# Learning Decomposed Spatial Relations for Multi-Variate Time-Series Modeling

**Yuchen Fang,**[1*] **Kan Ren,**[2] **Caihua Shan,**[2] **Yifei Shen,**[2]
**You Li,**[4] **Weinan Zhang,**[1] **Yong Yu,**[1] **Dongsheng Li** [2]

[1]Shanghai Jiao Tong University, [2]Microsoft Research Asia, [3]Central South University
{arthur_fyc, wnzhang}@sjtu.edu.cn, kan.ren@microsoft.com

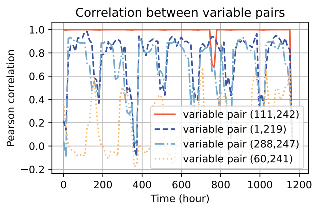
## Abstract

Modeling multi-variate time-series (MVTS) data is a long-standing research subject and has found wide applications. Recently, there is a surge of interest in modeling spatial relations between variables as graphs, i.e., first learning one static graph for each dataset and then exploiting the graph structure via graph neural networks. However, as spatial relations may differ substantially across samples, building one static graph for all the samples inherently limits flexibility and severely degrades the performance in practice. To address this issue, we propose a framework for fine-grained modeling and utilization of spatial correlation between variables. By analyzing the statistical properties of real-world datasets, a universal decomposition of spatial correlation graphs is first identified. Specifically, the hidden spatial relations can be decomposed into a prior part, which applies across all the samples, and a dynamic part, which varies between samples, and building different graphs is necessary to model these relations. To better coordinate the learning of the two relational graphs, we propose a min-max learning paradigm that not only regulates the common part of different dynamic graphs but also guarantees spatial distinguishability among samples. The experimental results show that our proposed model outperforms the state-of-the-art baseline methods on both time-series forecasting and time-series point prediction tasks.

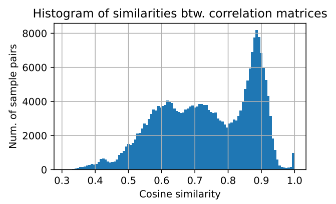
## Introduction

Multi-variate time-series (MVTS) data modeling has always been an important subject in a wide range of research domains, including healthcare (Stevenson et al. 2019), economics and finance (Lin et al. 2021), meteorology and traffic (Lai et al. 2018). One of the key observations for MVTS data is that there exist spatial relations among different variables. For example, the traffic speed on one road would be heavily affected by the traffic flow of a nearby street. Traditional methods like auto-regressive integrated moving average (ARIMA) (Box et al. 2015) and Gaussian process (GP) models (Roberts et al. 2013) either ignore spatial relations, or assume linear dependencies among variables. Deep learning methods based on recurrent neural networks (RNN) and convolutional neural networks (CNN) (Ismail Fawaz et al. 2020; Zhang et al. 2020; Shih, Sun, and Lee 2019; Lai et al. 2018) regard multi-variate information as an individual vector and rely on fully connected layers to conduct interactions between variables. However, these methods failed to exploit the spatial relationship efficiently without explicitly modeling correlations between variables, yielding high sample complexity and poor generalization ability.

[*]The work was conducted during the internship of Yuchen Fang and You Li at Microsoft Research. Correspondence to Kan Ren and Weinan Zhang.



(a) Correlation between variable pairs varies across time.



(b) Variable correlation matrices vary among samples.

Figure 1: We calculate the correlation (Benesty et al. 2009) between variables in each sample of Pems-bay traffic dataset (Li et al. 2017).

Inspired by the great success of graph neural networks (GNNs), many works utilize spatio-temporal GNNs to deal with the spatial dependencies between variables. The early attempts (Li et al. 2017; Yu, Yin, and Zhu 2017) rely on a predefined graph structure to model inter-variate dependencies, which is generally unavailable in real-world scenarios and can be inaccurate or incomplete. Recent works (Wu et al. 2019, 2020) proposed to use graph structure learning methods to discover spatial relations between variables, relaxing the requirement of predefined graphs and leading to promising results.

Nevertheless, the existing methods build a static graph for the whole dataset and ignore the fact that the relations usually vary across different samples, resulting in poor performance on some datasets. For example, suppose the sample is obtained from time-series segmentation in traffic data. Some factors (e.g., dynamic traffic flow, road maintenance or emergent accidents) affect the correlation among sensors at the crossing, which leads to dynamic variable relations in different samples. Except for these factors, some common relations, such as arterial traffic conditions, remain invariant across the samples. To verify this claim, we analyze the variable correlations on real-world data. As shown in Figure 4, we calculate the cosine similarity of the variable correlation matrices between different sample pairs. The result shows that not all the samples share the same variable relation, and it is not flexible to utilize a static graph for all the samples. Moreover, from another point of view to analyze Figure 1b, many samples have high similarities, which illustrates that purely constructing a dynamic graph for each sample would lose the shared common information and introduce additional noise.

Motivated by this example, we propose a framework for fine-grained modeling and utilization of spatial correlation between variables, which pushes the exploitation of latent spatial information to an extreme. Specifically, the hidden relation between variables is first decomposed as (1) a prior graph that depends on general information of all data and applies to all samples, and (2) a dynamic graph that varies across samples and depends on sample-level features. To ensure the diversity gain of our graph decomposition, we should simultaneously regulate the common part of different dynamic graphs and guarantee spatial distinguishability among samples. This task is shown to be highly challenging, and a novel min-max learning paradigm is proposed to tackle this challenge.

Our contribution can be summarized as follows:

- We propose a spatial relation decomposition (SRD) framework for MVTS modeling, which proposes the graph learning module to build a prior graph and dynamic graphs for sample-invariant and sample-specific dependencies between variables.
- We further introduce a min-max learning paradigm to balance the distinguishability between the information modeled by the common prior graph and the dynamic graphs for more efficient spatial relation mining.

## Related Work

**Multivariate time-series modeling** Multi-variate time-series modeling has been widely adopted in different fields and has been studied for a long time. Early approaches are mainly based on statistics, including linear models like auto-regressive (AR), moving average (MA), and auto-regressive moving average (ARMA) models. The auto-regressive integrated moving average (ARIMA) (Box et al. 2015) further generalizes the ARMA model. Gaussian process (GP) (Roberts et al. 2013) models the distribution of multi-variate time-series over continuous functions through a Bayesian approach. ROCKET (Dempster, Petitjean, and Webb 2020) extracts feature representations from time-series data using random convolutional kernels for downstream machine learning models. Shapelet-based methods (Hills et al. 2014) classify time-series through mining representative segments from datasets using statistical methods. These methods either do not consider spatial relations or assume a linear dependency among variables, thus failing to capture the intertwined dynamics of time-series data. Recently, with the rapid development of deep learning, many deep learning-based methods have been proposed to solve MVTS modeling. Ismail Fawaz et al. (2020); Zhang et al. (2020); Shih, Sun, and Lee (2019); Lai et al. (2018) all utilize recurrent neural networks or convolutional neural networks to model MVTS data. However, these methods only treat the value of multiple variables as a unified vector and then apply fully-connected layers to deal with them without explicitly mining the spatial relations in MVTS data. Wu et al. (2020); Cao et al. (2020); Li et al. (2017) represent the relations among variables as a graph and use graph neural networks for MVTS modeling. Nevertheless, these methods either rely on a predefined graph, which is noisy and inadequate, even not accessible in real-world scenarios, or extract a static graph for all the data without considering that the spatial relations may vary across different samples.

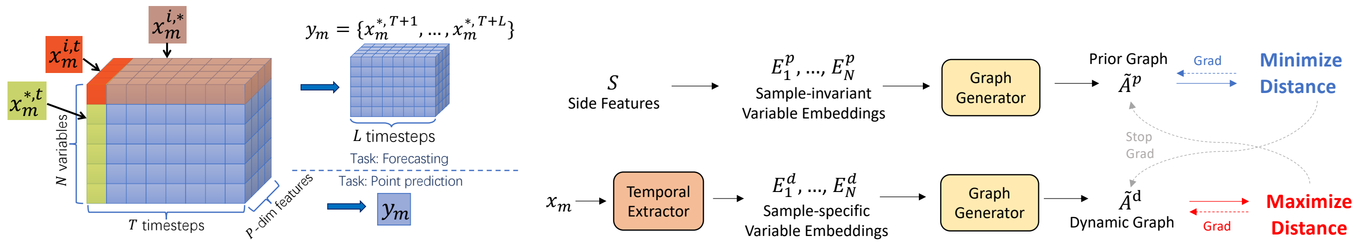
**Graph structure learning** Graph neural networks (GNN) have been widely employed across various domains to exploit the rich information inherent in the graph structure and attributes. However, most GNNs require a predefined graph structure, which is inevitable in real-world time-series data. Graph structure learning (GSL) solves the problem by jointly learning an optimized graph structure and its corresponding representations. Zhao et al. (2021); Yu et al. (2020); Li et al. (2018) employ a metric function on pairwise node embeddings to deduce edge weights. Kreuzer et al. (2021); Sun et al. (2022) utilize attention mechanisms to directly generate the adjacency matrix or amplify and attenuate existing edges. Other methods (Gao, Hu, and Guo 2020; Jin et al. 2020) directly optimize the adjacency matrix as part of the network parameters. However, these works often learn a globally shared static graph over the whole dataset, which is inflexible to represent and capture the fine-grained correlations within the data, as shown in Figure 4. In this paper, we propose to utilize a perspective of decomposed graph mining to capture the spatial correlation in MVTS data and incorporate a novel min-max training strategy to more effectively discover and utilize the latent spatial information.

## Preliminaries

We present the problem formulation of multi-variate time-series modeling and prediction in this section. Suppose there is a set of samples $\{(\boldsymbol{X}_m, \boldsymbol{y}_m)\}_{m=1}^{M}$ with size $M$. $\boldsymbol{X}_m \in \mathbb{R}^{N \times T \times P}$ represents a sample of multi-variate time-series data where $T$ is the temporal length of time series, $N$ is the number of variables and $P$ is the feature dimension, and $\boldsymbol{y}_m$ is the prediction target. As shown in Figure 2a, $\boldsymbol{x}_m^{i,t}$ is the feature of the $i$-th variable at the $t$-th timestep, $\boldsymbol{x}_m^{i,*}$ represents all the values of the $i$-th variable, and $\boldsymbol{x}_m^{*,t}$ includes all the values at the $t$-th timestep. Our goal is to find the mapping $f_{\boldsymbol{X}} \mapsto \boldsymbol{y}$ to minimize the average prediction error on the whole dataset, i.e.,

$$f^* = \arg\min_f \frac{1}{M} \sum_m \mathcal{L}_{\text{pred}}(f(\boldsymbol{X}_m), \boldsymbol{y}_m) \,. \tag{1}$$

In the different scenarios, the collection and format of $\boldsymbol{X}_m$ and $\boldsymbol{y}_m$ are various. $\{\boldsymbol{X}_m\}$ could be segments from a long time series, or many sequences from different sources. The format of $\boldsymbol{y}_m$ is decided by the prediction task. For time-series forecasting tasks, $\boldsymbol{y}_m = \{\boldsymbol{x}_m^{T+1}, \boldsymbol{x}_m^{T+2}, ..., \boldsymbol{x}_m^{T+L}\}$ is



(a) Multi-variate Time-series Data

(b) The Graph Learning Module

Figure 2: The illustration of the input MVTS data and the overall architecture of the graph learning module.

the future values of MVTS, where $L$ is a predefined forecasting horizon chosen according to the demands of environment. For point predictions tasks, $\boldsymbol{y}$ is a scalar, and the tasks can be further divided into classification and regression tasks depending on the space of the label such as event prediction or anomaly detection.

## Methodology

In this section, we design our method, a spatial relation decomposition (SRD) framework for dynamic graph mining in multi-variate time-series data.

We briefly introduce the intuition of spatial relation decomposition. As has been analyzed in Figure 1a, we have found that the correlation matrices among different variables for each sample are quite different across time and sample space. Thus, it is crucial to construct a dynamic graph for each sample. However, the complexity of building dynamic graphs is large, which can easily be affected by the noisy information within each sample, thus resulting in overfitting and poor learning effectiveness. Moreover, from Figure 1b, the spatial correlation similarity of most samples is high (peak in 0.9 cosine similarity) in the data population, which also hints the existence of the common part in the spatial information in the dataset. In order to better capture the dynamic spatial information while maintaining the common relation patterns, we propose the decomposition framework with a novel min-max learning paradigm.

In the following, we first present a novel learning process to generate the prior graph (for capturing common spatial information) and dynamic graphs (for modeling specific information of each sample) by a min-max optimization. Then we present how to integrate both temporal and spatial modules based on our learned prior and dynamic graphs for downstream prediction tasks. Note that the procedure is conducted on each sample $m$, and we omit the subscript $m$ for simplicity in the condition that the context is clear.

### Graph learning module

Based on the general information from the whole data and sample-specific information in each sample, we learn the prior graph $\boldsymbol{A}^p$ and the dynamic graph $\boldsymbol{A}^d$ to model the spatial relation among variables. The whole procedure of the graph learning module is shown in Figure 2b.

For each graph, we first learn the embedding of each variable from the input, and then train multi-layer perceptrons (MLPs) to transform these embeddings into the spatial relation among variables, i.e., adjacency matrices. A min-max paradigm is used to balance the spatial relation in different graphs.

**Learning the embedding of variables** At the beginning, we transform the extra information and time-series data of each variable into embeddings. For the prior graph, there are two situations. If the side feature exists for variables, we use it with a linear transformation to initialize $\boldsymbol{E}^p$ directly. Otherwise, we randomly initialize a learnable node embedding $\boldsymbol{E}^p \in \mathbb{R}^{N\times e}$ for each variable, where $e$ is the embedding dimension. For the dynamic graph, we extract a dynamic node embedding $\boldsymbol{E}^d \in \mathbb{R}^{N\times e}$ from each variable's time-series data in each sample as

$$\boldsymbol{z}^{i,t} = \text{Embedding}(\boldsymbol{x}^{i,t}) \tag{2}$$

$$\boldsymbol{E}^d_i = \text{TE}(\{\boldsymbol{z}^{i,t} | t \in [1,T]\}), \quad i \in [1,N]\,, \tag{3}$$

where we first map the original values $\boldsymbol{x}^{i,t}$ to a hidden representation $\boldsymbol{z}^{i,t}$, and then utilize a temporal extractor function (TE), such as Gated Recurrent Unit (GRU) (Cho et al. 2014), to further learn the embeddings of each variable.

**Generating adjacency matrices** The graph generator (GG) is to obtain the adjacency matrix $\boldsymbol{A}$ from variable embeddings $\boldsymbol{E}^p$ and $\boldsymbol{E}^d$. The resulting graph should be uni-directional as the dependencies of the variables are in chronological order, and sparse as we only aim to capture the most important spatial relations to ensure generalization ability and reduce computational cost. Following (Wu et al. 2020), the process is formulated as

$$\boldsymbol{M}^s_1 = \tanh(\alpha \boldsymbol{E}^s \Theta^s_1) \tag{4}$$

$$\boldsymbol{M}^s_2 = \tanh(\alpha \boldsymbol{E}^s \Theta^s_2) \tag{5}$$

$$\tilde{\boldsymbol{A}}^s = \text{ReLU}(\tanh(\alpha(\boldsymbol{M}^s_1 \boldsymbol{M}^{s\intercal}_2 - \boldsymbol{M}^s_2 \boldsymbol{M}^{s\intercal}_1))) \tag{6}$$

$$\boldsymbol{A}^s_i = \text{topk}(\tilde{\boldsymbol{A}}^s_i), i \in [1,N]\,, \tag{7}$$

where $\Theta^s_1$, $\Theta^s_2$ are model parameters, $\alpha$ is a hyper-parameter controlling the density of the resulting graph. The superscript $s \in \{p, d\}$ stands for prior and dynamic spatial relations, respectively. Based on the sparsity requirement of generated graphs (Jin et al. 2020), the topk$(\cdot)$ operation is used to keep only the top $k$ largest values of a vector and map the others as zero. Thus, the resulting graph is uni-directional and sparse to remove noisy, task-irrelevant edges and alleviate oversmoothing (Zhu et al. 2021).

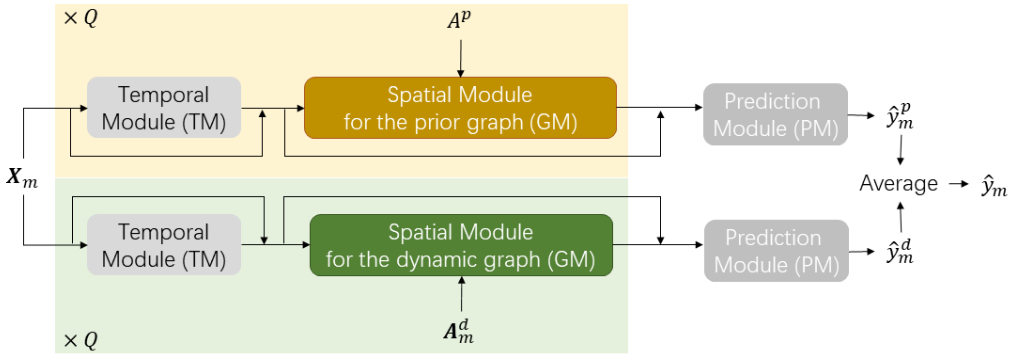


Figure 3: The overall network architecture

**Optimizing by a min-max learning paradigm** To ensure effective and efficient graph structure learning, we expect the prior graph and dynamic graph to include different spatial relations among variables. Specifically, the dependencies that are universal across the dataset should be captured by the prior graph, and the dynamic graphs should only contain the dependencies specific to a single sample inferred from its own data. To achieve this, we design a min-max learning paradigm by maximizing the discrepancy between the prior graph and dynamic graph while making sure they include meaningful spatial relations. We first define a metric $\mathcal{D}(\cdot,\cdot)$ to measure the distance between two graphs. For the **minimization phase**, we push the prior graph to *minimize* the distance compared with the dynamic graph, thus encouraging the prior graph to capture universal dependencies. For the **maximization phase**, we optimize the dynamic graph to *maximize* the distance between the prior graph, thus getting rid of the commonalities of the dynamic graph and paying attention only to the sample-specific information in the feature. The corresponding loss functions of two phases are

$$\mathcal{L}_{\text{min}} = \mathcal{D}(\tilde{\boldsymbol{A}}^p, (\tilde{\boldsymbol{A}}^d)_{\text{detach}}) \quad (8)$$

$$\mathcal{L}_{\text{max}} = -\mathcal{D}((\tilde{\boldsymbol{A}}^p)_{\text{detach}}, \tilde{\boldsymbol{A}}^d) \,, \quad (9)$$

where the subscript $(\cdot)_{detach}$ denotes stopping backpropagation of gradients, as shown in Figure 2b. Note that we use $\tilde{\boldsymbol{A}}$ rather than $\boldsymbol{A}$ to avoid the possible instability brought by top-k operation. The graph distance metric $\mathcal{D}$ is based on the Frobenius norm, formulated as

$$\mathcal{D}(\tilde{\boldsymbol{A}}^p, \tilde{\boldsymbol{A}}^d) = ||\tilde{\boldsymbol{A}}^p - \tilde{\boldsymbol{A}}^d||_F \,, \quad (10)$$

where $||\cdot||_F$ is the Frobenius norm.

## Overall network architecture

The overall architecture of our SRD framework is illustrated in Figure 3. It is composed of two branches for the prior graph and the dynamic graph, and each branch is characterized by stacking $Q$ temporal modules (TM) and spatial modules (GM). The prediction modules (PM) are connected to obtain the final prediction. Specifically, the formulation of one branch is:

$$\boldsymbol{H}_0 = \text{Embedding}(\{\boldsymbol{x}^{i,t} | i \in [1,N], t \in [1,T]\}) \quad (11)$$

$$\boldsymbol{H}_{q+1} = \text{LayerNorm}(\text{TM}_q(\boldsymbol{H}_q)) \,, q \in [0, Q-1] \quad (12)$$

$$\boldsymbol{H}_{q+1} = \text{LayerNorm}(\text{GM}_q(\boldsymbol{H}_{q+1}, \boldsymbol{A})) \quad (13)$$

$$\hat{\boldsymbol{y}} = \text{PM}(\boldsymbol{H}_Q) \,. \quad (14)$$

Similar with Equation (2), we first map the original values $\boldsymbol{x}^{i,t}$ to the initial representation $\boldsymbol{H}_0$. In the $q$-th layer, $\boldsymbol{H}_q \in \mathbb{R}^{T\times N\times D}$ are learnt by the temporal and spatial module together. Layer normalization (Ba, Kiros, and Hinton 2016) is applied after each module. After $Q$ temporal-spatial modules, we obtain $\boldsymbol{H}_Q$ and use the prediction module to get the prediction $\hat{\boldsymbol{y}}$.

The two branches both follow the above process, while different learned graph matrices $\boldsymbol{A}^p$ and $\boldsymbol{A}^d$ are utilized in the spatial module (GM) to obtain the prediction $\hat{y}^p$ and $\hat{y}^d$, respectively. The parameters are also different in the two branches. Finally, to aggregate the prediction of two branches, the final outputs have been averaged as:

$$\hat{y} = (\hat{y}^p + \hat{y}^d)/2 \,. \quad (15)$$

We describe the spatial and temporal module as follows.

**Spatial module (GM)** Given the hidden representation $\boldsymbol{H}$ and the spatial relation $\boldsymbol{A}$, we use GM$(\boldsymbol{H}, \boldsymbol{A})$ to aggregate information from different variables. In real-world MVTS data, the spatial dependencies can be strong or weak, or even non-existent. Thus, the spatial module should be able to adjust correspondingly to leverage the spatial information. To achieve this, we utilize a mix-hop graph convolution module (Abu-El-Haija et al. 2019) with a residual connection, denoted as:

$$\boldsymbol{Z}^0 = \boldsymbol{H} \quad (16)$$

$$\boldsymbol{Z}^{s+1} = \beta \boldsymbol{Z}^0 + (1-\beta)\bar{\boldsymbol{A}}\boldsymbol{Z}^s, s \in [0, S-1] \quad (17)$$

$$\boldsymbol{Z}^{\text{Final}} = [\boldsymbol{Z}^0, \boldsymbol{Z}^1, ..., \boldsymbol{Z}^S]\boldsymbol{W} + \boldsymbol{Z}^0 \,. \quad (18)$$

$S$ is the maximum depth of spatial module, $\beta$ is teleport probability and $\boldsymbol{W} \in \mathbb{R}^{SD\times D}$ are learnable parameters. Here we set $\bar{\boldsymbol{A}} = \boldsymbol{D}^{-1}(\boldsymbol{A} + \boldsymbol{I})$ where $\boldsymbol{D}_{ii} = 1 + \sum_j \boldsymbol{A}_{ij}$. $\boldsymbol{Z}^{\text{Final}}$ is the output of GM$(\boldsymbol{H}, \boldsymbol{A})$. Note that the initial representation $\boldsymbol{Z}^0$ is added to each graph convolution layer and the final output is the combination of previous layers, preventing over-smoothing and noise brought by weak spatial dependencies (Abu-El-Haija et al. 2019).

**Temporal module (TM)** For each variable, we use the temporal module to aggregate the information across the timesteps. The temporal module has a general form as $\text{TM}(\boldsymbol{H}) = f(\boldsymbol{H}) + \boldsymbol{H}$ with a residual connection, where $f(\boldsymbol{H})$ is to extract temporal information. We do not assume a specific temporal module structure and our framework can be applied to any existing temporal modeling methods which satisfy $f(\boldsymbol{H}) \in \mathbb{T} \times \mathbb{N} \times \mathbb{D}$, e.g., GRU and TCN (Bai, Kolter, and Koltun 2018), to enhance their ability of spatial relation modeling. We implement our framework with a variety of temporal module selections and present their performances in the experimental section.

**Prediction module (PM)** We implement the prediction module as a fully-connected layer whose activation and loss function depend on the specific task. For classification tasks, the softmax function $\sigma(\boldsymbol{z})_i = \frac{e^{z_i}}{\Sigma_{j=1}^{J} z_j}$ are used to generate $\hat{\boldsymbol{y}}$ and the loss function is the cross-entropy loss $\mathcal{L}_{\text{pred}}^{\text{CE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}_m) = -\boldsymbol{y}_m \log \hat{\boldsymbol{y}}$. For regression and forecasting tasks, no activation function is applied and the loss is squared error $\mathcal{L}_{\text{pred}}^{\text{SE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}_m) = (\boldsymbol{y}_m - \hat{\boldsymbol{y}})^2$. The total loss function is the combination of the prediction loss and the loss for our proposed min-max learning, calculated as

$$\begin{aligned}\mathcal{L} =& \mathcal{L}_{\text{prediction loss}} + \mathcal{L}_{\text{min-max learning loss}} \\ =& \frac{1}{m} \sum_{m=1}^{M} \Big[ \mathcal{L}_{\text{pred}}(\hat{\boldsymbol{y}}_m, \boldsymbol{y}_m) + \mathcal{L}_{\text{pred}}(\hat{\boldsymbol{y}}_m^p, \boldsymbol{y}_m) + \mathcal{L}_{\text{pred}}(\hat{\boldsymbol{y}}_m^d, \boldsymbol{y}_m) \\ & + \alpha_1 \mathcal{D}(\tilde{\boldsymbol{A}}^p, (\tilde{\boldsymbol{A}}_m^d)_{\text{detach}}) - \alpha_2 \mathcal{D}((\tilde{\boldsymbol{A}}^p)_{\text{detach}}, \tilde{\boldsymbol{A}}_m^d) \Big] \, . \end{aligned} \tag{19}$$

Here $\alpha_1$ and $\alpha_2$ are the hyper-parameters controlling the distinguishability between the prior graph and dynamic graphs. Note that we optimize $\hat{\boldsymbol{y}}$, $\hat{\boldsymbol{y}}^p$ and $\hat{\boldsymbol{y}}^d$ simultaneously to ensure the prior graph and dynamic graphs both model meaningful spatial relation information for final predictions.

## Experiments

In this section, we present the experimental settings and the corresponding results with extended investigations. The supplementary materials and codes are available online[1].

We first list four research questions (RQs) to lead the experimental discussion as follows. **RQ1:** Does the proposed spatial relation decomposition method achieve the best performance among all the compared methods? **RQ2:** Is the combination of the prior graph and dynamic graph better than utilizing a static graph or dynamic graphs only? **RQ3:** Does the min-max learning paradigm help to improve the effectiveness of our spatial relation decomposition framework? **RQ4:** With the novel learning paradigm, do the prior and dynamic graphs capture different aspects of the hidden spatial relations among variables?

[1]https://seqml.github.io/srd

### Datasets and Compared Methods

There are two kinds of tasks in multi-variate time-series modeling and prediction: forecasting and point prediction, as shown in Figure 2a. Specifically, we conduct the experiments on four datasets for forecasting tasks, and one dataset for point prediction tasks. We split time-series forecasting datasets into training set, validation set and test set following (Wu et al. 2020). And cross-validation has been conducted on the point prediction dataset. The detailed description and statistics of datasets can be referred to the appendix.

Several strong baselines have been compared on these two tasks, respectively. For a fair comparison, we repeat each experiment 3 times with different random seeds after determining the best hyper-parameters and report the average results on test sets for all the compared methods.

**Forecasting** The following datasets are used for evaluating the forecasting performances of compared methods.

- **Solar** (Lai et al. 2018): The solar power production records.
- **Electricity** (Lai et al. 2018): The hourly electricity consumption in kWh.
- **Pems-bay** (Li et al. 2017): The average traffic speed in the Bay Area.
- **Metr-la** (Li et al. 2017): The average traffic speed measured on the highways of Los Angeles County.

On these forecasting datasets, we compare our method with **ARIMA** (Box et al. 2015), the autoregressive integrated moving average model; **GP** (Roberts et al. 2013), which models the distribution of MVTS as a Gaussian process; **LSTNet** (Lai et al. 2018), that combines RNN and CNN; a transformer-based model **Autoformer** (Wu et al. 2021), which utilizes correlation-based attention mechanism for efficient time-series forecasting; GNN-based models **MTGNN** (Wu et al. 2020) and **DMSTGCN** (Han et al. 2021) who builds static graphs across all samples to capture the static spatial relations.

**Point prediction** The dataset used in the point prediction task is described below.

- **Neonatal seizure detection** (Stevenson et al. 2019) is a multi-channel electroencephalography (EEG) recording dataset from human neonates for seizure event detection.

On this dataset, we compare with **ROCKET** (Dempster, Petitjean, and Webb 2020), which utilizes random convolution kernels to extract feature vectors from time-series data and uses a ridge regression to get final predictions; **MLSTM-FCN** (Karim et al. 2019) and **InceptionTime** (Ismail Fawaz et al. 2020) both of which enhances CNN and RNN with time-series modeling blocks, i.e., squeeze-and-excitation blocks and Inception modules. We also conduct experiments with a graph-based method **DCRNN** (Tang et al. 2021), which utilizes correlation between variables to build a static graph for the seizure detection task.

Our method has two variants with two different temporal modules GRU (Cho et al. 2014) and TCN (Bai, Kolter, and Koltun 2018)) to evaluate the generalization ability.

| | Method | Metric | Electricity | | | | Solar-energy | | | | Pems-bay | | | | Metr-la | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 6 | 24 | 48 | 96 | 6 | 24 | 48 | 96 | 6 | 24 | 48 | 96 | 6 | 24 | 48 | 96 |
| no explicit spatial info. | ARIMA | RSE↓ | .522 | .534 | .564 | .599 | .202 | .365 | .588 | .589 | .532 | .548 | .562 | .612 | .575 | .742 | .889 | .902 |
| | | R2↑ | .963 | .960 | .914 | .863 | .951 | .847 | .725 | .682 | .741 | .723 | .692 | .670 | .687 | .441 | .282 | .265 |
| | GP | RSE↓ | .603 | .612 | .633 | .642 | .225 | .388 | .612 | .575 | .544 | .532 | .577 | .592 | .572 | .738 | .912 | .925 |
| | | R2↑ | .962 | .968 | .912 | .852 | .944 | .836 | .711 | .675 | .732 | .712 | .689 | .665 | .685 | .437 | .265 | .233 |
| | LSTNet | RSE↓ | .517 | .523 | .589 | .592 | .218 | .360 | .508 | .554 | .521 | .560 | .596 | .574 | .564 | .732 | .885 | .905 |
| | | R2↑ | .980 | .974 | .873 | .864 | .950 | .857 | .721 | .733 | .752 | .706 | .688 | .678 | .697 | .475 | .293 | .246 |
| | GRU | RSE↓ | .506 | .598 | .537 | .587 | .219 | .355 | .476 | .522 | .529 | .573 | .584 | .608 | .517 | .797 | .882 | .947 |
| | | R2↑ | .981 | .972 | .971 | .964 | .950 | .875 | .781 | .737 | .747 | .703 | .691 | .665 | .759 | .429 | .301 | .194 |
| | TCN | RSE↓ | .492 | .516 | .544 | .564 | .210 | .445 | .639 | .720 | .487 | .551 | .583 | .587 | .570 | .705 | .798 | .885 |
| | | R2↑ | .977 | .975 | .972 | $\underline{.965}$ | .954 | .804 | .608 | .512 | .785 | .718 | .695 | .689 | .698 | .545 | .431 | .293 |
| | Autoformer | RSE↓ | .481 | .506 | .566 | .548 | .212 | .432 | .622 | .685 | .452 | .543 | .577 | .565 | .565 | .692 | .785 | .872 |
| | | R2↑ | .980 | .977 | .975 | .963 | .960 | .852 | .791 | .701 | .782 | .711 | .689 | .668 | .762 | .548 | .411 | .282 |
| static | MTGNN | RSE↓ | .235 | .268 | .338 | .299 | .192 | .323 | .411 | .488 | .399 | $\underline{.491}$ | .582 | .351 | .452 | .678 | .769 | $\underline{.868}$ |
| | | R2↑ | .982 | .977 | .963 | .954 | .963 | .895 | .810 | .752 | .865 | .748 | .693 | .670 | **.818** | **.613** | .445 | $\underline{.309}$ |
| | DMSTGCN | RSE↓ | $\underline{.222}$ | $\underline{.267}$ | $\underline{.280}$ | **.280** | .184 | $\underline{.313}$ | $\underline{.406}$ | $\underline{.475}$ | **.364** | .520 | .578 | .342 | $\underline{.450}$ | $\underline{.659}$ | .772 | .883 |
| | | R2↑ | **.984** | $\underline{.980}$ | $\underline{.975}$ | **.972** | $\underline{.965}$ | $\underline{.906}$ | $\underline{.842}$ | $\underline{.790}$ | **.879** | .754 | .700 | .681 | .815 | .612 | .454 | .308 |
| ours | SRD-GRU | RSE↓ | .268 | .303 | .312 | .314 | **.182** | .318 | .422 | .482 | .388 | .511 | $\underline{.543}$ | $\underline{.339}$ | .466 | **.611** | **.755** | .876 |
| | | R2↑ | .978 | .911 | .901 | .887 | .963 | .889 | .809 | .764 | .849 | $\underline{.762}$ | **.732** | $\underline{.716}$ | .807 | .613 | $\underline{.455}$ | .279 |
| | SRD-TCN | RSE↓ | **.212** | **.255** | **.276** | $\underline{.292}$ | $\underline{.183}$ | **.302** | **.399** | **.469** | **.364** | **.399** | **.343** | **.302** | **.449** | .661 | $\underline{.762}$ | **.852** |
| | | R2↑ | $\underline{.983}$ | **.982** | **.978** | .962 | **.967** | **.911** | **.844** | **.804** | $\underline{.876}$ | **.772** | $\underline{.723}$ | **.720** | **.818** | **.613** | **.460** | **.321** |

Table 1: Experimental results of time-series forecasting on four datasets with different horizons $L$. The best and the second-placed results are formatted as bold font and underlined format. ↑ (↓) indicates the higher (lower) the better. All results have been aggregated from three runs with different random seeds.

- **SRD-GRU** is our proposed method with GRU (Cho et al. 2014) as the temporal module.
- **SRD-TCN** is another proposed method with TCN (Bai, Kolter, and Koltun 2018) as temporal modeling.

Moreover, the performances of original **GRU** and **TCN** are also compared in both tasks to illustrate the enhancement brought by our SRD method.

## Evaluation metrics

For time-series forecasting tasks, we use Root Relative Squared Error (RSE) and coefficient of determination ($R^2$) as the evaluation methods.

$$\text{RSE} = \sqrt{\frac{\sum_{m=1}^{M} ||\boldsymbol{y}_m - \hat{\boldsymbol{y}}_m||^2}{\sum_{m=1}^{M} ||\boldsymbol{y}_m - \bar{\boldsymbol{y}}||^2}}\ , \tag{20}$$

and

$$R^2 = \frac{1}{MNL} \sum_{m=1}^{M} \sum_{i=1}^{N} \sum_{l=1}^{L} \left[ 1 - \frac{(y_m^{i,l} - \hat{y}_m^{i,l})^2}{(y_m^{i,l} - \bar{y}^{i,l})^2} \right]\ . \tag{21}$$

Here $M$ is the size of test sets, $\bar{\boldsymbol{y}}$ is the average of $\boldsymbol{y}_m$, $y_m^{i,l}$ is the $l$-th future value of the $i$-th variable for the $m$-th sample (i.e., $x_m^{i,T+l}$), and $\bar{y}^{i,l}$ is the average of $y_m^{i,l}$ over all samples.

For the point prediction tasks, we use area under precision-recall curve (AUPRC) and area under receiver operating characteristic curve (AUROC) for evaluation.

## Experimental Results

We present the results of all the methods on two time-series modeling tasks in Table 1 and Table 2.

Before discussing the experiment results, note that, the compared methods have been divided into three groups depending on how they model spatial dependencies between variables. The first group does **not explicitly model the spatial information** in their methods, which only incorporates temporal models to capture temporal statistics and patterns. And the methods constructing a **static** graph for all the samples have been placed in the second group. The third group is the implementation of our proposed methodology based on two temporal module variants.

| | Method | Metric↑ | N-1 | N-2 | N-3 | N-4 | Avg. |
|---|---|---|---|---|---|---|---|
| no explicit spatial info. | ROCKET | AUROC | 75.7 | 76.4 | 86.0 | 85.4 | 80.9 |
| | | AUPRC | 49.2 | 43.1 | 60.5 | 59.9 | 53.2 |
| | MLSTM-FCN | AUROC | $\underline{78.3}$ | 75.6 | 85.5 | 84.0 | 80.8 |
| | | AUPRC | $\underline{51.2}$ | 48.7 | 61.4 | 61.4 | 55.7 |
| | InceptionTime | AUROC | 72.6 | 75.2 | 72.6 | 70.9 | 72.8 |
| | | AUPRC | 33.7 | 49.7 | 47.6 | 30.7 | 40.7 |
| | GRU | AUROC | 61.9 | 51.7 | 58.7 | 81.0 | 63.3 |
| | | AUPRC | 30.2 | 13.4 | 27.4 | 55.0 | 31.5 |
| | TCN | AUROC | 77.7 | 75.8 | **90.2** | $\underline{85.3}$ | $\underline{82.3}$ |
| | | AUPRC | **54.1** | 52.9 | **76.2** | $\underline{63.3}$ | $\underline{61.6}$ |
| static | DCRNN | AUROC | 73.3 | 80.7 | 84.5 | 85.3 | 81.0 |
| | | AUPRC | 42.8 | 59.0 | 64.5 | 59.2 | 56.4 |
| ours | SRD-GRU | AUROC | **80.8** | $\underline{78.0}$ | 84.2 | 83.7 | 81.7 |
| | | AUPRC | $\underline{52.6}$ | $\underline{58.2}$ | 65.2 | 53.7 | 57.4 |
| | SRD-TCN | AUROC | 75.3 | **83.4** | $\underline{87.6}$ | **86.3** | **83.2** |
| | | AUPRC | 47.0 | **60.9** | $\underline{76.1}$ | **64.9** | **62.2** |

Table 2: Results (%) on time-series point prediction over the four folds (N-1, N-2, N-3, N-4) of seizure detection dataset.

From the results in Tables 1 and 2, we can tell that (1) our proposed decomposed graph learning method has significantly enhanced the performance upon the baselines GRU and TCN. Specifically, SRD-TCN outperforms all the compared methods on most datasets, which answers **RQ1**. (2) Compared with the first group, the second and third groups where the methods incorporate explicit spatial relation modeling have performed much better, which indicates the importance of accurate spatial relation modeling. (3) By building the prior graph and dynamic graphs simultaneously, our methods achieve better performance than the methods in the second group, which only focus on static relations over the whole dataset, partially answering **RQ2**. (4) Our methods obtain a significant improvement on traffic datasets, i.e., Pems-bay and Metr-la, which might result from more complicated and varied spatial relations in traffic scenarios, making it hard to be captured by a single static graph. (5) For point prediction task shown in Table 2, our methods with decomposed relations obtain great performance no matter which temporal model has been used. Though modeling a static graph, DCRNN utilizes RNN as base model which makes its performance worse than TCN based methods.

## Extended Investigations

**Ablation study** We conduct the following ablation study on Pems-bay with SRD-TCN to show the effectiveness of the key components in our proposed method and further answer **RQ2** and **RQ3**.

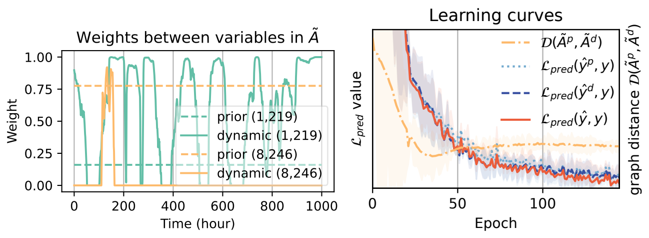
- **w/o MM**: SRD-TCN without the min-max learning paradigm, yet incorporating both the prior graph and dynamic graphs.
- **w/o DG**: SRD-TCN without dynamic graphs.
- **w/o PG**: SRD-TCN without the prior graph.

The $R^2$ results are presented in Table 3. We can tell from the results that utilizing both the prior graph and dynamic graphs (w/o MM) is superior to only using one graph (w/o DG & w/o PG), which answers our **RQ2**. Nevertheless, merely adding a graph does not bring a significant improvement, indicating that directly learning two graph structures simultaneously can bring very little gain. The performance improvement mainly comes from the proposed min-max learning paradigm, which guarantees an effective graph structure learning for both the prior graph and dynamic graphs (**RQ3**).

| Ablations \ Horizon $L$ | 6 | 24 | 48 | 96 |
|---|---|---|---|---|
| w/o MM | 0.872 | 0.750 | 0.708 | 0.695 |
| w/o DG | 0.872 | 0.750 | 0.695 | 0.674 |
| w/o PG | 0.869 | 0.750 | 0.698 | 0.673 |
| SRD-TCN | **0.876** | **0.772** | **0.723** | **0.720** |

Table 3: The $R^2$ results of ablated experiments on Pems-bay. The higher, the better.

**Analysis of edge weights in the learned graphs** To identify the different spatial relations captured by the prior graph and dynamic graphs, thus answering **RQ4**, we analyze weights between variables across different samples of Pems-bay in the learned prior graph $\tilde{\boldsymbol{A}}^p$ and dynamic graphs $\tilde{\boldsymbol{A}}_m^d$ in Figure 4a. The weights of the corresponding adjacent matrices are illustrated as dotted and solid lines, respectively. For variable pair (1, 219), the weight between them is low in the static graph, and the changing status of spatial relations has been captured by the dynamic graph. Moreover, the dynamics of the graph weights share a similar pattern with the correlation variation illustrated in Figure 1a, which proves that our learned dynamic graph does contain the changing spatial relations between variables which can hardly be modeled accurately by a static graph. For the variable pair (8, 246), the link between them is strong across



(a) The edge weights between variable pairs in the prior graph and dynamic graphs

(b) The learning situations of prediction and min-max learning losses on validation set.

Figure 4: The illustration of (a) the edge weights in the learned graphs and (b) the learning situations.

all samples, which is captured by the prior graph. The edge weight between this pair remains low in dynamic graphs on most samples to exclude the common information that has been modeled by the prior graph to reduce noise, thanks to the learning coordination brought by our min-max learning paradigm. These two cases answer **RQ4** that prior and dynamic graphs manage to capture different spatial dependencies with global and sample-wise modeling perspectives.

**Analysis of learning situations** We illustrate the learning situations and convergence process of the loss functions in Equation (19) during our min-max learning process on validation set in Pems-bay dataset, including the average value and standard deviations of prediction losses $\mathcal{L}_{\text{pred}}(\hat{\boldsymbol{y}}^p, \boldsymbol{y})$, $\mathcal{L}_{\text{pred}}(\hat{\boldsymbol{y}}^d, \boldsymbol{y})$ and $\mathcal{L}_{\text{pred}}(\hat{\boldsymbol{y}}, \boldsymbol{y})$, and the distance value between the dynamic graph and the prior graph $\mathcal{D}(\tilde{\boldsymbol{A}}^d, \tilde{\boldsymbol{A}}^p)$.

As shown in Figure 4b, all losses reach convergence with small variance, illustrating the stability of our min-max learning paradigm. At the beginning of training, $\mathcal{D}(\tilde{\boldsymbol{A}}^d, \tilde{\boldsymbol{A}}^p)$ quickly drops as both prior graph and dynamic graphs are pushed by large prediction losses to model similar meaningful spatial relations and benefit the minimization phase, as that in Equation (8). Then, the distance gets a little bit larger as the graph learning is driven by the min-max optimization paradigm to force the dynamic graph learning to model different dependencies among variables for different samples, which is achieved by the maximization phase. At the final stage, as the learning losses $\mathcal{L}_{\text{pred}}$ converge, the distance metric $\mathcal{D}$ also converges, which illustrates the learning stability of the whole system.

## Conclusion

In this paper, we propose a spatial relation decomposition framework for fine-grained modeling and utilization of spatial relations between variables in MVTS data. We achieve this by building prior and dynamic graphs for global and sample-specific spatial dependencies. A novel min-max learning paradigm has been proposed to better coordinate the graph learning process for the decomposed graphs. Our method demonstrates a superior performance to the existing MVTS modeling methods on multiple benchmark datasets.

## References

Abu-El-Haija, S.; Perozzi, B.; Kapoor, A.; Alipourfard, N.; Lerman, K.; Harutyunyan, H.; Ver Steeg, G.; and Galstyan, A. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, 21–29. PMLR.

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bai, S.; Kolter, J. Z.; and Koltun, V. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Benesty, J.; Chen, J.; Huang, Y.; and Cohen, I. 2009. Pearson correlation coefficient. In *Noise reduction in speech processing*, 1–4. Springer.

Box, G. E.; Jenkins, G. M.; Reinsel, G. C.; and Ljung, G. M. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.

Cao, D.; Wang, Y.; Duan, J.; Zhang, C.; Zhu, X.; Huang, C.; Tong, Y.; Xu, B.; Bai, J.; Tong, J.; et al. 2020. Spectral temporal graph neural network for multivariate time-series forecasting. *Advances in neural information processing systems*, 33: 17766–17778.

Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *EMNLP*.

Dempster, A.; Petitjean, F.; and Webb, G. I. 2020. ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5): 1454–1495.

Gao, X.; Hu, W.; and Guo, Z. 2020. Exploring structure-adaptive graph learning for robust semi-supervised classification. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, 1–6. IEEE.

Han, L.; Du, B.; Sun, L.; Fu, Y.; Lv, Y.; and Xiong, H. 2021. Dynamic and multi-faceted spatio-temporal deep learning for traffic speed forecasting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 547–555.

Hills, J.; Lines, J.; Baranauskas, E.; Mapp, J.; and Bagnall, A. 2014. Classification of time series by shapelet transformation. *Data mining and knowledge discovery*, 28(4): 851–881.

Ismail Fawaz, H.; Lucas, B.; Forestier, G.; Pelletier, C.; Schmidt, D. F.; Weber, J.; Webb, G. I.; Idoumghar, L.; Muller, P.-A.; and Petitjean, F. 2020. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6): 1936–1962.

Jin, W.; Ma, Y.; Liu, X.; Tang, X.; Wang, S.; and Tang, J. 2020. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 66–74.

Karim, F.; Majumdar, S.; Darabi, H.; and Harford, S. 2019. Multivariate LSTM-FCNs for time series classification. *Neural Networks*.

Kreuzer, D.; Beaini, D.; Hamilton, W.; Létourneau, V.; and Tossou, P. 2021. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34: 21618–21629.

Lai, G.; Chang, W.-C.; Yang, Y.; and Liu, H. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, 95–104.

Li, R.; Wang, S.; Zhu, F.; and Huang, J. 2018. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*.

Li, Y.; Yu, R.; Shahabi, C.; and Liu, Y. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*.

Lin, H.; Zhou, D.; Liu, W.; and Bian, J. 2021. Learning multiple stock trading patterns with temporal routing adaptor and optimal transport. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 1017–1026.

Roberts, S.; Osborne, M.; Ebden, M.; Reece, S.; Gibson, N.; and Aigrain, S. 2013. Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984): 20110550.

Shih, S.-Y.; Sun, F.-K.; and Lee, H.-y. 2019. Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, 108(8): 1421–1441.

Stevenson, N. J.; Tapani, K.; Lauronen, L.; and Vanhatalo, S. 2019. A dataset of neonatal EEG recordings with seizure annotations. *Scientific data*, 6(1): 1–8.

Sun, Q.; Li, J.; Peng, H.; Wu, J.; Fu, X.; Ji, C.; and Philip, S. Y. 2022. Graph Structure Learning with Variational Information Bottleneck. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4165–4174.

Tang, S.; Dunnmon, J.; Saab, K. K.; Zhang, X.; Huang, Q.; Dubost, F.; Rubin, D.; and Lee-Messer, C. 2021. Self-Supervised Graph Neural Networks for Improved Electroencephalographic Seizure Analysis. In *International Conference on Learning Representations*.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34: 22419–22430.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Chang, X.; and Zhang, C. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 753–763.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; and Zhang, C. 2019. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121*.

Yu, B.; Yin, H.; and Zhu, Z. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*.

Yu, D.; Zhang, R.; Jiang, Z.; Wu, Y.; and Yang, Y. 2020. Graph-revised convolutional network. In *Joint European* *conference on machine learning and knowledge discovery in databases*, 378–393. Springer.

Zhang, X.; Gao, Y.; Lin, J.; and Lu, C.-T. 2020. Tapnet: Multivariate time series classification with attentional prototypical network. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zhao, J.; Wang, X.; Shi, C.; Hu, B.; Song, G.; and Ye, Y. 2021. Heterogeneous graph structure learning for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zhu, Y.; Xu, W.; Zhang, J.; Du, Y.; Zhang, J.; Liu, Q.; Yang, C.; and Wu, S. 2021. A Survey on Graph Structure Learning: Progress and Opportunities. *arXiv e-prints*, arXiv–2103.