497	1827	0.604
RNN	1001	0.892	1259	0.833	1376	0.821	1696	0.616	1629	0.709
LSTM	1052	0.896	1246	0.873	1335	0.853	1622	0.681	1649	0.695
LSTNet-skip	1133	0.846	1459	0.728	1883	0.432	1811	0.518	1884	0.515
ST-GCN	996	0.902	1115	0.88	1129	0.872	1541	0.735	1527	0.773
Cola-GNN	929	0.915	1051	0.901	1117	0.89	**1372**	0.810	1475	0.753
Ours	**918**	**0.922**	**1030**	**0.914**	**1088**	**0.898**	1398	**0.813**	**1446**	**0.758**

Table 7: Comparison of forecasting performance on the Exchange rate dataset

Horizon	2		3		5		10		15	
Metric	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC
AR	0.00689	0.921	0.00814	0.887	0.0104	0.793	0.0143	0.703	0.0169	0.554
ARMA	0.00672	0.921	0.00829	0.882	0.0106	0.793	0.0142	0.694	0.0171	0.553
MLP	0.00656	0.927	0.00764	0.876	0.0109	0.789	0.014	0.716	0.0167	0.573
RNN	0.00627	0.92	0.00798	0.869	0.00988	0.799	0.013	0.7	0.0147	0.56
LSTM	0.00638	0.921	0.00701	0.868	0.00987	0.798	0.0129	0.701	0.0158	0.56
LSTNet-skip	0.00594	0.933	0.00785	0.897	0.0100	0.802	0.0121	0.722	0.0164	0.594
ST-GCN	**0.00572**	**0.949**	0.00729	0.894	0.0112	0.781	0.0122	0.741	0.0159	0.589
Cola-GNN	0.00624	0.940	0.00708	0.917	0.0094	0.812	0.0119	0.743	0.016	0.641
Ours	0.00587	0.947	**0.00700**	**0.921**	**0.0092**	**0.835**	**0.0118**	**0.751**	**0.0154**	**0.668**

Table 8: Comparison of forecasting performance on the US stock market price dataset

Horizon	2		3		5		10		15	
Metric	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC
AR	18.55	0.937	21.76	0.907	25.82	0.856	32.29	0.774	36.96	0.698
ARMA	18.27	0.939	20.98	0.918	25.94	0.859	32.58	0.781	36.73	0.690
MLP	17.97	0.943	19.07	0.906	25.60	0.848	32.08	0.780	34.78	0.657
RNN	17.49	0.930	18.99	0.920	24.87	0.867	30.52	0.771	34.51	0.672
LSTM	17.68	0.927	18.99	0.911	24.87	0.855	30.59	0.768	34.72	0.681
LSTNet-skip	17.12	0.943	18.97	0.927	25.06	0.862	31.06	0.783	34.16	0.669
ST-GCN	**16.71**	**0.949**	19.01	0.933	23.97	0.870	28.97	0.790	34.00	0.710
Cola-GNN	16.73	0.950	18.65	0.933	24.14	0.891	28.61	**0.807**	33.16	0.714
Ours	16.89	0.939	**18.44**	**0.950**	**23.43**	**0.910**	**28.04**	0.793	**32.79**	**0.721**

Table 9: Comparison of forecasting performance on the GEFCOM2012 dataset

Horizon	2		3		5		10		15	
Metric	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC
AR	6248	0.922	6814	0.887	7517	0.849	9914	0.765	11713	0.673
ARMA	6198	0.939	6808	0.888	7496	0.851	9871	0.781	11001	0.661
MLP	6112	0.943	6772	0.891	7444	0.869	9889	0.775	10988	0.680
RNN	6090	0.938	6669	0.912	7381	0.881	9718	0.783	10640	0.716

(Continued)

Table 9: Continued

Horizon	2		3		5		10		15	
Metric	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC
LSTM	6098	0.939	6663	0.914	7294	0.887	9634	0.802	11211	0.724
LSTNet-skip	6034	0.943	6571	0.920	7130	0.899	9313	0.807	9930	0.732
ST-GCN	5999	0.949	6498	0.934	6992	0.886	9226	0.812	9917	0.739
Cola-GNN	5910	0.959	6447	0.931	6924	0.890	9236	0.819	9900	0.752
Ours	**5872**	**0.963**	**6322**	**0.944**	**6814**	**0.919**	**9047**	**0.838**	**9793**	**0.760**

On the other hand, Table 5 for the ILI US-States dataset shows that the proposed model achieves the best accuracy, except for h = 2 and 5, where Cola-GNN had the best predictive performance. Even in these cases, our model achieves the second best performance, and the difference from the best case is negligible compared to the other models. This is because ILI US-States dataset has a more fine-grained pre-defined graph (51 nodes) compared to the ILI US-Regions dataset (10 nodes). As a result, in some cases, the Cola-GNN model performed better on the ILI US-tates dataset.

In Table 6 for the ILI Japan-Prefectures dataset, our model outperforms other models in all cases except when $h = 10$ in RMSE. However, our model achieves the best performance in PCC.

Table 7 presents the comparison results for the Exchange rate dataset, similar to the results for the ILI US-States dataset. It should be noted that ST-GCN performs well when h = 2 in this experiment. This is because unpredictable factors such as the international situation are more decisive than trends in time series data for short-term forecasting in the economic field [44]. Therefore, the effectiveness of our proposed graph construction may be reduced for short-term forecasting. Nevertheless, our model shows the best performance in other cases.

Table 8 presents the results for the US stock market price dataset, similar to the results of the Exchange rate dataset. Our model shows slightly lower performance for US stock market dataset compared to the Exchange rate dataset. This is because the US stock market price data set has less influence on each other than the Exchange rate dataset.

Lastly, Table 9 (GEFCOM 2012 dataset) shows that our model achieved the best accuracy for all cases. Also, Cola-GNN ranked second in most cases. Statistics-based forecasting models showed relatively low performance than GNN-based forecasting models. In addition, our model showed outstanding performance even in multi-regional electrical load forecasting.

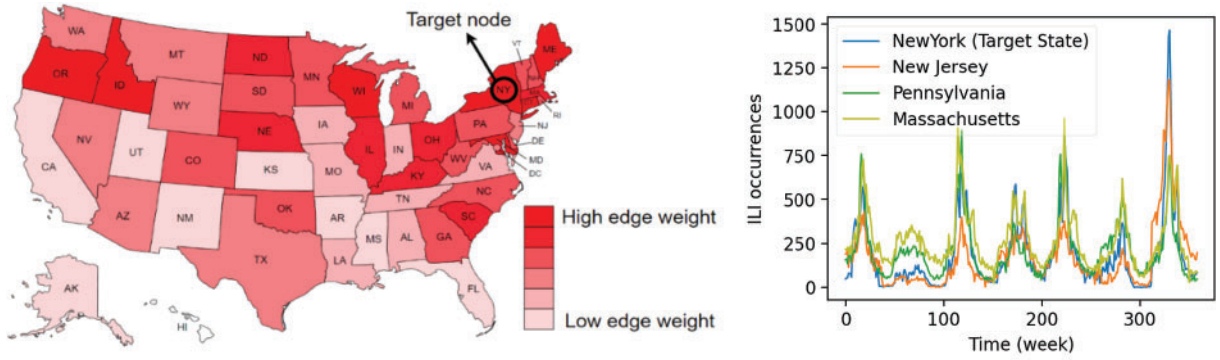
To sum up, our model showed the best performance in most cases (21 out of 25 for RMSE and 20 out of 25 for PCC). Also, basic deep learning models (MLP, RNN, LSTM) exhibited better accuracy than statistical models (AR, ARMA). However, GNNs (ST-GCN, Cola-GNN, our model) showed higher performance than basic deep learning models. Particularly, our model showed outstanding performance, in the long-term forecasting that has a large difference between the test period and training period, compared to short-term forecasting.

6.2 Graph Analysis

In this section, we present analysis results through comparison with the input data to provide a more intuitive understanding of the constructed graph.

Fig. 2a exhibits a heat map that visually represents the edge weights of New York (NY) state and other states when forecasting the future ILI occurrence in NY when $h = 10$, using the ILI US-State data. Fig. 2b shows the actual ILI for the states with high similarity to NY, such as New Jersey (NJ), Massachusetts (MA), and Pennsylvania (PA). In Fig. 2a, the red color of the state indicates a higher edge weight, and white color indicates a lower edge weight. Since infectious disease diffusion is greatly affected by geographical factors (e.g., distance from the origin), nearby states of NY show higher edge weight. For instance, NJ and MA, which are close to NY, have high edge weights. Conversely, states that are far away from NY, such as Florida (FL), have low edge weights. From this visualization, we can observe that the graph generated by the proposed method reflects the relationship between nodes well. In Fig. 2b, we can also observe that the occurrences trend and peak timing of ILI are almost identifical in NY and the states with high edge weights.

As a result, our model has good interpretability because it shows good performance in capturing interrelationships and can visually represent the results.



(a) Heat map of edge weights of New York (NY) and other states

(b) Comparison of ILI occurrences in states with high edges to NY

Figure 2: Visualization of similar nodes when forecasting NY in ILI US-States data

6.3 Ablation Study

To analyze the effect of each component in our model, we conducted some ablation tests using ILI US-Regions dataset. We compare the performance of the following three methods using the ILI US-Regions dataset when $h = 15$.

- **w/o *Spectral Similarity Calculation Module*:** Instead of the graph generated by the Spectral Similarity Calculation Module, we just use the pre-defined graph of Cola-GNN.
- **w/o *Time Delay Reflection Module*:** Here, we do not use the Time Delay Reflection Module. That is, we just pass the outputs of the Spectral Similarity Calculation Module to the next Node Weighting Module directly.
- **w/o *Node Weighting Module*:** Here, we do not perform the node weighting. That is, we pass the generated graph to the Forecasting part without refining the node weights depending on the connectivity.

Table 10 shows the RMSE and PCC of the three methods. The results show that all components of our model are contributing to improved performance. First, the Spectral Similarity Calculation Module significantly improves the performance because it creates a graph based on spectral similarity,

enabling efficient information flow between nodes. Furthermore, the second method demonstrates that incorporating time order and time delay into the graph help improve performance. Lastly, the Node Weighting Module shows the advantage of reflecting the importance of the hub nodes in the graph. That is, the information passing through the hub nodes has a lot of influence on the performance.

Table 10: Forecasting results for ablation study using ILI US-Regions dataset

Method	Our model	w/o *Spectral similarity calculation module*	w/o *Time delay reflection module*	w/o *Node weighting module*
RMSE	**1101**	1201	1138	1117
PCC	**0.755**	0.689	0.741	0.749

6.4 Applicability of the Graph Construction Method

In this section, we analyze the applicability of our graph method to other GNNs. Even though we used Cola-GNN for prediction, our graph can be applied to other GNNs. To demonstrate this, we consider ST-GCN [29] and MTGNN [12] for forecasting. Table 11 shows their forecasting results on the ILI US-Regions dataset. In the table, both models showed better forecasting performance using the proposed graph.

Table 11: Forcasting results of other GNN models using our graph

Method	Original ST-GCN	ST-GCN w/*our graph*	Original MTGNN	MTGNN w/*our graph*
RMSE	1289	**1259**	1317	**1291**
PCC	0.619	**0.623**	0.599	**0.608**

7 Conclusion

In this paper, we proposed a novel graph construction method for GNN-based MTSF models. For effective graph construction, we considered spectral similarity between nodes, time-delay between time-series data of each node, and weight of hub nodes. We then constructed a MTSF model based on Cola-GNN using the constructed graph. To demonstrate the effectiveness of the proposed method, we compared the proposed model with other existing forecasting models in terms of RMSE and PCC using public datasets from various fields (e.g., infectious disease, exchange rate, and stock market data). Experimental results show that our model outperforms other comparative models in most cases, especially in long-term forecasting.

We also conducted ablation tests to verify the importance of each component in the proposed method. Experimental results show that all the components in the proposed method are indispensable. In addition, we showed that the proposed model has an advantage of interpretability through the visualization of the interrelationships between nodes. In the analysis of the heat map representing node similarity, it was confirmed that the proposed model effectively captures the interrelationships between nodes.

Even though our model outperforms other models, the accuracy of short-term forecasting is still unsatisfactory compared to long-term forecasting. In addition, in node weighting process, forecasting accuracy can be improved further by optimizing the parameters that adjust the node weighting function. In future works, we will consider a graph that fits the characteristics of different time-series data and optimize the best parameters for node weighting to improve short-term forecasting performance.

Funding Statement: This research was supported by Energy Cloud R&D Program (grant number: 2019M3F2A1073184) through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] C. Jinming, X. Jiang and B. Zhao, "Mathematical modeling and epidemic prediction of COVID-19 and its significance to epidemic prevention and control measures," *Journal of Biomedical Research & Innovation*, vol. 1, no. 1, pp. 1–19, 2020.

[2] J. Barros, M. Araujo and R. J. F. Rossetti, "Short-term real-time traffic prediction methods: A survey," in *Proc. of the Int. Conf. on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, Budapest, Hungary, pp. 132–139, 2015.

[3] J. G. Agrawal, V. Chourasia and A. Mittra, "State-of-the-art in stock prediction techniques," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 2, no. 4, pp. 1360–1366, 2013.

[4] H. Y. Cheng, Y. C. Wu, M. H. Lin, Y. L. Liu, Y. Y. Tsai *et al.,* "Applying machine learning models with an ensemble approach for accurate real-time influenza forecasting in Taiwan: Development and validation study," *Journal of Medical Internet Research*, vol. 22, no. 8, pp. e15394, 2020.

[5] Y. Wang and T. Aste, "Sparsification and filtering for spatial-temporal GNN in multivariate time-series," arXiv preprint: 2203.03991, 2022.

[6] D. Cheng, F. Yang, S. Xiang and J. Liu, "Financial time series forecasting with multi-modality graph neural network," *Pattern Recognition*, vol. 121, pp. e108218, 2020.

[7] Y. Cui, K. Zheng, D. Cui, J. Xie, L. Deng *et al.,* "METRO: A generic graph neural network framework for multivariate time series forecasting," in *Proc. of the VLDB Endowment*, Copenhagen, Denmark, vol. 15, no. 2, pp. 224–236, 2021.

[8] D. Cao, Y. Wang, J. Duan, C. Zhang, X. Zhu *et al.,* "Spectral temporal graph neural network for multivariate time-series forecasting," in *Proc. of the Advances in Neural Information Processing Systems*, Virtual Conference, vol. 33, pp. 17766–17778, 2020.

[9] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang *et al.,* "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2020.

[10] Z. Chen, M. Dehmer and Y. Shi, "A note on distance-based graph entropies," *Entropy*, vol. 16, no. 10, pp. 5416–5427, 2014.

[11] M. Xie, H. Yin, H. Wang, F. Xu, W. Chen *et al.,* "Learning graph-based poi embedding for location-based recommendation," in *Proc. of the ACM Int. Conf. on information and Knowledge Management*, Indiana, USA, pp. 15–24, 2016.

[12] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang *et al.,* "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proc. of the ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, California, USA, pp. 753–763, 2020.

[13] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò *et al.,* "Graph attention networks," in *Proc. of the Int. Conf. on Learning Representations*, Vancouver, Canada, pp. 1–12, 2017.

[14] G. Perone, "An ARIMA model to forecast the spread of COVID-2019 epidemic in Italy," arXiv: 2004.00382, 2004.
[15] C. García-Ascanio and C. Maté, "Electric power demand forecasting using interval time-series: A comparasion between VAR and iMLP," *Energy Policy*, vol. 38, no. 2, pp. 715–725, 2009.
[16] R. Senanayake, S. O'Callaghan and F. Ramos, "Predicting spatio-temporal propagation of seasonal influenza using variational gaussian process regression," in *Proc. of the Conf. on Artificial Intelligence*, Arizona, USA, vol. 30, no. 1, 2016.
[17] E. Spiliotis, S. Makridakis, A. A. Semenoglou and V. Assimakopoulos, "Comparison of statistical and machine learning methods for daily SKU demand forecasting," *Operational Research*, vol. 22, no. 3, pp. 3037–3061, 2020.
[18] S. Volkova, E. Ayton, K. Porterfield and C. D. Corley, "Forecasting influenza-like illness dynamics for military populations using neural networks and social media," *PLoS One*, vol. 12, no. 12, pp. e0188941, 2017.
[19] B. Chakravarthi, S. C. Ng, M. R. Ezilarasan and M. F. Leung, "EEG-based emotion recognition using hybrid CNN and LSTM classification," *Frontiers in Computational Neuroscience*, vol. 16, pp. 1–9, 2022.
[20] S. Jung, J. Moon, S. Park and E. Hwang, "Self-attention-based deep learning network for regional influenza forecasting," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 2, pp. 922–933, 2021.
[21] J. Park and E. Hwang, "A two-stage multistep-ahead electricity load forecasting scheme based on LightGBM and attention-BiLSTM," *Sensors*, vol. 21, no. 22, pp. 7697, 2021.
[22] X. Zhu, B. Fu, Y. Yang, Y. Ma, J. Hao *et al.,* "Attention-based recurrent neural network for influenza epidemic prediction," *BMC Bioinfomatics*, vol. 20, no. 18, pp. 1–10, 2019.
[23] Q. Tang, T. Fan, R. Shi, J. Huang and Y. Ma, "Prediction of financial time series using LSTM and data denoising methods," arXiv:2103.03505, 2021.
[24] M. Khashei and Z. Hajirahimi, "A comparative study of series arima/mlp hybrid models for stock price forecasting," *Communications in Statistics-Simulation Computation*, vol. 48, no. 9, pp. 1–16, 2019.
[25] J. Moon, Y. Noh, S. Park and E. Hwang, "Model-agnostic meta-learning-based region-adaptive parameter adjustment scheme for influenza forecasting," *Journal of King Saud University-Computer and Information Sciences*, vol. 35, no. 1, pp. 175–184, 2022.
[26] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.
[27] S. Bloemheuvel, J. Hoogen, D. Jozinovic, A. Michelini and M. Atzmueller, "Graph neural networks for multivariate time series regression with application to seismic data," *International Journal of Data Science and Analytics*, pp. 1–16, 2022.
[28] S. Deng, S. Wang, H. Rangwala, L. Wang and Y. Ning, "Cola-GNN: Cross-location attention based graph neural networks for long-term ILI prediction," in *Proc. of the ACM Int. Conf. on Information & Knowledge Management*, Galway, Ireland, pp. 245–254, 2020.
[29] B. Yu, H. Yin and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," arXiv preprint:1709.04875, 2017.
[30] H. Musbah, M. El-Hawary and H. Aly, "Identifying seasonality in time series by applying fast fourier transform," in *Proc. of the IEEE Electrical Power and Energy Conf. (EPEC)*, Montreal, Quebec, Canada, pp. 1–4, 2019.
[31] M. Srivastava, C. L. Anderson and J. H. Freed, "A new wavelet denoising method for selecting decomposition levels and noise thresholds," *IEEE Access*, vol. 4, pp. 3862–3877, 2016.
[32] G. O. Agaba, Y. N. Kyrychko and K. B. Blyuss, "Time-delayed SIS epidemic model with population awareness," *Ecological Complexity*, vol. 31, pp. 50–56, 2017.
[33] P. Senin, "Dynamic time warping algorithm review," in *Proc. of the Information and Computer Science Department University of Hawaii at Manoa Honolulu*, Honolulu, HI, USA, vol. 40, pp. 1–23, 2008.
[34] W. H. K. Tsui, H. O. Balli, A. Gilbey and H. Gow, "Forecasting of Hong Kong airport's passenger throughput," *Tourism Management*, vol. 42, pp. 62–76, 2014.

[35] Centers for Disease Control and Prevention (CDC), "National, regional, and state level outpatient illness and viral surveillance," 2022. [Online]. Available: https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html

[36] National Institute of Infectious Diseases, Japan, "Infectious Diseases Weekly Report," 2022. [Online]. Available: https://www.niid.go.jp/niid/en/idwre.html

[37] G. Lai, W. C. Chang, Y. Yang and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *Proc. of the ACM SIGIR Conf. on Research & Development in Information Retrieval*, Michigan, USA, pp. 95–104, 2018.

[38] F. Feng, H. Chen, X. He, J. Ding, M. Sun *et al.,* "Enhancing stock movement prediction with adversarial training," in *Proc. of the Int. Joint Conf. on Artificial Intelligence*, Macao, China, pp. 5843–5849, 2019.

[39] T. Hong, P. Pinson and S. Fan, "Global energy forecasting competition 2012," *International Journal of Forecasting*, vol. 30, no. 2, pp. 357–363, 2014.

[40] S. Z. Mohd Jamaludin, M. S. Mohd Kasihmuddin, A. I. Md Ismail, M. A. Mansor and M. F. Md Basir, "Energy based logic mining analysis with hopfield neural network for recruitment evaluation," *Entropy*, vol. 23, no. 1, pp. 40, 2020.

[41] M. S. M. Kasihmuddin, S. Z. M. Jamaludin, M. A. Mansor, H. A. Wahab and S. M. S. Ghadzi, "Supervised learning perspective in logic mining," *Mathematics*, vol. 10, no. 6, pp. 915, 2022.

[42] S. Z. M. Jamaludin, N. A. Romli, M. S. M. Kasihmuddin, A. Baharum, M. AMansor *et al.,* "Novel logic mining incorporating log linear approach," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 10, pp. 9011–9027, 2022.

[43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury *et al.,* "PyTorch: An imperative style, high-performance deep learning library," in *Proc. of the Advances in Neural Information Processing Systems*, Vancouver, Canada, pp. 8026–8037, 2019.

[44] I. S. Choi, D. S. Kang, J. H. Lee, M. W. Kang, Y. D. Song *et al.,* "Prediction of the industrial stock price index using domestic and foreign economic indices," *Journal of the Korean Data and Information Science Society*, vol. 23, no. 2, pp. 271–283, 2012.