GSA-Forecaster: Forecasting Graph-Based Time-Dependent Data with Graph Sequence Attention

Yang Li, Di Wang, *Member IEEE,* and José M. F. Moura, *Life Fellow IEEE*

Abstract—Forecasting graph-based time-dependent data has many practical applications. This task is challenging as models need not only to capture spatial dependency and temporal dependency within the data, but also to leverage useful auxiliary information for accurate predictions. In this paper, we analyze limitations of state-of-the-art models on dealing with temporal dependency. To address this limitation, we propose GSA-Forecaster, a new deep learning model for forecasting graph-based time-dependent data. GSA-Forecaster leverages graph sequence attention (GSA), a new attention mechanism proposed in this paper, for effectively capturing temporal dependency. GSA-Forecaster embeds the graph structure of the data into its architecture to address spatial dependency. GSA-Forecaster also accounts for auxiliary information to further improve predictions. We evaluate GSA-Forecaster with large-scale real-world graph-based time-dependent data and demonstrate its effectiveness over state-of-the-art models with 6.7% RMSE and 5.8% MAPE reduction.

Index Terms—attention, graph data, spatial dependency, temporal dependency, Transformer, Forecaster, GSA-Forecaster.

1 Introduction

In the real world, many applications involve a set of dependent time series, where data may be dependent across time in each time series (intra-dependency) and dependent across different time series (inter-dependency). We refer to the former as *temporal dependency* and the latter as *spatial dependency*, even if physical space is not involved. Such time series are of interest in many fields, including economics [1], climatology [2], public health [3], transportation [4], cloud computing [5], [6], and Internet of things (IoT) [7], [8] to name a few. For example, the hourly taxi ride-hailing demand at various urban locations is a set of dependent time series, where the taxi demand at some locations may often change at the same time. The sales volumes of different products in a retail store are also a set of dependent time series, where the sales volumes of different products may depend on each other (in this example, spatial dependency refers to the relations between the sales volumes of different products; physical space is not involved). Following practice in prior work [9], we capture the data spatial dependency through a graph. We refer to these dependent time series as *graph-based time-dependent data*.

Fig. 1 illustrates how we formalize this type of data. In its underlying graph G, a node indexes a time series, and an edge corresponds to spatial dependency between two series. A graph-based time-dependent data is a series of graph signals $\{\mathbf{x}_t \mid \mathbf{x}_t \in \mathbb{R}^N,\ t = 1, \ldots, T\}$, where each *graph signal* $\mathbf{x}_t$ collects the data x_t^i at each node v_i of graph G

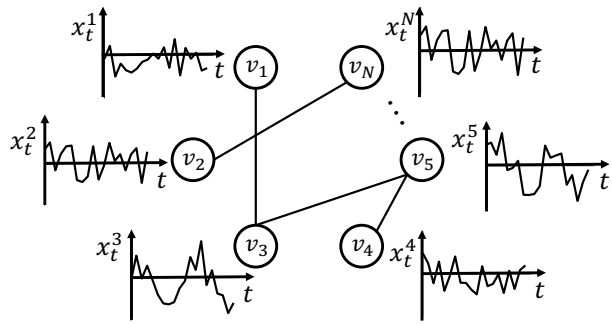


Fig. 1: Definition of graph-based time-dependent data.

at a certain time t, i.e., $\mathbf{x}_t = \left[x_t^1, \cdots, x_t^N\right]^\top$, N is the number of time series and the number of nodes in the graph G. We may visualize $\mathbf{x}_t$ as a heatmap. For example, the heatmap of hourly taxi demand over multiple locations at a certain time is a graph signal (shown in Fig. 2) where in this case the graph represents the dependency between the taxi demand at different locations of the city, while a series of such heatmaps is a graph-based time-dependent data (shown in Fig. 3).[1]

Forecasting graph-based time-dependent data has many practical applications that are important to our life and society. However, it is challenging as models must capture

- *Y. Li is with Carnegie Mellon University, Pittsburgh, PA 15213, USA (email: jerryyangli@gmail.com).*
- *D. Wang is with Microsoft Cloud & AI, Redmond, WA 98052, USA (email: wangdi@microsoft.com).*
- *J. M. F. Moura (corresponding author) is with Carnegie Mellon University, Pittsburgh, PA 15213, USA (email: moura@andrew.cmu.edu).*

1. A subset of graph-based time-dependent data is called *spatial time series* [10], [11] or *spatial- and time-dependent data* [9], where the graph represents the relations between the data at different physical locations. However, graph-based time-dependent data is not necessarily associated with physical locations, e.g., sales volumes of different products. Graph-based time-dependent data is different from *temporal graphs* [12], another type of graph data, where each node of the graph has no associated data and only the edges of the graph change over time.

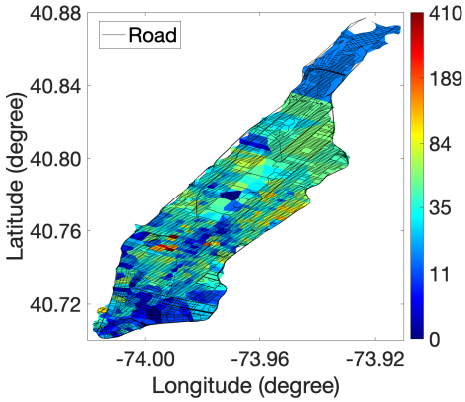


Fig. 2: Example of graph signal: hourly taxi ride-hailing demand heatmap on March 7, 2016 at 7:00 a.m. – 7:59 a.m. in Manhattan, New York City. We split Manhattan into 471 regions and measure the taxi demand in each region.

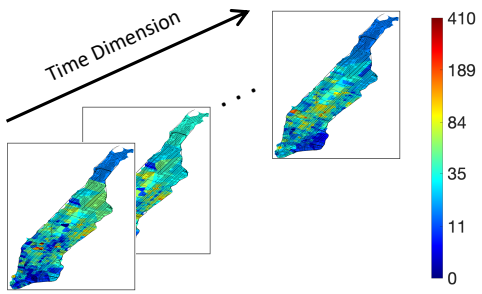


Fig. 3: Example of graph-based time-dependent data: a series of hourly taxi ride-hailing demand heatmaps.

spatial dependency and *temporal dependency* within the data and also account for useful *auxiliary information* for accurate predictions. First, the graph structure of the data, which captures spatial dependency between different time series, is often unknown in many applications. Models should account for the graph structure and leverage it for better forecasting. Second, graph-based time-dependent data may exhibit both *short-range* and *long-range* temporal dependency; for example, the data at the next time instant (for prediction) may be similar to not only recent previous data (i.e., short-range dependency) but also to data from long time ago (i.e, long-range dependency). This is further complicated by *non-stationarity* due to unexpected incidents or events that impact the temporal dependency between the next data and the historical data. To take these into account, models should pay careful attention to the historical data that the next data truly depends on. Third, auxiliary information may impact the evolution of graph-based time-dependent data. For example, weather may affect taxi demand at certain locations. Such auxiliary information is helpful for forecasting and should also be accounted for by models. In this paper, we use real-world data to illustrate these three challenges (see Section 2 for details).

In order to forecast graph-based time-dependent data, conventional models such as auto-regressive integrated moving average (ARIMA) [13] as well as more recent work like vector autoregression (VAR) [13] and causal graph processes [14] usually impose strong stationarity assumptions

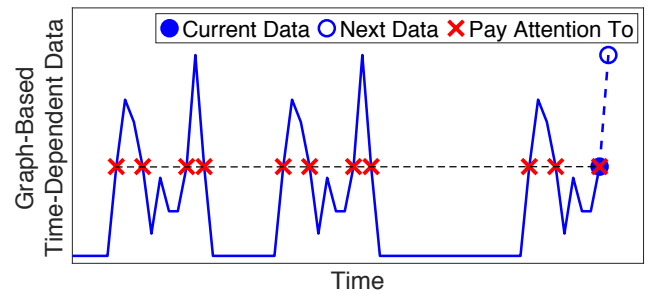


Fig. 4: Illustration of limitations of standard attention. In this example, there is only a single node in the graph structure. The graph-based time-dependent data has a pattern recurring over time, though not necessarily periodically.

on the data that often do not hold [4], [15]. To address the non-stationarity and highly nonlinear nature of the data, deep learning based models have been proposed [4], [5], [7], [8], [15]–[23]. These models usually use recurrent neural networks (RNN), convolutional neural networks (CNN), deep belief networks (DBN), or their variants to capture temporal dependency within the data. However, these networks may not be capable of capturing long-range temporal dependency between data at distant time instants [24], [25]. Recently, Forecaster [9], an attention mechanism-based deep learning model, has been proposed for capturing long-range temporal dependency within the data. The attention mechanism [25], which we refer to as the *standard attention mechanism* in this paper, was originally developed by the natural language processing (NLP) community and has been used by Transformer [25] and other alike NLP architectures [25]–[30] for capturing long-range temporal dependency, resulting in significant performance improvements on NLP tasks. Forecaster [9] extends Transformer to forecast graph-based time-dependent data by embedding the graph structure of the data into its architecture to encode spatial dependency. When the graph structure is unknown, Forecaster leverages the theory of Gaussian Markov random fields [31] to learn it. Forecaster has demonstrated state-of-the-art performance on predicting graph-based time-dependent data [9].

Though Forecaster achieves great success, we observe that the standard attention mechanism used in Forecaster is not sufficient to capture temporal dependency within graph-based time-dependent data.[2] For example, in Fig. 4, in order to predict the next data (blue empty circle), Forecaster first uses the current data (blue dot) as its initial estimate.[3] Then, Forecaster employs the standard attention mechanism to compare this initial estimate (equal to the current data) with each historical data and pays attention to similar historical ones (red crosses). As a result, Forecaster uses the historical data values that are similar to the current data value to predict the next data point, but this can make the prediction

2. The standard attention mechanism successfully captures temporal dependency in NLP tasks as consecutive words in such tasks usually have a close semantic relation. However, for graph-based time-dependent data with numerical values, data at consecutive time instants may be very different, especially when non-stationarity arises. As we will show in Section 3.3, this difference makes the standard attention mechanism ineffective in capturing temporal dependency for numerical valued data.

3. Here, the *data* at a time instant refers to the *graph signal* at the time instant; we use them interchangeably in this paper.

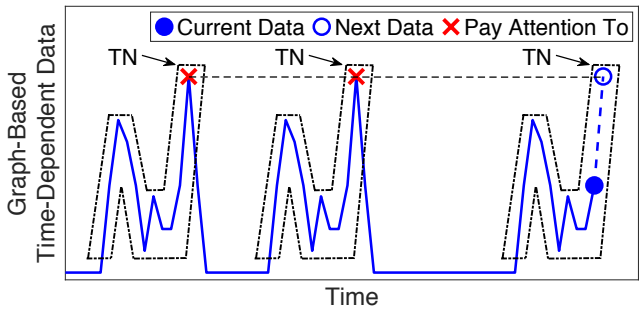


Fig. 5: Illustration of Graph Sequence Attention (GSA) and how GSA addresses the limitation of standard attention shown in Figure 4.

highly inaccurate. The reason for this is two-fold: *1) the stark difference between the current and next data makes the initial estimation using current data highly inaccurate; 2) Forecaster pays attention to the wrong historical data (e.g., red crosses) because Forecaster identifies it based solely on the current data, i.e., extrapolating inaccurate temporal dependency.*

To address this challenge, we propose *graph sequence attention* (GSA), a new attention mechanism for effectively capturing temporal dependency.[4] GSA addresses the limitation of standard attention (i.e., comparing the *inaccurate* initial estimate with each historical data that may lead to *inaccurate* attention). It achieves this by leveraging more recent data points to better capture temporal dependency. We use Fig. 5 to illustrate how GSA works. As shown in the figure, GSA groups together several recent past data points with the initial estimate, forming a *temporal neighborhood* (TN) for the next time instant. For each historical data, GSA also groups together the same number of previous data points with the historical data, forming a historical temporal neighborhood.[5] GSA then compares the temporal neighborhood for the next time instant with each historical temporal neighborhood, which involves *multiple comparisons* between their data, i.e., comparing their latest data, their second latest data, . . . , and their earliest data. After identifying similar historical temporal neighborhoods, GSA pays attention to their latest data points and uses the values of these data points to predict the next data. *As GSA uses temporal neighborhoods (rather than comparing individual data as standard attention does) to find where and how to pay attention for forecasting, it can better tolerate the inaccuracy and noise of individual data points (e.g., tolerate inaccuracy in the initial estimate), resulting in better attention.*

By leveraging GSA, we propose *GSA-Forecaster*, a new deep learning model for forecasting graph-based time-dependent data. GSA-Forecaster relies on graph sequence attention to capture temporal dependency. Similar to Forecaster, GSA-Forecaster embeds graph structure into its architecture to encode spatial dependency. GSA-Forecaster also accounts for crucial auxiliary information. We apply GSA-Forecaster to predicting the taxi ride-hailing demand in Manhattan, New York City. We divide Manhattan into 471 regions (locations) and forecast the hourly taxi ride-hailing demand over these 471 locations. We start by constructing a graph to represent the relations between the data over these locations by using an approach suggested by Forecaster [9]. Our evaluation uses the dataset of all the taxi trips recorded in Manhattan between January 1, 2009 and June 30, 2016 (the NYC Taxi dataset [32]) — 1.063 billion trips in total. Evaluation results show GSA-Forecaster outperforms state-of-the-art predictors. For example, compared with Forecaster, GSA-Forecaster reduces the root-mean-square error (RMSE) by 6.7% and the mean absolute percentage error (MAPE) by 5.8%. In addition, we applied GSA-Forecaster to forecasting the traffic speed collected at 207 sensors on the highways of Los Angeles between March 1, 2012 and June 27, 2012 (the METR-LA dataset [4]) and the traffic speed collected at 325 sensors in the San Francisco Bay Area between January 1, 2017 and June 30, 2017 (the PEMS-BAY dataset [4]).[6] Overall, GSA-Forecaster achieves higher forecasting accuracy on the METR-LA and PEMS-BAY datasets than state-of-the-art predictors.

This paper makes the following major contributions:

- We analyze real-world graph-based time-dependent data and demonstrate the benefits of forecasting such data by capturing spatial and temporal dependencies as well as by accounting for auxiliary information.
- We propose graph sequence attention (GSA), a new attention mechanism that leverages temporal neighborhood information to enable multiple comparisons for effectively locating the historical data that better captures temporal dependency. This overcomes the limitations of the standard attention mechanism used by Transformer [25], Forecaster [9], and other work on handling temporal dependency.
- We propose a new deep learning model called GSA-Forecaster for forecasting graph-based time-dependent data. This model captures both spatial and temporal dependency as well as accounts for auxiliary information.
- We apply GSA-Forecaster on a large-scale real-world dataset (NYC Taxi) and two medium-scale real-world datasets (METR-LA and PEMS-BAY) and demonstrate its effectiveness over prior methods on prediction tasks.

2 Challenges of Forecasting Graph-based Time-Dependent Data

In this section, we use the hourly taxi ride-hailing demand in New York City [32], a typical graph-based time-dependent data, to demonstrate the challenges of forecasting such data. Models need to address these challenges for achieving good forecasting accuracy.

2.1 Spatial Dependency

The first challenge is to model *spatial dependency*. Graph-based time-dependent data is a set of dependent time series. Spatial dependency refers to the dependency between the different time series. Two time series show spatial dependency when they often change accordingly to each other. To illustrate this, consider the hourly taxi demand around

4. Our proposed attention mechanism is called graph sequence attention as it performs attention-related operations over graph signals.

5. A temporal neighborhood also contains later data in some cases (see Section 4).

6. Traffic speed is measured every five minutes.

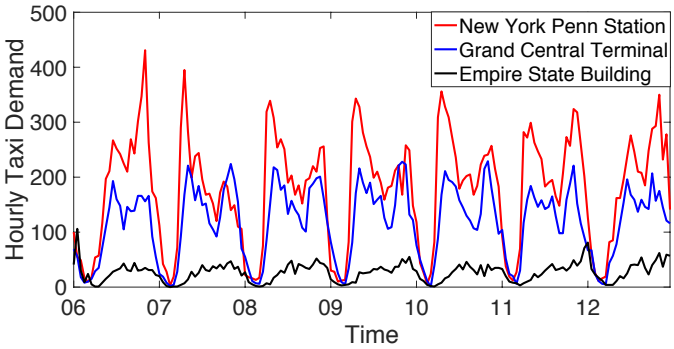


Fig. 6: Spatial dependency illustration: hourly taxi demand at locations close to New York Penn Station, Grand Central Terminal, and the Empire State Building from Sunday, March 6, 2016 at 12:00 a.m. to Saturday, March 12, 2016 at 11:59 p.m. Each tick label in the x-axis refers to the beginning of a date (e.g., "06" refers to March 6, 2016 at 12:00 a.m.).

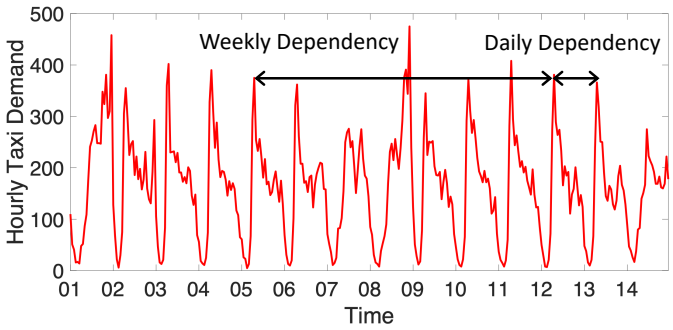


Fig. 7: Temporal dependency illustration: hourly taxi demand at a location near New York Penn Station from Sunday, May 1, 2016 at 12:00 a.m. to Saturday, May 14, 2016 at 11:59 p.m. Each tick label in the x-axis refers to the beginning of a date (e.g., "01" refers to May 1, 2016 at 12:00 a.m.).

New York Penn Station and Grand Central Terminal, two major train stations in Manhattan, New York City. As Fig. 6 shows, when the taxi demand near New York Penn Station increases (decreases), the demand near Grand Central Terminal will generally increase (decrease) with a 0.82 Pearson correlation coefficient. Therefore, the taxi demand at these two locations exhibits spatial dependency. Spatial dependency can be *short-range* (between adjacent locations) and *long-range* (between distant locations), and it is not necessarily related to the physical distance between locations. For example, though the physical distance between New York Penn Station and Grand Central Terminal is larger than the distance between either of them to the Empire State Building, their taxi demands correlate more (i.e., exhibit long-range spatial dependency) than either of their taxi demands correlates with the demand near the Empire State Building (0.33 and 0.48 correlation coefficients, respectively).

Leveraging spatial dependency can lead to better forecasting graph-based time-dependent data, as demonstrated in prior work [9]. For example, if we know there is spatial dependency between time series A and B, we can encode the data in time series A by considering the data in time series B, alleviating noise or fluctuations caused by random factors; we can also leverage their spatial dependency to refine the predictions of future data in time series A by considering the predictions of future data in time series B, and vice versa.

Following practice in [9], we use a graph to represent spatial dependency, where a node of the graph indexes a time series, and an edge corresponds to spatial dependency between two time series. Such graph is not provided as prior knowledge in many applications (e.g., in predicting the taxi demand in Manhattan). Models need to account for this graph and leverage it for forecasting.

2.2 Temporal Dependency

The second challenge relates to capturing *temporal dependency*. Temporal dependency refers to relations between data at different time instants. Data at two time instants show temporal dependency if they are similar to each other. Capturing temporal dependency allows a model to leverage similar historical data/trend to better forecast future data.

We find that graph-based time-dependent data can show both *short-range* and *long-range* temporal dependency. For example, Fig. 7 shows that hourly taxi demand near New York Penn Station manifests both daily dependency (short-range; 24 hours between similar data) and weekly dependency (long-range; 168 hours between similar data). Models must be able to capture both short-range and long-range temporal dependency for accurate forecasting.

2.3 Auxiliary Information

The third challenge is to account for *auxiliary information* that can augment existing data used for forecasting. Some auxiliary information may impact the evolution of graph-based time-dependent data. Accounting for such information helps improve the accuracy of forecasting. We illustrate this with the taxi demand around the Javits Center, a major convention center in New York City. Fig. 8 shows the impact of weather (temperature) on the average hourly taxi demand between January 1, 2009 and June 30, 2016, for different hours of the day. We can see that when there is notable taxi demand (e.g., > 20), a decrease in temperature leads to an increase in the taxi demand, which may be because colder weather encourages people to choose more convenient ways of travel (e.g., taxi). Major exogenous factors like weather can cause taxi demand changes (but these exogenous factors are not included in the taxi demand dataset). Models need to consider them as helpful auxiliary information for better predicting taxi demand.

3 Forecaster and Its Limitations

State-of-the-art models on addressing these challenges can be characterized into two categories: (1) using RNN or CNN-based mechanisms or (2) using attention mechanisms as what Transformer does. The attention-based models generally achieve better performance due to better long-range temporal dependency encoding. Forecaster [9] is a representative attention-based model that achieves state-of-the-art performance. In this section, we briefly review Forecaster that our proposed approach builds upon. We first describe how Forecaster addresses the challenges of spatial

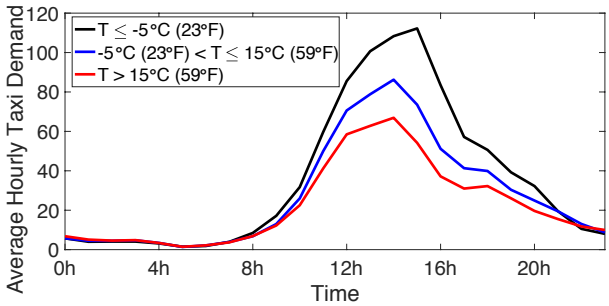


Fig. 8: Auxiliary information illustration: the impact of temperature (T) on the average hourly taxi demand between January 1, 2009 and June 30, 2016 at a location near the Javits Center, for different hours of the day.

and temporal dependency (Sections 3.1 and 3.2).[7] Then, we analyze the limitations of standard attention mechanism that Forecaster relies upon to capture temporal dependency (Section 3.3).

3.1 Forecaster: Spatial Dependency

To encode spatial dependency, there are two key problems to be solved: (1) determining the graph that represents spatial dependency when the graph is implicit or not provided; (2) leveraging the graph for prediction.

Determining graph. The statistical model of Gaussian Markov random fields [31] is adopted to learn the spatial dependency graph from data. This way, we can infer spatial dependency based on the variation of the data at each node. This is usually better than estimating spatial dependency based on ad-hoc metrics or heuristics. For instance, in the example shown in Fig. 6, using a physical distance metric to estimate spatial dependency (i.e., assuming that nodes that are physically adjacent to each other are dependent) may mistakenly consider the taxi demand at New York Penn Station and the Empire State Building as spatially dependent because they are physically adjacent, and may mistakenly neglect the spatial dependency between the taxi demand at New York Penn Station and Grand Central Terminal just because of the longer distance between them. For this example, by exploiting the statistical correlation between time series, the theory of Gaussian Markov random fields successfully discovers the spatial dependency between the taxi demand at New York Penn Station and Grand Central Terminal [9].

Gaussian Markov random fields model a graph signal $\mathbf{x}_t$ that consists of the data x_t^i at each node v_i at a certain time t, i.e., $\mathbf{x}_t = \left[x_t^1, \cdots, x_t^N\right]^\top$, as a random vector drawn from a multivariate Gaussian distribution. The probability density function of $\mathbf{x}_t$ is:

$$f\left(\mathbf{x}_t\right) = \frac{|Q|}{(2\pi)^{N/2}} \exp\left(-\frac{1}{2}\left(\mathbf{x}_t - \boldsymbol{\mu}\right)^\top Q\left(\mathbf{x}_t - \boldsymbol{\mu}\right)\right) \quad (1)$$

where $\boldsymbol{\mu}$ and Q are the mean and precision matrix of the distribution.

7. Forecaster also encodes auxiliary information. Details are omitted due to page limits.

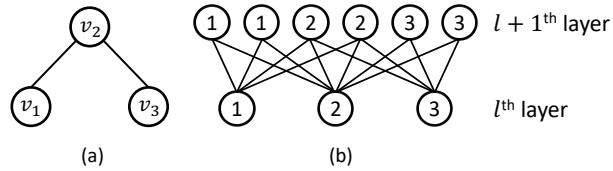


Fig. 9: Example of (a) graph structure and (b) sparse linear layer (neurons marked as '1,', '2,' and '3' are for nodes v_1, v_2, and v_3, respectively).

The precision matrix Q characterizes the relations between data at different nodes. It can be estimated by solving *graphical lasso* [33], a sparse-penalized maximum likelihood estimator. After estimating the precision matrix Q, Forecaster uses it to calculate the conditional correlation between the data of every two nodes [9], [34]. A conditional correlation with large absolute value reveals the spatial dependency between two nodes and hence an edge is introduced to connect them.

Leveraging graph. Forecaster embeds graph structure into the architecture of Transformer, enabling the modified Transformer to encode spatial dependency. To do this, Forecaster proposes *sparse linear layers* (neural layers that are sparse and represent graph structure) and uses them as substitutes for the fully-connected layers within Transformer (while keeping the nonlinear activation functions and other components). Fig. 9 shows an example of a graph with three nodes v_1, v_2, and v_3 and two sparse linear layers (l^{th} and $l+1^{\text{th}}$ layers) designed based on this graph.[8] To design a sparse linear layer, Forecaster first assigns neurons to the nodes of the graph for encoding their data. In Fig. 9b, as an example, one neuron of the lth layer and two neurons of the $l+1^{\text{th}}$ layer are assigned to each node in Fig. 9a. Then, Forecaster connects the neurons assigned to the same node or graph-dependent nodes, enabling the learnable weight on each connection to encode the impact of spatial dependency. In Fig. 9b, Forecaster connects the neurons assigned to nodes v_1 and v_2 (and connects the neurons assigned to nodes v_2 and v_3) as these nodes are strongly dependent. However, as nodes v_1 and v_3 are not dependent, their neurons are not connected. In this way, the encoding for the data at a node is only impacted by its own encoding and the encoding of its dependent nodes, which captures the spatial dependency between nodes.

Sparse linear layers are a type of graph convolutional network similar to GCN [35] and TAGCN [36], [37]. All of these networks enrich the encodings of a node by the encodings of its adjacent nodes. The only major difference is, instead of treating adjacent nodes equally, sparse linear layers learn different weights for different adjacent nodes, considering that a node may depend with different strengths on its different adjacent nodes.

3.2 Forecaster: Temporal Dependency

Forecaster leverages the *standard attention mechanism*, which has been employed in Transformer and other alike architec-

8. Different sparse linear layers can be designed based on this graph, e.g., by changing the number of neurons assigned to different nodes. This is a user choice.

tures for NLP tasks [25]–[30], to capture temporal dependency within graph-based time-dependent data.[9]

Forecaster first encodes the graph signal $\mathbf{x}_t$ at each time instant t with encoding $\mathbf{e}_t \in \mathbb{R}^{d_{\text{model}}}$, where d_{model} represents the dimension of the encoding. To predict the encoding $\mathbf{e}_{t+1}$ of the next graph signal $\mathbf{x}_{t+1}$, Forecaster uses the encoding of the current graph signal as its initial estimate $\widetilde{\mathbf{e}}_{t+1}$:

$$\widetilde{\mathbf{e}}_{t+1} = \mathbf{e}_t \tag{2}$$

Then, Forecaster employs the standard attention mechanism to find historical graph signals that are similar to the initial estimate and uses them to develop a refined estimate. Intuitively, developing an estimate based on these *multiple* similar graph signals alleviates the impact of noise of *each individual* graph signal on the estimate,[10] as suggested by the central limit theorem [38]. This improves the initial estimate that is based solely on the current graph signal.

Specifically, Forecaster compares the initial estimate $\widetilde{\mathbf{e}}_{t+1}$ with the encoding of each historical graph signal $\mathbf{e}_{t+i}$ ($i = -T+1, \ldots, 0$; T is the length of the history), and obtains their *similarity scores* $s^{(h)}_{t+1,\,t+i}$ from multiple perspectives (a *perspective* is also termed as a *head* [9], [25]; $h = 1, \ldots, H$, H is the number of heads):

$$s^{(h)}_{t+1,\,t+i} = \left\langle W^{(h)}_Q \widetilde{\mathbf{e}}_{t+1},\ W^{(h)}_K \mathbf{e}_{t+i} \right\rangle \tag{3}$$

where $s^{(h)}_{t+1,\,t+i} \in \mathbb{R}$ is the similarity score between $\widetilde{\mathbf{e}}_{t+1}$ and $\mathbf{e}_i$ under head h; $W^{(h)}_Q, W^{(h)}_K \in \mathbb{R}^{\frac{d_{\text{model}}}{H} \times d_{\text{model}}}$ are learnable parameter matrices for head h that are implemented as a sparse linear layer, respectively; $\langle \cdot, \cdot \rangle$ is the inner product between two vectors, measuring their similarity.

After this step, Forecaster passes the similarity scores through a softmax layer [39] and obtains the *attention scores* $\alpha^{(h)}_{t+1,\,t+i}$ between the initial estimate $\widetilde{\mathbf{e}}_{t+1}$ and the encoding of each historical graph signal $\mathbf{e}_{t+i}$:

$$\alpha^{(h)}_{t+1,\,t+i} = \frac{\exp\left(s^{(h)}_{t+1,\,t+i}\right)}{\sum_{k=-T+1}^{0} \exp\left(s^{(h)}_{t+1,\,t+k}\right)} \tag{4}$$

where $\alpha^{(h)}_{t+1,\,t+i} \in (0, 1)$, $\sum_{k=-T+1}^{0} \alpha^{(h)}_{t+1,\,t+k} = 1$.

The attention scores characterize the amount of attention that Forecaster pays to each historical graph signal. Forecaster uses the attention scores as weights for historical graph signals to predict the next graph signal:

$$\begin{aligned} \widehat{\mathbf{e}}_{t+1} &= \widetilde{\mathbf{e}}_{t+1} + W_O \left[\Delta\widetilde{\mathbf{e}}^{(1)}_{t+1} \,\Big\|\, \Delta\widetilde{\mathbf{e}}^{(2)}_{t+1} \,\Big\|\, \cdots \,\Big\|\, \Delta\widetilde{\mathbf{e}}^{(H)}_{t+1} \right] \\ \Delta\widetilde{\mathbf{e}}^{(h)}_{t+1} &= \textstyle\sum_{k=-T+1}^{0} \alpha^{(h)}_{t+1,\,t+k} W^{(h)}_V \mathbf{e}_{t+k} \end{aligned} \tag{5}$$

The resulting $\widehat{\mathbf{e}}_{t+1}$ is a refined estimate for the encoding of the next graph signal; $W_O \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ is a learnable parameter matrix that combines the update $\Delta\widetilde{\mathbf{e}}^{(h)}_{t+1}$ from each head h; $W^{(h)}_V \in \mathbb{R}^{\frac{d_{\text{model}}}{H} \times d_{\text{model}}}$ is a learnable parameter matrix for calculating the updates; W_O and W_V are implemented as a sparse linear layer, respectively; $\cdot\|\cdot$ represents a concatenation of two vectors.

9. We give a simplified review of how Forecaster leverages the standard attention mechanism. More details are in [9].

10. The noise refers to the impact of random factors on graph signals, e.g., a random taxi pickup.

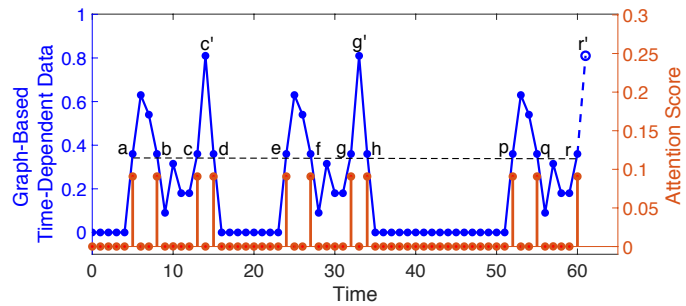


Fig. 10: Results of standard attention for the example shown in Fig. 4.

After getting $\widehat{\mathbf{e}}_{t+1}$, Forecaster decodes it and obtains the predicted next graph signal $\widehat{\mathbf{x}}_{t+1}$. This way, Forecaster completes a single-step forecasting (i.e., forecasting the next graph signal). Forecaster repeats the above process (by advancing the current timestamp t to future timestamps) for multi-step forecasting.

3.3 Limitations of Standard Attention

The standard attention mechanism directly relates historical graph signals with predicted graph signals, capturing temporal dependency better than RNN or CNN-based approaches. However, it is still not sufficient to fully capture temporal dependency. To illustrate this, we revisit the example shown in Fig. 4 and perform the standard attention mechanism (described in Section 3.2) using this example. For simplicity, we assume that standard attention has a single head (i.e., $H = 1$) and parameter matrices W_Q and W_K are equal to the identity matrix (i.e., $W_Q = W_K = I$); and Forecaster encodes the graph signal x_t at each time instant with encoding $\mathbf{e}_t = \left[e^1_t, \cdots, e^n_t\right]^\top$, such that

$$x_t = \sum_{i=1}^{n} \left(\frac{e^i_t + 1}{2} \right) \cdot 2^{m-i} \tag{6}$$

where $n = 8$, $m = 0$, $e^i_t = 1$ or -1.

Fig. 10 shows the resulting attention scores when Forecaster predicts the next graph signal (data point r'). We can see that Forecaster pays attention to the historical graph signals (data points a, b, c, ..., r) that are similar to the current graph signal (data point r), rather than those (data points c' and g') that are similar to the next one. The reason behind this is: (1) Forecaster uses the current graph signal as an initial estimate for the next one, introducing error in the initial estimate; (2) the standard attention mechanism compares *only* the inaccurate initial estimate with the historical graph signals, causing the initial erroneous estimate to induce Forecaster to pay attention to the wrong data. As a result, Forecaster fails to capture the temporal dependency between the data, leading to erroneous prediction.

4 GSA-Forecaster: Forecaster with Graph Sequence Attention

In this section, we introduce GSA-Forecaster, our proposed model for forecasting graph-based time-dependent data. We first formalize the forecasting task that GSA-Forecaster is applied to (Section 4.1). We then present the architecture

of GSA-Forecaster (Section 4.2). Lastly, we describe the core architecture components, our proposed graph sequence attention (GSA) for predicting and filtering purposes in Sections 4.3 and 4.4, respectively.

4.1 Forecasting Task

GSA-Forecaster performs multi-step forecasting: predicting T' future graph signals based on T historical graph signals and $T+T'$ historical/future auxiliary information. The task can be formalized as learning a function $f(\cdot)$ that performs the following mapping:

$$\left\{ \begin{array}{c} \{\mathbf{x}_{t-T+1}, \cdots, \mathbf{x}_t\} \\ \{\mathbf{a}_{t-T+1}, \cdots, \mathbf{a}_{t+T'}\} \end{array} \right\} \xrightarrow{f(\cdot)} \{\mathbf{x}_{t+1}, \cdots, \mathbf{x}_{t+T'}\} \quad (7)$$

where $\mathbf{x}_t$ is the graph signal at time t, $\mathbf{x}_t = \left[x_t^1, \cdots, x_t^N\right]^\top \in \mathbb{R}^N$, with x_t^i the data at node i at time t; $\mathbf{a}_t \in \mathbb{R}^A$ is the auxiliary information at time t, A the dimension of the auxiliary information.[11] More specifically, GSA-Forecaster performs the multi-step forecasting in an autoregressive manner: after predicting the graph signals $\{\mathbf{x}_{t+i} \mid i = 1, \ldots, k-1\}$, GSA-Forecaster uses them together with other known information to predict the next graph signal $\mathbf{x}_{t+k}$. GSA-Forecaster iterates this process (i.e., $k = 1, \ldots, T'$) until predicting all the T' future graph signals.[12] This way, GSA-Forecaster transforms the multi-step forecasting task into a series of single-step tasks of forecasting $\mathbf{x}_{t+k}$. In the rest of Section 4, we focus on how GSA-Forecaster solves this single-step forecasting task. We use the following conventions: t as current timestamp, $\widehat{\mathbf{x}}_{t+k}$ as our prediction result for $\mathbf{x}_{t+k}$, and, by abuse of notation, $\{\mathbf{x}_{t+i} \mid i = 1, \ldots, k-1\}$ as our previously predicted graph signals.[13]

4.2 Architecture of GSA-Forecaster

As Fig. 11(a) shows, the architecture of GSA-Forecaster consists of *embeddings*, an *encoder*, and an *decoder*. The encoder and the decoder employ two types of graph sequence attention with similar mathematical formalism but different goals (i.e., *GSA (Filtering)* for noise filtering and *GSA (predicting)* for data prediction, respectively) as shown in Fig. 11(b). To capture spatial dependency, GSA-Forecaster uses sparse linear layers as shown in Section 3.1 that reflect the graph structure throughout its architecture.

4.2.1 Embeddings

GSA-Forecaster has two embedding components: graph data embedding and auxiliary embedding. They generate embedding representations for the graph data and auxiliary information, which are leveraged by the encoder and decoder to predict the next future graph signal. The graph data embedding is a feedforward neural network [40] implemented with sparse linear layers and nonlinear activation functions. It encodes each graph signal $\mathbf{x}_{t+i}$ with encoding $\mathbf{e}_{t+i} \in \mathbb{R}^{d_{\text{model}}}$, $i = -T+1, \ldots, k-1$. The auxiliary embedding is a feedforward neural network implemented with fully-connected layers and nonlinear activation functions. It encodes each auxiliary information $\mathbf{a}_{t+i}$ with encoding $\mathbf{e}_{t+i}^a \in \mathbb{R}^{d_{\text{aux}}}$, $i = -T+1, \ldots, k$.

4.2.2 Encoder

The encoder builds a *refined* encoding for each historical graph signal. In practice, random factors (e.g., a random taxi pickup) may impact the graph signals, adding noise to them. It is desirable to remove such noise from the encodings of historical graph signals, so that our prediction can be based on less noisy historical data. To do this, the encoder takes the encodings of historical graph signals $\{\mathbf{e}_{t+i} \mid i = -T+1, \ldots, 0\}$ and the encodings of associated auxiliary information $\left\{\mathbf{e}_{t+i}^a \mid i = -T+1, \ldots, 0\right\}$ as inputs, passes them through N_{encoder} *encoder layers*, and outputs the *filtered encodings* of historical graph signals $\left\{\mathbf{e}'_{t+i} \mid i = -T+1, \ldots, 0\right\}$.

Each encoder layer consists of a *graph sequence attention (filtering)* layer and a feedforward neural network. *The graph sequence attention (filtering) layer is the core component for noise filtering* (see Section 4.4). It filters out noise by enriching the encoding of each historical graph signal with the encodings of other similar historical graph signals (with the help of the auxiliary information). The feedforward neural network (consisting of sparse linear layers and nonlinear activation functions) refines the encoding of each graph signal by capturing the impact of spatial dependency.

4.2.3 Decoder

The decoder predicts the next graph signal. It uses the encoding of the last predicted graph signal $\mathbf{e}_{t+k-1}$ as the *initial estimate* for the encoding of the next graph signal. It also forms a *previous data sequence* $\{\mathbf{e}'_{t+i} \mid i = -T+1, \ldots, k-1\}$ by combining the filtered encodings of historical graph signals $\{\mathbf{e}'_{t+i} \mid i = -T+1, \ldots, 0\}$ with the encodings of previously predicted graph signals $\{\mathbf{e}_{t+i} \mid i = 1, \ldots, k-1\}$ (we let $\mathbf{e}'_{t+i} = \mathbf{e}_{t+i}$, $i = 1, \ldots, k-1$). The decoder takes the initial estimate $\mathbf{e}_{t+k-1}$, the previous data sequence $\{\mathbf{e}'_{t+i} \mid i = -T+1, \ldots, k-1\}$, and relevant auxiliary information $\{\mathbf{e}_{t+i}^a \mid i = -T+1, \ldots, k\}$ as inputs, passes them through N_{decoder} *decoder layers*, and obtains the *final estimate* for the encoding of the next graph signal $\widehat{\mathrm{e}}_{t+k}$.

Each decoder layer consists of a *graph sequence attention (predicting)* layer and a feedforward network. *The graph sequence attention layer (predicting) is the core component for capturing temporal dependency* (see Section 4.3). It refines the estimate for the next graph signal by learning from the previous data sequence with the help of the auxiliary information. The feedforward network, which is made of sparse linear layers and nonlinear activation functions, further refines the estimate by accounting for spatial dependency.

After obtaining the final estimate $\widehat{\mathrm{e}}_{t+k}$, a de-embedding layer decodes it and gets the predicted next graph signal $\widehat{\mathrm{x}}_{t+k}$. This de-embedding layer is a feedforward network implemented with sparse linear layers and non-linear activation functions.

11. For simplicity, we assume that different nodes share the same auxiliary information. It is easy to generalize our model to other cases.

12. The same computations that appear in multiple iterations are performed only once.

13. In the rest of this section, the *actual values* of the graph signals $\{\mathbf{x}_{t+i} \mid i = 1, \ldots, k-1\}$ are not involved. Given this, and to simplify our equations, we use the same notations to refer to the *predicted values* of these graph signals.

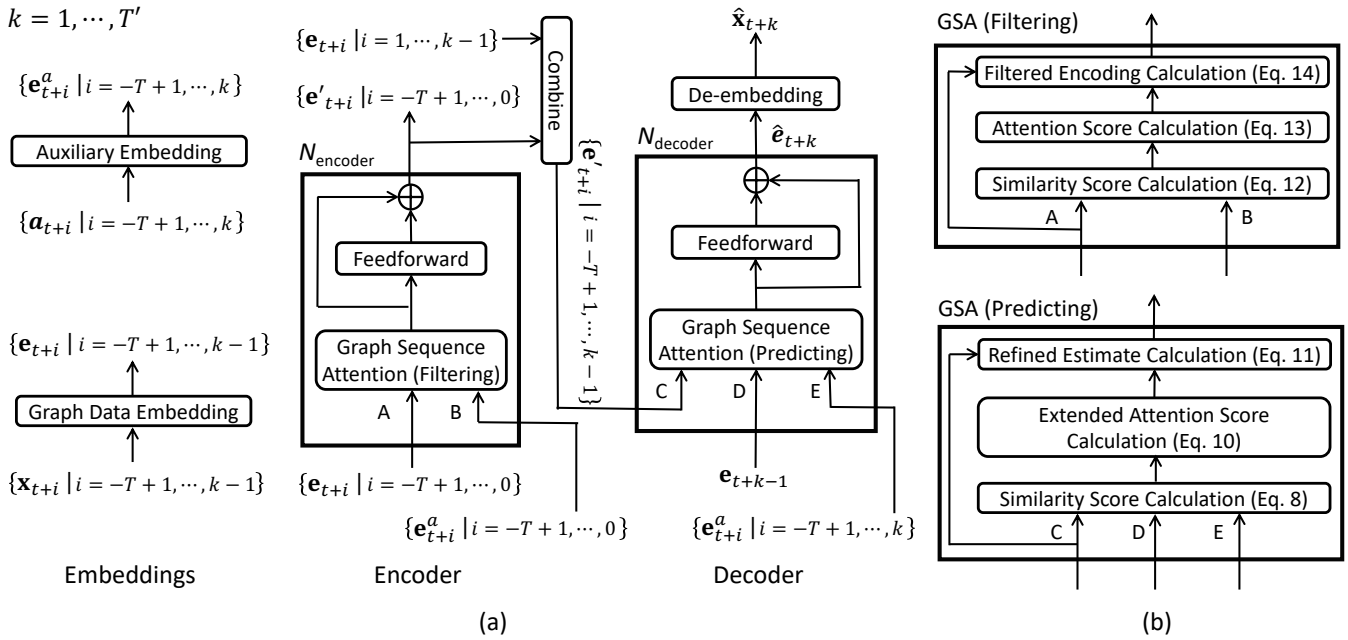


Fig. 11: Architecture of (a) GSA-Forecaster and (b) graph sequence attention (GSA) components.

4.3 Graph Sequence Attention (Predicting)

The goal of GSA (predicting) in each decoder layer is to predict the encoding of the next graph signal. To do this, it looks for the previous graph signals that show temporal dependency with the next graph signal, and it uses these previous graph signals to predict the next graph signal. It takes the following entities as input as shown in Fig. 11:

- entry C: previous data sequence $\{\mathbf{e}'_{t+i} \mid i = -T+1, \ldots, k-1\}$
- entry D: estimated encoding of the next graph signal [14] $\mathbf{e}'_{t+k}$
- entry E: auxiliary information $\{\mathbf{e}^a_{t+i} \mid i = -T+1, \ldots, k\}$

It outputs a *refined* estimate for the encoding of the next graph signal $\widehat{\mathbf{e}}_{t+k}$. The procedures for calculating $\widehat{\mathbf{e}}_{t+k}$ are described below.

First, GSA (predicting) estimates the similarity between the next graph signal $\mathbf{x}_{t+k}$ and each previous graph signal $\mathbf{x}_{t+i}$ by calculating their *similarity scores*. With such scores, we can find the previous graph signals that the next graph signal truly depends on. The similarity scores are calculated based on the following equation:

$$\begin{aligned} s^{(h)}_{t+k,\,t+i} \;=\; & \frac{1}{M}\sum_{m=0}^{M-1} w \left\langle \frac{W^{(h)}_Q \mathbf{e}'_{t+k-m}}{\left|W^{(h)}_Q \mathbf{e}'_{t+k-m}\right|}, \frac{W^{(h)}_K \mathbf{e}'_{t+i-m}}{\left|W^{(h)}_K \mathbf{e}'_{t+i-m}\right|} \right\rangle \\ & + w_A \left\langle \frac{W^{(h)}_{QA} \mathbf{e}^a_{t+k}}{\left|W^{(h)}_{QA} \mathbf{e}^a_{t+k}\right|}, \frac{W^{(h)}_{KA} \mathbf{e}^a_{t+i}}{\left|W^{(h)}_{KA} \mathbf{e}^a_{t+i}\right|} \right\rangle \\ & + w_P \left\langle \frac{W^{(h)}_{QP} \mathbf{e}^p_{k}}{\left|W^{(h)}_{QP} \mathbf{e}^p_{k}\right|}, \frac{W^{(h)}_{KP} \mathbf{e}^p_{i}}{\left|W^{(h)}_{KP} \mathbf{e}^p_{i}\right|} \right\rangle \end{aligned} \tag{8}$$

14. For the first decoder layer, the entry D input $\mathbf{e}'_{t+k}$ is equal to the initial estimate $\mathbf{e}_{t+k-1}$. For each of the other decoder layers, the entry D input $\mathbf{e}'_{t+k}$ is equal to the refined estimate made and output by the previous decoder layer and may be different from the initial estimate.

where $s^{(h)}_{t+k,\,t+i}$ is the similarity score between the next graph signal $\mathbf{x}_{t+k}$ and the previous graph signal $\mathbf{x}_{t+i}$ under head h,[15] $i = -T+M, \ldots, k$, $h = 1, \ldots, H$, M is the temporal neighborhood size, H is the number of heads; $w, w_A, w_P \in \mathbb{R}^+$ are learnable parameters; $|\cdot|$ calculates the norm of a vector; $\langle \cdot, \cdot \rangle$ is the inner product between two vectors, measuring their similarity.

The similarity score $s^{(h)}_{t+k,\,t+i}$ consists of three terms. The first term approximates the similarity between the next graph signal $\mathbf{x}_{t+k}$ and the previous graph signal $\mathbf{x}_{t+i}$ by comparing their temporal neighborhoods. Here, the *temporal neighborhood* (TN) at time $t+j$ ($j = -T+M, \ldots, k$) contains the encodings of graph signals $\mathbf{e}'_{t+j-M+1}, \cdots, \mathbf{e}'_{t+j}$ (see Fig. 12 for illustration). As the first term is based on comparing *multiple* graph signals within the two temporal neighborhoods, the inaccuracy or noise of *individual* graph signal (e.g., inaccuracy in the estimate $\mathbf{e}'_{t+k}$) has less impact on it. Therefore, it more accurately and robustly characterizes the temporal dependency between graph signals. In the first term, $W^{(h)}_Q, W^{(h)}_K \in \mathbb{R}^{\frac{d_{\text{model}}}{H} \times d_{\text{model}}}$ are learnable parameter matrices for comparing graph signals. They are implemented as sparse linear layers.

The second term of $s^{(h)}_{t+k,\,t+i}$ approximates the similarity by comparing auxiliary information $\mathbf{e}^a_{t+k}$ and $\mathbf{e}^a_{t+i}$. Intuitively, similar auxiliary information has similar impact on the graph signals and thus should be accounted for. In the second term, $W^{(h)}_{QA}, W^{(h)}_{KA} \in \mathbb{R}^{\frac{d_{\text{aux}}}{H} \times d_{\text{aux}}}$ are learnable parameter matrices implemented as fully-connected layers.

The third term of $s^{(h)}_{t+k,\,t+i}$ approximates the similarity by comparing temporal positional encodings $\mathbf{e}^p_k$ and $\mathbf{e}^p_i$. In some applications, graph signals at some relative temporal positions may often show temporal dependency (e.g., the current taxi demand usually depends on the taxi demand

15. Same as prior work [9], [25], we also compare graph signals under multiple perspectives, where a perspective is termed as a *head*.

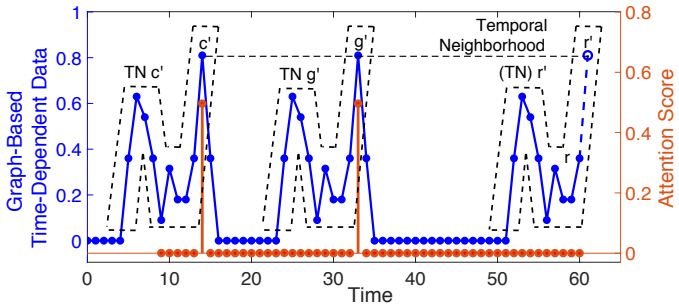


Fig. 12: Results of graph sequence attention for the example shown in Fig. 4.

a week ago). We use the learnable temporal positional encodings $\{\mathbf{e}_i^p \mid i = -T+M, \ldots, k\}$ to capture this kind of temporal dependency that often hold at some relative temporal positions. In the third term, $W_{QP}^{(h)}, W_{KP}^{(h)} \in \mathbb{R}^{\frac{d_{\text{pos}}}{H} \times d_{\text{pos}}}$ are learnable parameter matrices implemented as fully-connected layers.

After getting the similarity scores, we pass them through a softmax layer to calculate the *attention scores*, which characterize the amount of attention that is paid to each previous graph signal for predicting the next graph signal:

$$\alpha_{t+k,\,t+i}^{(h)} = \frac{\exp\left(s_{t+k,\,t+i}^{(h)}\right)}{\sum_{j=-T+M}^{k-1} \exp\left(s_{t+k,\,t+j}^{(h)}\right)} \tag{9}$$

where $\alpha_{t+k,\,t+i}^{(h)} \in (0,1)$ is the attention score for the previous graph signal $\mathbf{x}_{t+i}$, $i = -T+M, \ldots, k-1$.

To illustrate the attention scores, we perform GSA (predicting) on the example shown in Fig. 4. For simplicity, we let $M = 10, H = 1, w = 10, w_A = w_P = 0$, matrices $W_Q^{(h)} = W_K^{(h)} = I$ (i.e., identity matrix) and reuse the encoding strategy described in Eq. 6. Fig. 12 shows the attention scores. We can see that when predicting the next graph signal (data point r'), GSA (predicting) pays attention to only the previous graph signals (data points c' and g') similar to the next graph signal, successfully capturing temporal dependency.

In some cases, especially when strong non-stationarity happens, the next graph signal is not similar to any of the previous graph signals. In these circumstances, we cannot predict the next graph signal based on the previous ones that show temporal dependency with it. Instead, we should focus on the recent graph signals and try to find the pattern of how they evolve over time. We term this pattern as *the recent trend*. We use RNN to learn the recent trend and predict based on it. To this end, we extend the attention scores to reflect the relative importance between capturing temporal dependency and capturing the recent trend:

$$\alpha_{t+k,\,t+i}^{\prime(h)} = \frac{\exp\left(s_{t+k,\,t+i}^{(h)}\right)}{\sum_{j=-T+M}^{k} \exp\left(s_{t+k,\,t+j}^{(h)}\right)} \tag{10}$$

where $\alpha_{t+k,\,t+i}^{\prime(h)} \in (0,1)$ is the *extended* attention score for the previous graph signal $\mathbf{x}_{t+i}$ ($i = -T+M, \ldots, k-1$), which measures the amount of attention paid to $\mathbf{x}_{t+i}$ for capturing temporal dependency; while $\alpha_{t+k,\,t+k}^{\prime(h)} \in (0,1)$ is the *extended* attention score that measures the amount of attention paid to the recent trend. $\sum_{j=-T+M}^{k} \alpha_{t+k,\,t+j}^{\prime(h)} = 1$.

The difference between Eqs. 10 and 9 is that the denominator on the right hand side of Eq. 10 includes the term associated with $s_{t+k,\,t+k}^{(h)}$ (the similarity score between the next graph signal and itself). When the next graph signal is not similar to any of the previous graph signals, $s_{t+k,\,t+k}^{(h)}$ is much larger than other similarity scores, and, as a result, $\alpha_{t+k,\,t+k}^{\prime(h)} \approx 1$, $\alpha_{t+k,\,t+i}^{\prime(h)} \approx 0$, $i = -T+M, \ldots, k-1$, which means we should focus on the recent trend to predict.

With the extended attention scores, we calculate the refined estimate for the next graph signal $\widehat{\mathbf{e}}_{t+k}$ as:

$$\begin{aligned}
\widehat{\mathbf{e}}_{t+k} &= \mathbf{e}_{t+k}^{\prime} + W_O \left[\Delta\mathbf{e}_{t+k}^{\prime(h)} \,\Big\|\, \Delta\mathbf{e}_{t+k}^{\prime(2)} \,\Big\|\, \cdots \,\Big\|\, \Delta\mathbf{e}_{t+k}^{\prime(H)}\right] \\
\Delta\mathbf{e}_{t+k}^{\prime(h)} &= \textstyle\sum_{j=-T+1}^{k-1} \alpha_{t+k,\,t+j}^{\prime(h)} W_V^{(h)} \mathbf{e}_{t+j}^{\prime} \\
&\quad + \alpha_{t+k,\,t+k}^{\prime(h)} \text{GRU}\left(\mathbf{e}_{t+k-M+1}^{\prime}, \cdots, \mathbf{e}_{t+k-1}^{\prime}\right)
\end{aligned} \tag{11}$$

where $\Delta\mathbf{e}_{t+k}^{\prime(h)}$ is the update to the original estimate $\mathbf{e}_{t+k}^{\prime}$ under head h.

$\Delta\mathbf{e}_{t+k}^{\prime(h)}$ consists of two terms. The first term uses the extended attention score $\alpha_{t+k,\,t+i}^{\prime(h)}$ as the weight for the encoding of each previous graph signal $\mathbf{e}_{t+i}^{\prime}$, capturing the impact of temporal dependency ($W_V^{(h)} \in \mathbb{R}^{\frac{d_{\text{model}}}{H} \times d_{\text{model}}}$ is a learnable parameter matrix implemented as a sparse linear layer).

The second term reflects the impact of the recent trend, which is captured by GRU [41], a popular type of RNN. The GRU takes the encodings of graph signals in the recent temporal neighborhood $\{\mathbf{e}_{t+i}^{\prime} \mid i = k-M+1, \ldots, k-1\}$ as inputs, and it outputs its update to the original estimate of the next graph signal. Specifically, we first reset the GRU to its default state and then input $\mathbf{e}_{t+k-M+1}^{\prime}$, $\mathbf{e}_{t+k-M+2}^{\prime}$, $\cdots$, and $\mathbf{e}_{t+k-1}^{\prime}$ to the GRU one by one. After that, the GRU outputs its update. Note that, for different k ($k = 1, \cdots, T'$), the inputs to the GRU are different. Therefore, we need to repeat the above procedures (i.e., reset state and input corresponding graph signals) for each k.

We concatenate the update $\Delta\mathbf{e}_{t+k}^{\prime(h)}$ of all the heads ($\cdot\|\cdot$ represents a concatenation of two vectors) and use a learnable parameter matrix $W_O \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ (implemented as a sparse linear layer) to aggregate them. Finally, the aggregated update is added to the original estimate $\mathbf{e}_{t+k}^{\prime}$, and we obtain the final estimate $\widehat{\mathbf{e}}_{t+k}$.

4.4 Graph Sequence Attention (Filtering)

The goal of GSA (filtering) in each encoder layer is to filter out noise in the encodings of historical graph signals. GSA (filtering) inputs the following entities as shown in Fig. 11:

- entry A: the encodings of historical graph signals $\{\mathbf{e}_{t+i} \mid i = -T+1, \ldots, 0\}$
- entry B: auxiliary information $\{\mathbf{e}_{t+i}^{a} \mid i = -T+1, \ldots, 0\}$

It outputs the *filtered* encodings of historical graph signals $\{\mathbf{e}_{t+i}^{\prime} \mid i = -T+1, \ldots, 0\}$. The procedures of calculating the filtered encodings are described below.

First, GSA (filtering) estimates the similarity between the true values of any two graph signals $\mathbf{x}_{t+i}$ and $\mathbf{x}_{t+j}$

by calculating their *similarity scores*. Here, true values refer to the values of graph signals as if noise were removed, which are of course unknown. With the similarity scores, we can add graph signals with similar true values together, letting their noise cancel each other to filter out noise, as the central limit theorem suggests [38]. The similarity scores are calculated as:

$$\begin{aligned} s^{(h)}_{t+i,\,t+j} = & \frac{1}{l_2 - l_1 + 1} \sum_{m=l_1}^{l_2} w \left\langle \frac{W_Q^{(h)} \mathbf{e}_{t+i+m}}{\left| W_Q^{(h)} \mathbf{e}_{t+i+m} \right|}, \frac{W_K^{(h)} \mathbf{e}_{t+j+m}}{\left| W_K^{(h)} \mathbf{e}_{t+j+m} \right|} \right\rangle \\ & + w_A \left\langle \frac{W_{QA}^{(h)} \mathbf{e}^a_{t+i}}{\left| W_{QA}^{(h)} \mathbf{e}^a_{t+i} \right|}, \frac{W_{KA}^{(h)} \mathbf{e}^a_{t+j}}{\left| W_{KA}^{(h)} \mathbf{e}^a_{t+j} \right|} \right\rangle \\ & + w_P \left\langle \frac{W_{QP}^{(h)} \mathbf{e}^p_{i}}{\left| W_{QP}^{(h)} \mathbf{e}^p_{i} \right|}, \frac{W_{KP}^{(h)} \mathbf{e}^p_{j}}{\left| W_{KP}^{(h)} \mathbf{e}^p_{j} \right|} \right\rangle \end{aligned} \tag{12}$$

where $s^{(h)}_{t+i,\,t+j}$ is the similarity score between the true values of graph signals $\mathbf{x}_{t+i}$ and $\mathbf{x}_{t+j},\ i,\, j = -T+1, \ldots, 0, h = 1, \ldots, H$, H is the number of heads; $l_1 = -\min(M_1,\, T - 1 + \min(i,\, j))$, $l_2 = \min(M_2,\, -\max(i,\, j))$, M_1 and M_2 are temporal neighborhood parameters. Simiar to the case before, $W_Q^{(h)}, W_K^{(h)} \in \mathbb{R}^{\frac{d_{\text{model}}}{H} \times d_{\text{model}}}$, $W_{QA}^{(h)}, W_{KA}^{(h)} \in \mathbb{R}^{\frac{d_{\text{aux}}}{H} \times d_{\text{aux}}}$, $W_{QP}^{(h)}, W_{KP}^{(h)} \in \mathbb{R}^{\frac{d_{\text{pos}}}{H} \times d_{\text{pos}}}$, and $w, w_A, w_P \in \mathbb{R}^+$ are learnable parameters, and $\{\mathbf{e}^p_i \mid i = -T+1, \ldots, 0\}$ are learnable temporal positional encodings.

The similarity score $s^{(h)}_{t+i,\,t+j}$ consists of three terms. The first term approximates the similarity between the true values of the graph signals $\mathbf{x}_{t+i}$ and $\mathbf{x}_{t+j}$ by comparing their temporal neighborhoods. Here, the temporal neighborhood at time $t+n$ ($n = -T+1, \ldots, 0$) contains the encodings of graph signals $\mathbf{e}_{t+n-\min(n+T-1,\, M_1)}, \cdots, \mathbf{e}_{t+n+\min(-n,\, M_2)}$, where parameters M_1 and M_2 limit the size of the temporal neighborhood. As comparing temporal neighborhoods involves comparing *multiple* graph signals, the first term is less impacted by the noise from *individual* graph signals, and it thus more accurately and robustly reflects the similarity. The second and third terms capture the impact of auxiliary information and temporal positional information, respectively.

After obtaining the similarity scores, GSA (filtering) passes them through a softmax layer to calculate the *attention scores*:

$$\alpha^{(h)}_{t+i,\,t+j} = \frac{\exp\left(s^{(h)}_{t+i,\,t+j}\right)}{\sum_{n=-T+1}^{0} \exp\left(s^{(h)}_{t+i,\,t+n}\right)} \tag{13}$$

where the attention score $\alpha^{(h)}_{t+i,\,t+j} \in (0, 1)$ characterizes the amount of attention paid to the graph signal $\mathbf{x}_{t+j}$ for filtering the graph signal $\mathbf{x}_{t+i},\ i,\, j = -T+1, \ldots, 0$.

Finally, GSA (filtering) uses the attention scores as weights to add graph signals with similar true values together, filtering out their noise. The *filtered* encoding $\mathbf{e}'_{t+i}$ ($i = -T+1, \ldots, 0$) is computed as:

$$\begin{aligned} \mathbf{e}'_{t+i} &= \mathbf{e}_{t+i} + W_O \left[\Delta\mathbf{e}^{(h)}_{t+i} \,\middle\|\, \Delta\mathbf{e}^{(2)}_{t+i} \,\middle\|\, \cdots \,\middle\|\, \Delta\mathbf{e}^{(H)}_{t+i} \right] \\ \Delta\mathbf{e}^{(h)}_{t+i} &= \textstyle\sum_{j=-T+1}^{0} \alpha^{(h)}_{t+i,\,t+j}\, W_V^{(h)}\, \mathbf{e}_{t+j} \end{aligned} \tag{14}$$

where $\Delta\mathbf{e}^{(h)}_{t+i}$ is the update to the original encoding $\mathbf{e}_{t+i}$ under head h; $W_O \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ and $W_V^{(h)} \in \mathbb{R}^{\frac{d_{\text{model}}}{H} \times d_{\text{model}}}$ are learnable parameter matrices implemented as sparse linear layers.

5 Evaluation Setups

In this section, we describe how we evaluate the forecasting accuracy of GSA-Forecaster.

5.1 Datasets

5.1.1 NYC Taxi

Hourly Taxi Demand. We use the taxi dataset released by the New York City Taxi and Limousine Commission [32] from January 1, 2009 to June 30, 2016. This dataset contains detailed information of all the taxi rides taking place in Manhattan, New York City, in this period (7.5 years, 1.063 billion taxi rides in total). We select 471 locations in Manhattan and measure the hourly taxi demand at each location. Here, the hourly taxi demand at a location refers to the number of taxi pickups close to the location in each hour.[16] After getting the hourly taxi demand, we build a graph over the 471 locations to represent their spatial dependency by applying the theory of Gaussian Markov random fields. This way, we build a large-scale graph-based time-dependent data that contains 31 million data points (471 nodes $\times$ number of hours in 7.5 years).

Auxiliary Information. The hourly weather data in Manhattan is used as auxiliary information. Our weather dataset is from Weather Underground [42], which records the temperature, precipitation, visibility, wind speed, and the Booleans for snow, rain, and fog in Manhattan at each hour between January 1, 2009 and June 30, 2016.

Data Split. We split the data of hourly taxi demand and auxiliary information into training set, validation set, and test set, as follows:

- The training set (80% of the data):
  01/01/2009 – 12/31/2011 and 07/01/2012 – 06/30/2015
- The validation set (6.7% of the data):
  01/01/2012 – 06/30/2012
- The test set (13.3% of the data):
  07/01/2015 – 06/30/2016

5.1.2 METR-LA and PEMS-BAY

Traffic Speed. The METR-LA dataset [4] records the traffic speed collected at 207 loop detectors in the highways of Los Angeles from March 1, 2012 to June 27, 2012. The PEMS-BAY dataset [4] records the traffic speed collected at 325 sensors in the San Francisco Bay Area from January 1, 2017 to June 30, 2017. For both datasets, the traffic speed readings are aggregated into five-minute windows. METR-LA contains seven million data points (207 nodes $\times$ number of "five minutes" over almost four months);

16. We use a geometric threshold (150 meters) and a demand threshold (20 pickups per hour) to select locations from the roadmap of Manhattan (the roadmap contains 5464 locations) for measuring hourly taxi demand. The selected locations need to ensure that (1) the distance between any two selected locations is beyond the geometric threshold, and that (2) the average hourly taxi demand at each selected location (calculated after mapping the taxi pickups in Manhattan to only the selected locations) is larger than the demand threshold. This is how we select the 471 locations.

TABLE 1: Key hyperparameters of GSA-Forecaster.

Dataset	Hyperparameters
NYC Taxi	number of encoder layers $N_{\text{encoder}} = 1$, number of decoder layers $N_{\text{decoder}} = 3$, number of attention heads $H = 4$, loss function $\eta = 0.0025$, temporal neighborhood parameters for GSA (filtering) $M_1 = M_2 = 9$ (see Eq. 12), temporal neighborhood size for GSA (predicting) $M = 20$ (see Eq. 8), dimension of the graph signal encoding $d_{\text{model}} = 1884$ (471 nodes $\times$ 4 neurons/node), dimension of the auxiliary information encoding $d_{\text{aux}} = 64$, dimension of the temporal positional encoding $d_{\text{pos}} = 64$
METR-LA	$N_{\text{encoder}} = 1$, $N_{\text{decoder}} = 3$, $H = 3$, $\eta = 0.002$, $M_1 = M_2 = 5$, $M = 12$, $d_{\text{model}} = 621$ (207 nodes $\times$ 3 neurons/node), $d_{\text{pos}} = 63$
PEMS-BAY	$N_{\text{encoder}} = 1$, $N_{\text{decoder}} = 3$, $H = 4$, $\eta = 0.0025$, $M_1 = M_2 = 5$, $M = 12$, $d_{\text{model}} = 1300$ (325 nodes $\times$ 4 neurons/node), $d_{\text{pos}} = 64$

PEMS-BAY contains 17 million data points (325 nodes $\times$ number of "five minutes" in six months) — both are medium-scale datasets. We build spatial dependency graph for each dataset by applying the theory of Gaussian Markov random fields.

Auxiliary Information. Due to the difficulty of obtaining high-quality weather data at the locations of the sensors, we do not leverage auxiliary information for forecasting tasks on thest two datasets.

Data Split. For both datasets, the training set uses the first 70% of the data; the validation set uses the next 10% of the data; the test set contains the last 20% of the data.

5.2 Forecasting Task

5.2.1 Forecasting Task for NYC Taxi

Similar to prior work [9], we forecast the next three hourly taxi demands based on the taxi demand at relevant hours in the previous month and the corresponding auxiliary information. Specifically, letting the current time be t and $\mathbf{x}_t$ be hourly taxi demand, we use the following historical hourly taxi demand in our prediction:

- The past week: $\mathbf{x}_{t+i-j\times 24}$, $i = -23, \ldots, 0$, $j = 0, \ldots, 6$;
- Relevant hours on the same weekday of the past four weeks: $\mathbf{x}_{t+i-j\times 24\times 7}$, $\{i = -18, \ldots, 5,\ j = 2, \ldots, 4\}$ and $\{i = -18, \ldots, 0,\ j = 1\}$.

5.2.2 Forecasting Task for METR-LA and PEMS-BAY

We forecast the traffic speed at every five minutes in the next hour based on the traffic speed at relevant times in the previous month. Specifically, letting the current time be t and $\mathbf{x}_t$ be traffic speed measured at five-minute granularity, we use the following historical traffic speed in our prediction:[17]

- The past hour: $\mathbf{x}_{t+i}$, $i = -11, \ldots, 0$;
- Relevant times in the past week: $\mathbf{x}_{t+i-j\times 24\times 12}$, $i = -11, \ldots, 18$, $j = 1, \ldots, 6$;
- Relevant times on the same weekday of the past four weeks: $\mathbf{x}_{t+i-j\times 7\times 24\times 12}$, $i = -11, \ldots, 18$, $j = 1, \ldots, 4$;.

5.3 Metrics

Similar to prior work [4], [9], [19], our evaluation uses the root-mean-square error (RMSE) [43] and the mean absolute percentage error (MAPE) [43] to measure the accuracy of the forecasting results. RMSE characterizes the absolute error, while MAPE characterizes the relative error. Lower RMSE and MAPE mean better accuracy.

For the NYC Taxi dataset, following the practice in prior work [9], [19], we set a threshold on taxi demand when calculating MAPE: if $x_t^i < 20$ (x_t^i refers to the hourly taxi demand at location i at time t), exclude x_t^i from the calculation of MAPE.

5.4 Implementation Details

The key hyperparameters of GSA-Forecaster are in Table 1. GSA-Forecaster uses the following loss function:

$$\text{loss}\,(\cdot) = \eta \times \text{RMSE}^2 + \text{MAPE} \tag{15}$$

where a constant weight η is used such that RMSE and MAPE have almost equal contributions to the loss function.

5.5 Baseline Models

We compare GSA-Forecaster against a conventional time series model called VAR [13] and several state-of-the-art deep learning models including DCRNN [4], Graph WaveNet [22], Transformer [25], and Forecaster [9]. We briefly summarize these baseline models as below.

Vector autoregression (VAR) [13] is a conventional multivariate time series predictor. It assumes linear relations between future and past data.

DCRNN [4] is an RNN- and GCN-based model. It uses RNN to capture temporal dependency and uses graph convolutional network (GCN) to capture spatial dependency.

Graph WaveNet [22] is a CNN- and GCN-based model. It uses dilated CNN to model temporal dependency and uses GCN to model spatial dependency.

Transformer [25] is an attention mechanism-based model. It is originally proposed for NLP tasks but can be applied to forecasting graph-based time-dependent data [9]. It uses the standard attention mechanism to capture temporal dependency.

Forecaster [9] is also an attention mechanism-based model. It is built upon Transformer, leveraging its capability of capturing temporal dependency. It embeds the graph structure into its architecture to handle spatial dependency.

6 Evaluation Results

In this section, we evaluate the forecasting accuracy of GSA-Forecaster and compare it against state-of-the-art predictors.

17. If a baseline method uses a different way to select relevant times, we let it stick to its own way.

6.1 Performance on NYC Taxi

Table 2 compares GSA-Forecaster with the baseline models on accuracy of forecasting hourly taxi demand in New York City — overall accuracy and accuracy of the predictions in each of the future three hours. As deep learning models are subject to the randomness of initialization, we run each of the evaluated deep learning models (DCRNN, Graph WaveNet, Transformer, Forecaster, and GSA-Forecaster) five times and report the mean and standard deviation of their forecasting accuracy. As the table shows, VAR model has significant worse forecast accuracy compared to deep learning based models as the RMSE and MAPE of VAR are both higher than those of the deep learning models. This is because VAR assumes stationarity and linearity between the data at different timestamps and thus fails to capture temporal dependency within the data. Among the deep learning models, the baseline model Forecaster outperforms the other three baselines DCRNN, Graph WaveNet, and Transformer. This is because (1) the standard attention mechanism in Forecaster captures temporal dependency better than the RNN in DCRNN and CNN in Graph WaveNet do; (2) Forecaster captures spatial dependency (by embedding the graph structure into its architecture), while Transformer fails to do so. Our proposed GSA-Forecaster achieves the best forecasting accuracy as it further enhances Forecaster's capability to capture temporal dependency by using graph sequence attention (vs. the standard attention used by Forecaster). Compared with the best baseline Forecaster, GSA-Forecaster achieves more accurate predictions for each of the future three hours and reduces the overall RMSE and MAPE by 6.7% and 5.8%, respectively.

To provide a visual example for the performance of the models, Fig. 13 (a) – (f) show their three-hour-ahead forecasting results for the taxi demand around New York Penn Station during November 8 – 14, 2015. We compare the forecasting error of each baseline model with that of GSA-Forecaster and show their difference in Fig. 13 (g) – (k) (a positive difference means the baseline model has larger error). We can see that, overall, GSA-Forecaster has better accuracy. In this example, compared with the best baseline Forecaster, GSA-Forecaster has smaller error 63% of the time; on average, GSA-Forecaster has 13% less error.

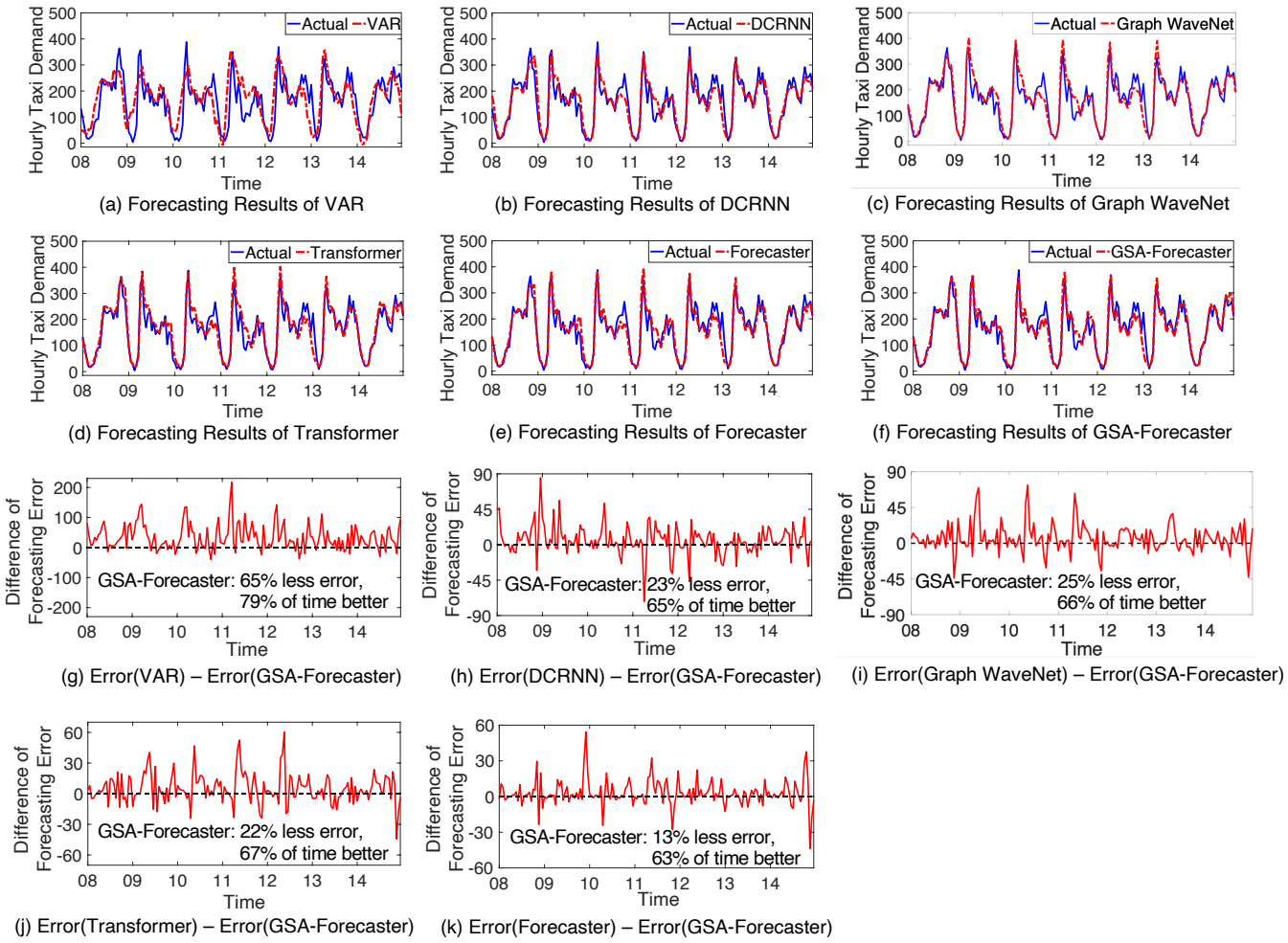


Fig. 13: Three-hour-ahead forecasting for the hourly taxi demand at a location near New York Penn Station from Sunday, November 8, 2015 at 12:00 a.m. to Saturday, November 14, 2015 at 11:59 p.m. Error (model) represents the forecasting error of a model, i.e., the absolute value of the difference between its predicted taxi demand and the actual taxi demand. "Actual" refers to the actual taxi demand. Each tick label "DD" in the x-axis refers to the beginning of a date (e.g., "08" means November 8, 2015 at 12:00 a.m.).

TABLE 2: Accuracy of GSA-Forecaster and baseline models on forecasting hourly taxi demand in New York City.

Model	Overall		Next Hour		Second Next Hour		Third Next Hour	
	RMSE	MAPE (%)	RMSE	MAPE (%)	RMSE	MAPE (%)	RMSE	MAPE (%)
VAR	12.573	33.207	11.102	30.245	12.855	33.827	13.629	35.550
DCRNN	8.742 ± 0.361	19.828 ± 0.632	8.286 ± 0.327	19.046 ± 0.619	8.813 ± 0.358	19.911 ± 0.594	9.109 ± 0.397	20.523 ± 0.700
Graph WaveNet	8.864 ± 0.042	20.298 ± 0.074	8.464 ± 0.042	19.792 ± 0.070	8.882 ± 0.057	20.271 ± 0.165	9.228 ± 0.055	20.814 ± 0.127
Transformer	8.866 ± 0.104	17.862 ± 0.263	8.434 ± 0.068	17.006 ± 0.067	8.894 ± 0.094	17.875 ± 0.262	9.251 ± 0.166	18.702 ± 0.477
Forecaster	8.412 ± 0.156	16.559 ± 0.154	7.890 ± 0.116	15.927 ± 0.136	8.484 ± 0.169	16.615 ± 0.173	8.834 ± 0.180	17.135 ± 0.181
GSA-Forecaster	**7.848 ± 0.009**	**15.600 ± 0.012**	**7.588 ± 0.007**	**15.276 ± 0.013**	**7.900 ± 0.011**	**15.657 ± 0.009**	**8.048 ± 0.011**	**15.865 ± 0.018**

TABLE 3: Accuracy of GSA-Forecaster and baseline models for forecasting *non-zero* traffic speed (following the evaluation methodology of Graph WaveNet) in Los Angeles (METR-LA) and the San Francisco Bay Area (PEMS-BAY). Results with * are referenced from [4], [22].

Dataset	Model	15 minutes		30 minutes		60 minutes	
		RMSE	MAPE (%)	RMSE	MAPE (%)	RMSE	MAPE (%)
METR-LA	VAR	7.89 *	10.2 *	9.13 *	12.7 *	10.11 *	15.8 *
	DCRNN	5.38 *	7.3 *	6.45 *	8.8 *	7.59 *	10.5 *
	Graph WaveNet	**5.15** *	**6.90** *	6.22 *	**8.37** *	7.37 *	**10.01** *
	Transformer	6.48 ± 0.13	10.10 ± 0.19	7.70 ± 0.24	11.86 ± 0.22	8.97 ± 0.18	13.72 ± 0.10
	Forecaster	5.83 ± 0.10	8.01 ± 0.09	7.30 ± 0.12	9.94 ± 0.11	8.95 ± 0.31	12.34 ± 0.38
	GSA-Forecaster	5.23 ± 0.02	7.31 ± 0.05	**6.13 ± 0.01**	8.66 ± 0.03	**7.28 ± 0.04**	10.15 ± 0.07
PEMS-BAY	VAR	3.16 *	3.6 *	4.25 *	5.0 *	5.44 *	6.5 *
	DCRNN	2.95 *	2.9 *	3.97 *	3.9 *	4.74 *	4.9 *
	Graph WaveNet	**2.74** *	**2.73** *	**3.70** *	**3.67** *	4.52 *	4.63 *
	Transformer	3.89 ± 0.02	4.68 ± 0.03	4.79 ± 0.02	5.70 ± 0.05	5.64 ± 0.08	6.62 ± 0.16
	Forecaster	3.09 ± 0.05	3.27 ± 0.04	4.21 ± 0.03	4.41 ± 0.05	5.31 ± 0.11	5.54 ± 0.01
	GSA-Forecaster	2.97 ± 0.01	3.18 ± 0.02	3.72 ± 0.01	3.96 ± 0.02	**4.29 ± 0.02**	**4.58 ± 0.02**

6.2 Performance on METR-LA and PEMS-BAY

Following the evaluation practice in Graph WaveNet [22], we first compare GSA-Forecaster with the baseline models on accuracy of forecasting *non-zero* traffic speed in Los Angeles and the Bay Area — accuracy of the predictions in future 15 minutes (i.e., short-term), 30 minutes (i.e., mid-term), and 60 minutes (i.e., long-term). As Table 3 shows, GSA-Forecaster and Graph WaveNet make the best forecasts for non-zero traffic speed among these models — GSA-Forecaster performs better in some categories (e.g., 60-minute-ahead forecasting in PEMS-BAY), while Graph WaveNet performs better in the other categories (e.g., 15-minute-ahead forecasting in METR-LA). However, the comparison in Table 3 neglects prediction for zero traffic speed, which has significant societal implications, e.g., accurately predicting zero traffic speed helps government agencies and transportation departments to take measures to reduce traffic congestion. Due to this reason, we then more comprehensively compare GSA-Forecaster with Graph Wavenet by incorporating their forecasting results for zero traffic speed into the evaluation. Table 4 shows the RMSE of their results on forecasting both zero and non-zero traffic speed.[18] As Table 4 shows, overall, GSA-Forecaster outperforms Graph WaveNet on both datasets. For example, compared with Graph WaveNet, GSA-Forecaster reduces the RMSE of long-term forecasting by 1.6% and 5.3% on the METR-LA and PEMS-BAY datasets, respectively. The performance advantage of GSA-Forecaster arises from its capacity of better capturing long-range temporal dependency.

GSA-Forecaster makes larger performance improvement over Graph WaveNet on the NYC Taxi dataset compared with what it achieves on the METR-LA and PEMS-BAY datasets. This is due to the nature of these datasets — time series in METR-LA and PEMS-BAY are less varying and diverse than those in NYC Taxi. We find that most time series in METR-LA and PEMS-BAY show the following two patterns: (1) for most of the time, data in a time series slightly fluctuate around a constant, high traffic speed; and (2) occasionally, data drops to zero or low traffic speed. For data

18. In Table 4, we do not report MAPE as it is invalid to calculate MAPE for zero traffic speed. We run both GSA-Forecaster and Graph WaveNet five times. For sanity check, we compare our results of Graph WaveNet on forecasting non-zero traffic speed with the results reported in [22] and find that they are similar.

TABLE 4: RMSE of GSA-Forecaster and Graph WaveNet for forecasting traffic speed (including both zero and non-zero traffic speed) in Los Angeles (METR-LA) and the San Francisco Bay Area (PEMS-BAY).

Dataset	Model	15 minutes	30 minutes	60 minutes
METR-LA	Graph WaveNet	21.783 ± 0.247	22.028 ± 0.206	22.318 ± 0.148
	GSA-Forecaster	**21.027 ± 0.040**	**21.418 ± 0.059**	**21.970 ± 0.083**
PEMS-BAY	Graph WaveNet	**2.790 ± 0.012**	3.761 ± 0.035	4.566 ± 0.062
	GSA-Forecaster	3.015 ± 0.013	**3.758 ± 0.011**	**4.322 ± 0.019**

TABLE 5: Models with some key features of graph sequence attention.

Model	Key Features	Implementation Details
Model 4	temporal neighborhood, GRU, auxiliary information	Based on GSA-Forecaster, remove temporal positional encoding from graph sequence attention, i.e., set $w_P = 0$ in Eqs. 8 and 12.
Model 3	temporal neighborhood, GRU	Based on Model 4, further remove auxiliary information from graph sequence attention, i.e., set $w_A = 0$ in Eqs. 8 and 12.
Model 2	temporal neighborhood	Based on Model 3, further remove GRU from graph sequence attention, i.e., in Eq. 11, set $\alpha'^{(h)}_{t+k,\,t+k} = 0$, and substitute $\alpha'^{(h)}_{t+k,\,t+j}$ with $\alpha^{(h)}_{t+k,\,t+j}$ calculated by Eq. 9.
Model 1	standard attention	Based on Model 2, set the temporal neighborhood size of graph sequence attention to 1, i.e., set $M = 1$ in Eq. 8 and $l_1 = l_2 = 0$ in Eq. 12.

with the first pattern, we expect a minor difference between the performance of decent models like GSA-Forecaster and Graph WaveNet because all these models should make good predictions for slightly fluctuating data. As for data with the second pattern, part of (though not all of) such data is caused by random factors like traffic accidents and thus is unpredictable — the accuracy of GSA-Forecaster and Graph WaveNet both suffers on predicting such unpredictable data.[19] Therefore, GSA-Forecaster improves performance over Graph WaveNet less significantly on METR-LA and PEMS-BAY than it does on NYC Taxi. The nature of these datasets is quantitatively confirmed by their entropy [44], [45] — the average entropy (bin width = 0.1) of NYC Taxi, METR-LA, and PEMS-BAY is 5.07, 3.42, and 4.18, respectively.[20] This implies that NYC Taxi is more varying and diverse than METR-LA and PEMS-BAY.

6.3 Key Features of GSA: Performance Impacts

Graph sequence attention (GSA) is the core component of GSA-Forecaster. It has the following key features: (1) temporal neighborhood to capture temporal dependency (Eqs. 8 and 12), (2) GRU to capture the recent trend of data (Eqs. 10 and 11), (3) auxiliary information (Eqs. 8 and 12), and (4) temporal positional encoding (Eqs. 8 and 12).

TABLE 6: Forecasting accuracy of the models with some key features of graph sequence attention.

Model	Overall RMSE	Overall MAPE (%)
Model 1	8.493 ± 0.016	16.698 ± 0.060
Model 2	8.119 ± 0.015	16.109 ± 0.043
Model 3	8.034 ± 0.013	15.931 ± 0.041
Model 4	7.973 ± 0.014	15.874 ± 0.033
GSA-Forecaster	7.848 ± 0.009	15.600 ± 0.012

To evaluate the impact of these key features, we construct a series of models by removing these features from GSA-Forecaster one by one. Table 5 lists the details of these models. Table 6 shows the overall RMSE and MAPE of their results on forecasting taxi demand in future three hours in New York City.

Impact of temporal neighborhood. The most important difference between graph sequence attention and standard attention is that graph sequence attention uses temporal neighborhood to better capture temporal dependency. Compared to Model 1 (standard attention, not using temporal neighborhood), Model 2 (using temporal neighborhood) reduces the overall RMSE and MAPE by 4.40% and 3.53%, respectively.

Impact of GRU. GRU helps graph sequence attention learn the recent trend of the data so that we can predict the next data based on this trend. This is especially helpful in situations where strong non-stationarity happens and no similar historical data can be leveraged to predict the next data. Compared with Model 2 (no GRU), Model 3 (with GRU) reduces the overall RMSE and MAPE by 1.05% and 1.11%, respectively.

19. This is why the RMSE of the forecasting results on METR-LA dramatically increases after including zero traffic speed in the evaluation. We observe a large amount of zero traffic speed in METR-LA. Part of them is caused by random factors and thus unpredictable. Both Graph WaveNet and GSA-Forecaster have low accuracy for predicting such data. In PEMS-BAY, the amount of zero traffic speed is less.

20. We use the following procedure to calculate the average entropy. First, we process the time series at each node by removing its mean and normalizing it by its standard deviation. Then, we divide the data in the time series into multiple bins. Each bin has a width of 0.1 (We find that using different bin widths does not change our conclusion). Next, we estimate the probability of each bin by dividing the number of data in that bin by the total number of data. Based on these probabilities, we calculate entropy [44], [45] for the time series of the node. Finally, averaging entropy over all the nodes inside a dataset, we get the average entropy of the dataset.

Impact of auxiliary information. Considering auxiliary information allows graph sequence attention to capture the impact of major exogenous factors. Compared with Model 3 (no auxiliary information), (1) Model 4 (accounting for auxiliary information) reduces the overall RMSE and MAPE by 0.76% and 0.36%, respectively, for forecasting the hourly taxi demand in the *entire* test set (as Table 6 shows); and (2) Model 4 reduces the overall RMSE by 7.0% and the overall MAPE by 6.0% for forecasting the hourly taxi demand on *rainy* or *snowy* hours in the test set. Model 4 is aware of auxiliary information, while Model 3 is not. As a result, the prediction of Model 3 implicitly assumes average weather, while the prediction of Model 4 reflects the impact of actual weather including those less frequent weather conditions (rainy or snowy). This demonstrates the benefits of considering auxiliary information in predicting rare or less frequent situations.

Impact of temporal positional encoding. Temporal positional encoding helps graph sequence attention better deal with the kind of data in which temporal dependency usually exists between some relative temporal positions. Compared with Model 4 (no temporal positional encoding), GSA-Forecaster (with temporal positional encoding) reduces the overall RMSE and MAPE by l.57% and 1.73%, respectively.

7 Related Work

Graph-based time-dependent data is a set of dependent time series where the data may show dependency within each time series (called temporal dependency) and between different time series (called spatial dependency, even if physical space is not involved). A graph is used to capture the spatial dependency: a node of the graph indexes a time series and an edge represents spatial dependency between two time series. If each time series is associated with a physical location (though it is not necessary to be so), the data is also called spatial time series [10], [11] or spatial- and time-dependent data [9]. Graph-based time-dependent data is different from temporal graphs [12], another type of graph data, as, in temporal graphs, each node of the graph is not associated with a time series, and only the edges of the graph change over time.

For forecasting graph-based time-dependent data, we propose (1) a *new* attention mechanism (graph sequence attention) that captures temporal dependency significantly better than prior attention mechanisms; (2) a *new* model (GSA-Forecaster) that achieves significantly better forecasting accuracy compared with state-of-the-art models. In this section, we review prior models and attention mechanisms and discuss the differences between them and our work.

Conventional time series predictors such as auto-regressive integrated moving average (ARIMA) [13] as well as more recent work like vector autoregression (VAR) [13] and causal graph processes [14] usually make stationarity assumptions, which are often violated in real-world data [4]. Deep learning models [4], [5], [7]–[9], [16]–[23], [25], [46], [47] generally do not make such assumptions and thus gain popularity in forecasting graph-based time-dependent data.

Most of these deep learning models [4], [8], [16]–[23] use RNN or CNN to extract temporal dependency within the data. However, both RNN and CNN do not learn well long-range temporal dependency between the data at distant time positions [25]. This is because the number of operations used by RNN and CNN to relate data at two distant time positions grows at least logarithmically with the distance between them [25].

Conversely, attention mechanisms, originally developed by the natural language processing (NLP) community, relate the data at distant positions in a sequence directly and thus are good at capturing long-range temporal dependency [25]. Transformer [25] and its extensions [26]–[30], [48] employ attention mechanisms to solve NLP tasks such as machine translation and word embedding, achieving great success. Later, attention mechanisms have been applied to other fields including image/video generation [49]–[51], object detection [51], node and graph classification [52]–[54], and recently time series forecasting [9], [46], [47], [55]. Forecaster [9] is a state-of-the-art attention mechanism-based model for forecasting graph-based time-dependent data. Forecaster is built upon the architecture of Transformer to leverage its capability of capturing temporal dependency and further uses the sparse linear layers reflecting the graph structure of the data in its architecture to learn the impact of spatial dependency.

The attention mechanisms employed by most of these prior models [9], [25]–[30], [46]–[54] share the same feature: When they relate the data at positions A and B in a sequence, they compare the data encodings only between positions A and B. In this paper, we term the attention mechanisms with this feature as the *standard attention*. As demonstrated in Section 3.3, the standard attention mechanism is not sufficient to capture temporal dependency within graph-based time-dependent data. In contrast to the standard attention mechanism, when our proposed graph sequence attention considers the data at positions A and B, it compares the data encodings not only between positions A and B but also between their temporal neighborhoods. This allows graph sequence attention to more effectively capture temporal dependency (see the example in Figure 12). Li et al. [55] also incorporate temporal neighborhoods in the design of attention mechanisms. They first perform convolutions within each temporal neighborhood to extract its aggregated features. Then, they compare the aggregated features of different temporal neighborhoods to detect temporal dependency. Different from their mechanism, our graph sequence attention directly compares the corresponding data encodings within different temporal neighborhoods to capture temporal dependency. Our mechanism does not require convolutions — its performance is not subject to the accuracy of the extracted aggregated features. Our mechanism takes auxiliary information into account while [55] does not.

As for spatial dependency, prior work [4], [7], [9], [22], [23], [47] proposes to use graph neural networks to model its impact. GSA-Forecaster follows the approach of Forecaster [9] that uses sparse linear layers, a kind of graph neural network, to account for spatial dependency.

8 Conclusion

Forecasting graph-based time-dependent data shows benefits in many applications. To achieve good forecasting

accuracy, models need to capture spatial and temporal dependencies within the data and account for helpful auxiliary information. In this paper, we propose GSA-Forecaster, a new deep learning model for forecasting graph-based time-dependent data. GSA-Forecaster employs graph sequence attention, a new attention mechanism that we propose in this paper, to capture temporal dependency. Compared with the standard attention mechanism widely adopted by prior models, our proposed graph sequence attention has significantly better ability to capture temporal dependency. Additionally, GSA-Forecaster embeds the graph structure of the data into its architecture, capturing the impact of spatial dependency. It also accounts for helpful auxiliary information in its architecture. We apply GSA-Forecaster to forecasting large-scale real-world graph-based time-dependent data — hourly taxi demand in New York City. Evaluation results show that GSA-Forecaster reduces RMSE by 6.7% and MAPE by 5.8% compared with the state-of-the-art predictors (DCRNN, Graph WaveNet, Transformer, and Forecaster). We also apply GSA-Forecaster to forecasting two medium-scale real-world graph-based time-dependent data — traffic speed in Los Angeles and the San Francisco Bay Area — and demonstrate that, overall, GSA-Forecaster forecasts more accurately than state-of-the-art predictors.

Acknowledgments

This work is partially supported by the Division of Computing and Communication Foundations, the National Science Foundation (award 1513936).

References

[1] A. Greiner, W. Semmler, and G. Gong, *The Forces of Economic Growth: A Time Series Perspective*. Princeton University Press, 2016.

[2] J. Li, K. Zhang, and Z.-P. Meng, "Vismate: Interactive Visual Analysis of Station-Based Observation Data on Climate Changes," in *IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2014, pp. 133–142.

[3] D. B. Neill, "Expectation-Based Scan Statistics for Monitoring Spatial Time Series Data," *International Journal of Forecasting*, vol. 25, no. 3, pp. 498–517, 2009.

[4] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting," in *International Conference on Learning Representations (ICLR)*, 2018, pp. 1–16.

[5] W. Zhang, P. Duan, L. T. Yang, F. Xia, Z. Li, Q. Lu, W. Gong, and S. Yang, "Resource Requests Prediction in the Cloud Computing Environment with a Deep Belief Network," *Software: Practice and Experience*, vol. 47, no. 3, pp. 473–488, 2017.

[6] Y. Li, D. Wang, S. Ghose, J. Liu, S. Govindan, S. James, E. Peterson, J. Siegler, R. Ausavarungnirun, and O. Mutlu, "SizeCap: Efficiently Handling Power Surges in Fuel Cell Powered Data Centers," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2016.

[7] W. Zhang, Y. Zhang, L. Xu, J. Zhou, Y. Liu, M. Gu, X. Liu, and S. Yang, "Modeling IoT Equipment With Graph Neural Networks," *IEEE Access*, vol. 7, pp. 32 754–32 764, 2019.

[8] W. Zhang, W. Guo, X. Liu, Y. Liu, J. Zhou, B. Li, Q. Lu, and S. Yang, "LSTM-Based Analysis of Industrial IoT Equipment," *IEEE Access*, vol. 6, pp. 23 551–23 560, 2018.

[9] Y. Li and J. M. F. Moura, "Forecaster: A Graph Transformer for Forecasting Spatial and Time-Dependent Data," in *European Conference on Artificial Intelligence (ECAI)*, 2020, pp. 1293 – 1300.

[10] P. Zhang, Y. Huang, S. Shekhar, and V. Kumar, "Correlation Analysis of Spatial Time Series Datasets: A Filter-and-Refine Approach," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2003, pp. 532–544.

[11] J. Li, S. Chen, K. Zhang, G. Andrienko, and N. Andrienko, "COPE: Interactive Exploration of Co-Occurrence Patterns in Spatial Time Series," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 8, pp. 2554–2567, 2019.

[12] O. Michail, "An Introduction to Temporal Graphs: An Algorithmic Perspective," *Internet Mathematics*, vol. 12, no. 4, pp. 239–280, 2016.

[13] H. Lütkepohl, *New Introduction to Multiple Time Series Analysis*. Springer, 2005.

[14] J. Mei and J. M. F. Moura, "Signal Processing on Graphs: Causal Modeling of Unstructured Data," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 2077–2092, 2017.

[15] A. Ziat, E. Delasalles, L. Denoyer, and P. Gallinari, "Spatio-Temporal Neural Networks for Space-Time Series Forecasting and Relations Discovery," in *IEEE International Conference on Data Mining (ICDM)*, 2017, pp. 705–714.

[16] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, and Z. Li, "Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction," in *AAAI Conference on Artificial Intelligence*, 2018, pp. 2588–2595.

[17] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-RNN: Deep Learning on Spatio-Temporal Graphs," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 5308–5317.

[18] B. Yu, H. Yin, and Z. Zhu, "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018, pp. 3634–3640.

[19] X. Geng, Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, and Y. Liu, "Spatiotemporal Multi-Graph Convolution Network for Ride-Hailing Demand Forecasting," in *AAAI Conference on Artificial Intelligence*, 2019, pp. 3656–3663.

[20] J. Zhang, Y. Zheng, and D. Qi, "Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction," in *AAAI Conference on Artificial Intelligence*, 2017, pp. 1655–1661.

[21] H. Yao, X. Tang, H. Wei, G. Zheng, and Z. Li, "Revisiting Spatial-Temporal Similarity: A Deep Learning Framework for Traffic Prediction," in *AAAI Conference on Artificial Intelligence*, 2019, pp. 5668–5675.

[22] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph WaveNet for Deep Spatial-Temporal Graph Modeling," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019, pp. 1907–1913.

[23] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[24] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, *Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies*. A Field Guide to Dynamical Recurrent Neural Networks. IEEE Press, 2001.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All You Need," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 5998–6008.

[26] Z. Dai, Z. Yang, Y. Yang, W. W. Cohen, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive Language Models beyond a Fixed-Length Context," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019, pp. 2978–2988.

[27] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 5753–5763.

[28] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019, p. 4171–4186.

[29] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving Language Understanding by Generative Pre-training," *OpenAI*, pp. 1–12, 2018.

[30] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language Models are Few-Shot Learners," *arXiv preprint arXiv:2005.14165*, pp. 1–75, 2020.

[31] H. Rue and L. Held, *Gaussian Markov Random Fields: Theory And Applications (Monographs on Statistics and Applied Probability)*. Chapman & Hall/CRC, 2005.

[32] NYC Taxi and Limousine Commission, "TLC Trip Record Data," *https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page*, 2018.

[33] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse Inverse Covariance Estimation with the Graphical Lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.

[34] L. Gu, E. P. Xing, and T. Kanade, "Learning GMRF Structures for Spatial Priors," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007, pp. 1–6.

[35] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *International Conference on Learning Representations (ICLR)*, 2017, pp. 1–14.

[36] J. Du, S. Zhang, G. Wu, J. M. F. Moura, and S. Kar, "Topology Adaptive Graph Convolutional Networks," *arXiv preprint arXiv:1710.10370*, pp. 1–13, 2017.

[37] J. Shi, M. Cheung, J. Du, and J. M. F. Moura, "Classification with Vertex-Based Graph Convolutional Neural Networks," in *Asilomar Conference on Signals, Systems, and Computers (ACSSC)*, 2018, pp. 752–756.

[38] L. Wasserman, *All of Statistics: A Concise Course in Statistical Inference*. Springer Science & Business Media, 2013.

[39] J. S. Bridle, "Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition," in *Neurocomputing*, 1990, pp. 227–236.

[40] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press Cambridge, 2016.

[41] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, p. 1724–1734.

[42] Weather Underground, "Historical Weather," *https://www.wunderground.com/*, 2018.

[43] R. J. Hyndman and A. B. Koehler, "Another Look at Measures of Forecast Accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006.

[44] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[45] ——, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, no. 4, pp. 623–656, 1948.

[46] L. Cai, K. Janowicz, G. Mai, B. Yan, and R. Zhu, "Traffic Transformer: Capturing the Continuity and Periodicity of Time Series for Traffic Forecasting," *Transactions in Geographic Information System*, vol. 24, no. 3, pp. 736–755, 2020.

[47] C. Zheng, X. Fan, C. Wang, and J. Qi, "GMAN: A Graph Multi-Attention Network for Traffic Prediction," in *AAAI Conference on Artificial Intelligence*, 2020, pp. 1234–1241.

[48] R. Koncel-Kedziorski, D. Bekal, Y. Luan, M. Lapata, and H. Hajishirzi, "Text Generation from Knowledge Graphs with Graph Transformers," in *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019, pp. 2284–2293.

[49] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, "Image Transformer," in *International Conference on Machine Learning (ICML)*, 2018, pp. 4055–4064.

[50] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating Long Sequences with Sparse Transformers," *arXiv preprint arXiv:1904.10509*, pp. 1–10, 2019.

[51] X. Wang, R. B. Girshick, A. Gupta, and K. He, "Non-Local Neural Networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7794–7803.

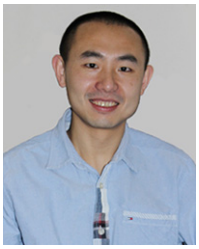
[52] B. Chen, R. Barzilay, and T. Jaakkola, "Path-Augmented Graph Transformer Network," in *Workshop on Learning and Reasoning with Graph-Structured Data (International Conference on Machine Learning Workshop)*, 2019, pp. 1–5.

[53] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph Transformer Networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 11 960–11 970.

[54] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph Attention Networks," in *International Conference on Learning Representations (ICLR)*, 2018, pp. 1–12.

[55] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 5243–5253.

Yang Li received the B.E. degree in automation from Tsinghua University, Beijing, China, in 2011, the M.S.E. degree in electrical and computer engineering from the University of Texas at Austin, Austin, TX, USA, in 2013, and the M.S. and Ph.D. degrees in electrical and computer engineering from Carnegie Mellon University (CMU), Pittsburgh, PA, USA, in 2020. He is a postdoctoral research associate at CMU. His research interests include machine learning and computer architecture.



Di Wang (Member, IEEE) earned his B.E. in computer science and technology from Zhejiang University (2005), M.S. in computer systems engineering from Technical University of Denmark (2008), and Ph.D. in computer science and engineering from the Pennsylvania State University (2014). He is currently a principal research lead with Microsoft, Redmond, WA, USA. His research spans the areas of artificial intelligence, computer systems, computer architecture, and energy efficient system design and management. He has authored more than 40 peer-reviewed papers. He received five best paper awards and two best paper nominations.

José M. F. Moura (Life Fellow, IEEE) received the Engenheiro Electrotécnico degree from the Instituto Superior Técnico (IST), Lisbon, Portugal, in 1969, and the M.Sc., E.E., and D.Sc. degrees in electrical engineering and computer science (EECS) from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 1973, 1973, and 1975, respectively.

He is currently the Philip L. and Marsha Dowd University Professor at Carnegie Mellon University (CMU), Pittsburgh, PA, USA. He was with IST as a Faculty Member and has held visiting faculty appointments at MIT and New York University (NYU), New York, NY, USA. He founded and directs a large education and research program between CMU and Portugal. Two of his patents (co-inventor A. Kavčić) are used in over four billion disk drives in 60% of all computers sold in the last 15 years worldwide and were, in 2016, the subject of a $750 million settlement between CMU and a chip manufacturer, the largest ever university verdict/settlement in the information technologies area. He has published over 550 articles and holds 16 U.S. patents. His research interests include data science and graph signal processing.

Dr. Moura is a Fellow of the American Association for the Advancement of Science (AAAS) and the U.S. National Academy of Inventors. He is a Corresponding Member of the Academy of Sciences of Portugal and a member of the U.S. National Academy of Engineering. He received the Technical Achievement Award and the Society Award from the IEEE Signal Processing Society (SPS), the CMU College of Engineering Distinguished Professor of Engineering Award, and the Doctor Honoris Causa from the University of Strathclyde, Glasgow, U.K. and from Universidade de Lisboa, Portugal. He received the Great Cross of Prince Henry bestowed to him by the President of Portugal. He was the 2019 IEEE President and Chief Executive Officer (CEO). He was Editor-in-Chief of the IEEE TRANSACTIONS ON SIGNAL PROCESSING.